THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Quentin REYNAUD

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Une architecture hybride et flexible pour agents virtuels en environnement urbain :

Problématiques de la composition de comportements et de l'anticipation.

---

soutenance prévue le 29 avril 2014

devant le jury composé de :

Mme. Amal El Fallah Seghrouchni	Directeur de thèse
M. Vincent Corruble	Encadrant de thèse
M. Geber Ramalho	Rapporteur
M. Marc Cavazza	Rapporteur
M. Nicolas Bredèche	Examinateur
Mme. Domitile Lourdeaux	Examinateur
M. Jean-Yves Donnart	Encadrant industriel
M. Etienne de Sevin	Invité

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Chapitre 1

Introduction

Sommaire

1.1 Contexte et motivations

1.1.1 Contexte général et définitions

Un agent est une "entité qui peut être considérée comme percevant son environnement grâce à des capteurs et qui agit sur cet environnement via des effecteurs" [Russell 2010]. Par extension, un système multi-agents est un système composé d'un ensemble d'agents situés dans un même environnement et interagissant selon certaines relations. Dans le cadre de cette thèse nous nous intéressons à un type particulier d'agent, les **agents virtuels**, c'est-à-dire des agents situés dans un environnement (réel ou simulé) et capables de simuler le comportement d'un humain de manière autonome et crédible.

La notion de simulation est liée à celle de modèle, puisque seul un modèle de la réalité peut être simulé. Un modèle a été défini par [Minsky 1965] de la manière suivante : *Pour un observateur B, un objet A^* est un modèle d'un objet A si B peut utiliser A^* pour répondre à des questions qui l'intéressent concernant A*. Le modèle d'une simulation est donc complètement subjectif et dépendant de l'observateur et de ce qu'il souhaite observer. Pour une même réalité il existe donc différents modèles et simulations pertinents, selon les questions qui sous-tendent les besoins. Les applications de la simulation urbaine sont considérables, puisque la ville peut être le sujet d'études sociologiques, démographiques, géographiques, d'urbanisme,

d'architecture, d'économie, d'histoire, etc. toutes ayant des objectifs différents, et donc nécessitant des simulations et modèles bien particuliers.

Dans notre travail, nous allons apporter une attention particulière aux comportements des agents et à leur crédibilité. Pour définir la notion de comportement, nous prendrons une définition classique, partagée entre de nombreux courants de recherches, notamment l'intelligence artificielle, les sciences cognitives, l'approche éthologique[Seth 1998], ou le domaine de la vie intelligente (*Artificial Life*)[Langton 1997] et nous la définissons comme l'observation des actions d'un agent dans son environnement. L'étude du comportement implique donc nécessairement un observateur, et c'est seulement à travers lui que le comportement sera caractérisé.

En ce qui concerne la crédibilité, notion des plus complexes à définir en raison de sa forte subjectivité, nous considérons qu'il s'agit de la capacité à "suspendre l'incrédulité" d'un observateur[Coleridge 1817]. Nous complétons cette définition avec celle proposée par [Sengers 1999], qui est la capacité d'un système à immerger un observateur dans un environnement virtuel en lui donnant la possibilité de donner du sens aux comportements des agents. Cette notion est en lien avec celle du réalisme, c'est-à-dire le caractère de ce qui est une description objective de la réalité, qui est beaucoup plus couramment étudiée, notamment dans le domaine des jeux vidéos, du cinéma, et de la simulation.

De nombreux chercheurs ont travaillé sur la notion de crédibilité, notamment sur les agents crédibles (*Believable Agents* [Bates 1994]. Les agents crédibles sont nés du besoin de simuler des comportements humains dans des environnements virtuels peuplés de personnages animés. Dans de tels environnements, il est souvent souhaitable que les personnages virtuels adoptent un comportement qui ne soit pas nécessairement optimal en terme de résolution de problème, mais similaire au comportement qu'un humain adopterait dans une situation similaire. Le personnage donne ainsi une "illusion de vie" [Barbosa 2011].

[Vanbergue 2002] définit la ville comme "un regroupement durable mais dynamique de populations humaines dans un espace construit restreint". Ce type d'environnement s'avère particulièrement complexe à appréhender, puisqu'il est nécessaire de l'aborder à la fois sur le plan social, mais également du point de vue temporel et spatial. Par ailleurs, la coexistence en son sein de nombreuses dynamiques interdépendantes la rend elle-même fortement dynamique. Il s'ensuit que l'étude d'une telle organisation humaine pose de nombreux problèmes dans la réalité, mais qu'il est plus facile de réaliser des expérimentations dans le virtuel.

Or la simulation multi-agents est un outil puissant pour ce genre d'étude, puisqu'elle permet de "pratiquer des tests sur un substitut de la situation réelle", et donc de s'affranchir des limites du monde réel, en accédant à toutes les données et tous paramètres du problème étudié, en s'affranchissant des problèmes matériels, éthiques, ou d'échelle. D'après [Kubera 2011], la simulation multi-agents est particulièrement efficace pour comprendre ou expliquer les mécanismes à l'origine des phénomènes émergents, prédire comment un phénomène va évoluer, et construire

des vies et sociétés artificielles. [Vanbergue 2002] donne différentes approches multi-agents pour la simulation urbaine. La simulation urbaine est donc un cadre applicatif parfait pour l'étude de la crédibilité du comportement d'agents virtuels. Dans le cadre de cette thèse, nous allons donc nous concentrer sur le cas particulier des agents virtuels en environnement urbain, bien que les méthodologies et outils que nous proposerons puissent s'appliquer à un domaine plus vaste.

1.1.2 Contexte pratique

Le travail a été réalisé sous la forme d'une thèse CIFRE, se déroulant au Laboratoire d'Informatique de Paris 6, dans l'équipe Système Multi-Agents, et à THALES Training and Simulations. Elle s'est déroulée dans le cadre du projet Terra Dynamica [1], soutenu et labellisé par les pôles de compétitivité Cap Digital et Advancity, et sélectionné par le Fonds Unique Interministériel 8. Ce projet collaboratif regroupe de nombreux partenaires industriels et académiques, en particulier THALES Training & Simulation et l'UPMC (équipe Systèmes Multi-Agents du Laboratoire d'Informatique de Paris 6). Le projet vise à développer un framework d'outils permettant de modéliser et d'animer une ville virtuelle réaliste, c'est-à-dire de représenter "la vie dans la ville" : ses habitants et ses foules, ses véhicules et sa circulation. Ce projet fait suite au projet Terra Numerica qui avait pour objectif de développer des technologies de modélisation automatisée et de représentation 3D de grandes bases de données urbaines. L'apport central du projet Terra Dynamica est donc de peupler une ville virtuelle d'habitants et de véhicules réalisant leurs activités habituelles, mais également capables de réagir aux événements et de s'adapter aux évolutions de l'environnement.

FIGURE 1.1 – Aperçu du simulateur Terra Dynamica

Une des particularités les plus intéressantes d'un tel projet est la multiplicité des domaines d'application, et la variété de leurs critères d'évaluation. En effet, les applications tout particulièrement visées par le projet sont l'urbanisme, les transports, le jeu vidéo, et la sécurité. Pour les applications liées à l'urbanisme, les critères les plus importants sont par exemple la capacité à simuler des flux de véhicules similaires à ceux observés dans la réalité, ou modéliser avec précision le mobilier urbain d'un quartier. En ce qui concerne les transports, les points clés portent plutôt sur la modélisation du réseau de transports en commun, la précision des flux d'utilisateurs de ces services, et la capacité à simuler des flux de véhicules cohérents par rapport

1. www.terradynamica.com

aux données réelles. Dans le jeu vidéo, le plus intéressant est plutôt la génération d'humains virtuels, par exemple en vue de peupler de manière automatique une ville. Les applications liées à la sécurité sont quant à elles plus sensibles à la capacité à reproduire le déplacement de foules de piétons, par exemple pour des scénario de tests d'évacuation de grands bâtiments (stades, gares, etc.).

A chacune de ces thématiques correspondent des partenaires spécifiques intégrés au consortium, et des simulations potentiellement très distinctes, par exemple : modélisation d'une modification du mobilier urbain d'un quartier et simulation de son impact sur les habitants ; simulation de l'impact sur le trafic routier d'un changement du plan de circulation ; simulation d'évacuation de foules dans une gare ; etc.

C'est pour cette raison (la multiplicité des domaines d'application et des critères d'évaluation) qu'une des problématiques au cœur de notre démarche est de proposer une architecture d'agents qui soit la plus flexible et générique possible, afin de pouvoir s'adapter aux différentes contraintes. Cette spécificité, classique dans la recherche académique, est plus rare dans les applications industrielles, ce qui rend ce projet quelque peu atypique.

En dehors de ces contraintes liées au domaine applicatif, un certain nombre d'autres contraintes sont associées de manière inhérente au projet. Tout d'abord les simulations proposées devront avoir lieu en temps réel. En effet, pour un certain nombre d'applications, les utilisateurs doivent pouvoir interagir en direct avec la simulation. Par ailleurs, les simulations devront être capables de se dérouler dans des environnements de grande taille, complexes et dynamiques, tout en permettant la simulation en parallèle d'un grand nombre d'agents. Ces contraintes sont intimement liées au domaine d'application, la simulation urbaine.

Ce projet a été le fruit de la collaboration d'un grand nombre de partenaires, à la fois académiques : le Laboratoire d'Informatique de Paris 6 (LIP6), l'Équipe Cybermédia, Interactions, Transdisciplinarité, et Ubiquité (CiTu), l'Institut Français des Sciences et Technologies des Transports, de l'Aménagement et des Réseaux (IFSTTAR), le Conservatoire National des Arts et Métiers (CNAM) ; mais aussi industriels : THALES, Kylotonn, BeTomorrow, DAVI Interactive, et STAR-APIC.

1.1.3 L'environnement urbain

Le type d'environnement dans lequel se déroulent les simulations est d'une importance centrale puisqu'il pose les premières contraintes du modèle. Regardons de plus près les particularités des environnements urbains traités dans le projet, en nous basant sur une classification établie par [Russell 2010] :

- Ce sont des environnements **partiellement observables**. Les capteurs d'un agent ne lui donnent pas accès à la totalité de l'état de l'environnement. En effet, un agent ne pourra voir que ce qui se trouve devant lui, jusqu'à une certaine distance, il ne pourra entendre que les bruits provenant d'une source suffisamment proche de sa position, il n'a pas une connaissance parfaite de

l'environnement, etc. Les agents doivent donc intégrer cette part d'incertitude dans leurs réflexions.

- Comme nous l'avons vu précédemment, les espaces urbains étant densément peuplés, ces environnements sont fortement **multi-agents**, ce qui complique la tâche des agents, puisque cela pose des questions de coopération, concurrence, et communication.
- *Du point de vue d'un agent*, ces environnements paraissent **stochastiques**. En effet, l'état suivant d'un agent n'est pas complètement déterminé par son état courant et l'action qu'il exécute. L'environnement est impacté par les actions des autres agents, par des événements non entièrement connaissables par des agents (météo, embouteillages, travaux, etc.), voire par des interactions externes (interactions humaines notamment).
- Ces environnements sont **séquentiels**, c'est-à-dire qu'il ne sont pas réductibles en épisodes atomiques pendant lesquels un agent reçoit un percept puis exécute une action unique. De plus, dans les environnements séquentiels, la décision courante est susceptible d'affecter les décisions futures : les actions à court-terme peuvent avoir des conséquences à long-terme. Les agents vont donc devoir gérer une mémoire d'une manière ou d'une autre.
- Ces environnements sont fortement **dynamiques** : ils peuvent être modifiés pendant qu'un agent délibère. La réactivité des agents, ainsi que leur rapidité de délibération deviennent des éléments clés.
- Ces environnements sont à états **continus**, par exemple la vitesse et la position des agents passent par un intervalle de valeurs continues, sans à-coups dans le temps. Il en va de même pour les fluctuations de l'état interne des agents ou l'intensité des percepts.
- En fonction du point de vue, ces environnements peuvent également être considérés comme partiellement **inconnus**, c'est-à-dire que l'agent ne connaît ni les conséquences exactes de ses actions, ni comment il fonctionne (nous y reviendrons ultérieurement).

Comme on peut s'y attendre, entre tous les types d'environnements possibles, les plus complexes à prendre en compte pour un agent sont les environnements partiellement observables, multi-agents, stochastiques, séquentiels, dynamiques, continus et inconnus, c'est-à-dire exactement ceux que nous prenons en considération. Il s'avère donc que le type de simulations envisagé dans ce projet, de part la complexité de l'environnement étudié, représente à lui seul un intérêt de recherche et un défi à part entière. Les environnements rassemblant toutes ces caractéristiques sont rares[Russell 2010], et la modélisation et la simulation d'agents virtuels dans de tels environnements l'est encore plus.

1.1.4 Objectifs et motivations

Cette thèse porte sur l'intelligence décisionnelle d'agents virtuels, c'est-à-dire leur capacité à se comporter de manière intelligente, autonome, et crédible au sein

d'un environnement urbain simulé. Ce travail est au carrefour de nombreux thèmes de recherche : intelligence artificielle (IA), systèmes multi-agents, simulation, psychologie cognitive, éthologie, décision etc. qui s'intéressent à des aspects différents de la simulation de comportement.

L'intelligence artificielle s'intéresse à la fois aux processus de la pensée et du raisonnement mais aussi au comportement, et elle tente d'approcher ces deux domaines par deux biais perpendiculaires : d'une part par rapport aux performances humaines, et d'autre part par rapport à la rationalité, c'est-à-dire, un concept *idéal* de l'intelligence [Russell 2010]. Elle offre donc une multitude d'exemples d'architectures d'agents.

Les systèmes multi-agents se préoccupent plus particulièrement des relations entre entités autonomes, ce qui est utile à deux niveaux très différents. Premièrement au niveau décisionnel en considérant que l'intelligence d'un système peut naître de la coopération de nombreux agents simples. Et deuxièmement au niveau de la simulation globale, en considérant les relations entre les différents agents, leurs liens, collaboration ou compétition.

Le domaine de la simulation vise généralement un certain niveau de réalisme dans les simulations. Les apports les plus importants de ce domaine concernant souvent la qualité des déplacements, des animations, des interactions avec l'environnement, etc.

La psychologie cognitive se concentre sur les différentes fonctions psychologiques humaines. Elle donne donc des pistes intéressantes concernant la gestion des connaissances, de la mémoire, du langage, de l'intelligence, du raisonnement, de la perception, etc.

D'un autre côté, l'éthologie étudie le comportement animal, elle est très utilisée comme source d'inspiration pour des architectures centrées sur l'autonomie et l'adaptabilité des agents.

Se placer au carrefour d'autant de domaines de recherches permet de se poser des questions intéressantes pour plusieurs d'entre eux, et d'adapter des techniques de l'un pour des problématique d'un autre. Quels sont les différents types d'architecture d'agents les plus adaptés à de la simulation urbaine, étant donné la complexité des environnements qui lui sont liés ? Comment permettre à une architecture de s'adapter à d'autres types de simulations et d'environnements ? Comment permettre à une même architecture de modéliser des agents et des comportements fortement hétérogènes ? Comment augmenter l'efficacité et la crédibilité des comportements exhibés par les agents ? Comment coupler la qualité de la modélisation et de la simulation d'un agent avec le nombre d'agents simulés ? Quels sont les liens entre la crédibilité globale d'une simulation multi-agents (niveau macroscopique) et la crédibilité locale de chaque agent qui la compose (niveau microscopique) ?

Ces dernières années, le domaine de la simulation a pris une importance considérable, à tel point qu'elle fait désormais partie intégrante de nombreux autres domaines de recherche (biologie, architecture, mécanique, aéronautique, etc.). Les modèles simulés deviennent de plus en plus complexes et précis. Un des plus grands challenges de la simulation reste peut-être celui du comportement humain. Sans

chercher à imiter les mécanismes internes du cerveau, nous nous limiterons à essayer de simuler une de ses facultés primordiales : sa capacité de décision. Nous allons nous servir de cette modélisation pour simuler non pas un seul humain virtuel, mais un ensemble d'humains, connectés les uns avec les autres au sein d'une ville. En plus des questions liées à la simulation du comportement humain, nous allons donc devoir répondre à de nombreuses questions venant des interactions entre les nombreux agents, ainsi que leurs relations avec un environnement complexe et dynamique.

Le domaine de la simulation a bien souvent des objectifs de réalisme : simuler une situation ressemblant le plus possible à la réalité. Nous ne nous placerons pas dans ce contexte. Plutôt que de chercher du réalisme, nous nous intéresserons plutôt à la **crédibilité** des comportements des agents. Cette crédibilité sera jugée par des observateurs humains externes à la simulation. En effet, notre objectif principal n'est pas qu'un observateur externe ait l'impression, en regardant l'écran d'une simulation, de voir la réalité, puisque cette caractéristique dépend fortement de la qualité des graphismes et des animations. Or, nous développons une architecture d'agents, visant à être intégrée à des simulations, et simulateurs, différents. Tout ce qui est extérieur aux agents (environnement, graphisme, etc.) n'est donc pas sous notre contrôle, mais dépendant de la simulation. Il serait donc incohérent de viser à un certain niveau de réalisme.

Par contre, nous espérons que les comportements adoptés par les agents seront suffisamment cohérents et intelligents pour qu'un observateur externe puisse "y croire", et "suspende son incrédulité" (voir 1.1.1), quelle que soit la simulation dans laquelle ils évoluent.

Le premier objectif de la thèse sera de développer une architecture d'agents permettant aux agents virtuels de "percevoir" correctement leur environnement, de s'y adapter, et de répondre de manière cohérente aux divers événements perçus. Leurs comportements devront donc être réévalués dynamiquement pour prendre en compte l'évolution du contexte et de leurs connaissances. Ils devront être capables d'interagir avec leur environnement, ainsi qu'avec les autres agents de la simulation. Les différentes contraintes liées au domaine de la simulation urbaine devront être prises en compte et le choix de l'architecture devra permettre aux agents de se positionner clairement face à ces objectifs. L'architecture devra de plus être générique afin de pouvoir intégrer des théories diverses, venant de domaines différents. La généricité de l'architecture sera rendue possible par la capacité du processus décisionnel à intégrer des comportements fortement hétérogènes, ce qui rendra par ailleurs l'architecture flexible et permettra de modéliser un large spectre d'agents, et de comportements. Par ailleurs, les agents devront exhiber des comportements jugés suffisamment proches de ceux d'un être humain afin de garantir leur crédibilité.

1.2 Problématiques de recherches abordées

Cette thèse explore 3 pistes de recherches complémentaires. En premier lieu nous proposerons une architecture d'agents répondant aux contraintes liées au domaine de la simulation urbaine (temps réel, complexité de l'environnement, et grand nombre d'agents simulés en parallèle), et dans un deuxième temps nous nous intéresserons au mécanisme de composition de comportements situé au cœur du processus décisionnel, permettant à l'architecture de gagner en généricité et en flexibilité. Enfin, nous intègrerons dans le processus décisionnel des capacités de prédiction et d'anticipation, permettant d'augmenter la crédibilité des comportements.

1.2.1 Choix de l'architecture d'agents

Dans une simulation multi-agents, l'architecture permettant de modéliser les agents a une importance primordiale. En effet, le choix de l'architecture impacte fortement les caractéristiques et capacités des agents, les types de simulations possibles, etc. Ce choix doit donc être effectué après avoir analysé en détails les besoins et les caractéristiques désirées des agents. En raison des contraintes inhérentes au projet, en première analyse une architecture réactive semble tout indiquée. En effet, les avantages habituellement accordés aux agents réactifs sont leur simplicité de modélisation, leur faible besoin en ressources de calcul, leur facilité d'adaptation aux changements de l'environnement et leur autonomie. Cependant, ce type d'agent ne permet pas de répondre à l'ensemble des objectifs fixés, car la crédibilité des comportements observés est un point critique, auquel le paradigme des agents réactifs ne permet pas de répondre.

D'un autre côté les architectures dites cognitives intègrent des processus cognitifs qui augmentent la crédibilité des comportements des agents. Néanmoins, les agents cognitifs sont souvent plus gourmands en ressources de calcul, et peuvent avoir du mal à offrir la même rapidité d'adaptation aux modifications environnementales que leurs homologues réactifs.

Étant donné la complémentarité de ces deux types classiques d'agents, l'idée de coupler ces deux approches au sein d'un même agent a fait surface, c'est l'approche *hybride*[Duch 2008]. Mais l'association de ces deux paradigmes antagonistes n'est pas une tâche aisée, et il existe de nombreuses manières de la réaliser. La première contribution de cette thèse est une étude en détail de ces différentes manières, puis le développement d'une nouvelle approche, fortement générique, permettant de répondre à l'ensemble des contraintes précédemment exposées.

1.2.2 La généricité et la flexibilité du modèle nécessitent un mécanisme de composition comportementale

Au cours des dernières années, de nombreux domaines de recherche se sont intéressés à la modélisation agent (l'intelligence artificielle, la psychologie cognitive, l'éthologie, les jeux vidéo, la robotique, etc.), et ces approches différentes ont conduit au développement de techniques permettant la simulation de certains aspects bien

précis du comportement humain, en relation avec leurs problématiques. Plutôt que d'être obligés de choisir l'une de ces approches, nous préférons développer une architecture générique permettant de toutes les intégrer. Cette généricité n'est possible que si le processus décisionnel des agents est capable d'intégrer, dans un même mécanisme décisionnel, des comportements variés venant de modèles fortement hétérogènes. Il devra également être en mesure de comparer ces comportements les uns avec les autres afin de réaliser des combinaisons et des choix. La question du passage à l'échelle ne devra pas être oubliée, et ce mécanisme devra permettre de gérer un grand nombre d'agents en temps réel.

De plus, dans le cadre de la simulation urbaine, les applications visées sont variées, et les agents devant être modélisés grâce à l'architecture peuvent donc être très différents les uns des autres. C'est pourquoi développer une architecture flexible et générique est un atout majeur, permettant d'élargir le spectre des agents et des comportements modélisables. Au final, ces caractéristiques doivent permettre d'obtenir une architecture d'agents pouvant être utilisée dans un domaine beaucoup plus vaste que la simple simulation urbaine.

La deuxième contribution de cette thèse est cette capacité de notre système à intégrer dans un même processus décisionnel des comportements hétérogènes. Nous appelons **composition de comportements** cet axe de recherche.

1.2.3 Intégration de capacités d'anticipation

Puisque l'architecture est flexible et générique, il est possible d'intégrer des théories et modèles divers. Or un agent purement réactif est difficile à rendre crédible en temps qu'humain virtuel, l'absence de capacités de délibération et de planification étant vraiment pénalisante dans ce domaine. Le doter de capacités cognitives est un moyen efficace de lui faire gagner en profondeur de réflexion et en cohérence, afin de le rendre plus crédible. Il existe de nombreuses capacités considérées comme cognitives, mais une d'entre elles, l'anticipation, parait plus importante que les autres à tel point que [Pezzulo 2008] déclare que seuls les systèmes cognitifs dotés d'un mécanisme d'anticipation peuvent être crédibles, adaptatifs et interagir correctement avec leur environnement et les autres systèmes autonomes. Il s'avère en effet que l'anticipation est une capacité dans laquelle les humains se distinguent, et qu'un excellent moyen de permettre à un observateur de donner du sens au comportement d'un agent virtuel est de lui faire exhiber des comportements d'anticipation.

De plus, en prenant en compte une prédiction des états futurs de lui-même et de son environnement, cette capacité permet à un agent d'améliorer grandement l'efficacité de ses comportements, son adaptation aux modifications de contexte, et la précision de ses plans. Par ailleurs, l'ajout de ce mécanisme dans le processus décisionnel des agents ne devra pas remettre en cause la capacité du système à passer à l'échelle.

La troisième contribution de cette thèse est l'intégration dans une architecture générique d'un processus d'anticipation, ainsi que les réponses à de nombreuses questions telles que : quoi anticiper ? Comment coupler anticipation et généricité

de l'architecture ? Jusqu'à quel point est-il pertinent d'anticiper ? Quel est le lien entre anticipation, efficacité et crédibilité ?

1.3 Organisation du document

Dans le prochain chapitre, nous présenterons en détail l'état de l'art des différents domaines de recherche abordés dans le cadre de cette thèse, c'est-à-dire tout d'abord les architectures d'agents, puis la notion de composition de comportement, ensuite les processus d'anticipation, et enfin les questions relatives au passage à l'échelle. Les trois chapitres suivants seront centrés sur chacune des trois principales pistes de recherche de la thèse ayant abouti aux contributions principales.

Le troisième chapitre est focalisé sur l'architecture proposée : c'est la première contribution de ce travail. Nous commencerons par présenter le vocabulaire utilisé tout au long du document, ainsi que les différents formalismes structurant notre travail. Nous présenterons ensuite l'architecture décisionnelle générique et notamment son modèle de principe. Nous nous attarderons également sur les notions d'agent et d'individualité qui sont centrales dans toutes les simulations multi-agents. Pour clore ce chapitre, nous détaillerons l'implémentation de cette architecture générique, ainsi que les différents modules de haut-niveaux intégrés.

Dans le quatrième chapitre, c'est le travail concernant le processus décisionnel et le mécanisme de composition comportementale qui seront étudiés. Nous verrons tout d'abord comment ce travail trouve sa source dans différents travaux de l'état de l'art, puis nous nous intéresserons aux différentes phases du processus, à savoir l'intégration des propositions, leur décomposition, puis la création des différents plans et la recherche de leurs meilleures instanciations possibles, et enfin la sélection du meilleur plan. L'algorithme au cœur du processus sera détaillé, et un exemple de son déroulement sera donné. Avant de conclure, nous nous attarderons sur les mesures ayant été prises afin d'assurer que ce mécanisme est capable de passer à l'échelle.

La chapitre 5 traite de la troisième contribution de la thèse : le processus d'anticipation intégré dans l'architecture, permettant aux agents d'avoir un caractère cognitif certain. Nous commencerons par détailler ce qui distingue ce processus des mécanismes d'anticipation plus classiques. Un soin particulier sera apporté à la description des modèles prédictifs utilisés par le processus. Après avoir présenté l'algorithme, nous nous attacherons à discuter des principaux paramètres du processus, à savoir sa fréquence ainsi que son horizon temporel. Après avoir donné un exemple d'exécution du processus, nous montrerons comment l'ajout d'un module cognitif capable d'anticiper le futur des agents à partir de prédictions est compatible avec la contrainte liée au passage à l'échelle.

Le chapitre 6 sera consacré aux différentes évaluations réalisées afin de valider les différents composants développés au cours de la thèse. Ces évaluations reprendront donc, dans l'ordre, les modèles présentés dans les trois chapitres précédents, c'est-à-dire d'abord l'architecture d'agents, puis la décision et le mécanisme de composition

de comportements, et enfin le processus d'anticipation. Les évaluations sont composées de deux types très différents d'expérimentations : objectives et subjectives. Les évaluations objectives consistent à mesure expérimentalement des valeurs caractéristiques d'un modèle et de vérifier de manière objective leur signification. Dans des expérimentations subjectives par contre, nous allons faire appel à des participants, des observateurs externes du système, et nous allons leur demander de donner leur propre appréciation sur les comportements observés.

Dans le chapitre 7, nous donnerons quelques détails supplémentaires concernant la mise en œuvre du modèle. Il s'agira principalement de donner des informations sur le contexte de notre travail, c'est-à-dire le logiciel développé dans le cadre du projet dans lequel s'est déroulée la thèse. Nous présenterons aussi différentes interfaces ou fichiers de paramètres permettant de mieux comprendre la manière dont fonctionne le système. Les informations nécessaires à l'intégration de nouveaux modules de haut-niveau seront également données.

Pour conclure ce document, nous ferons une synthèse de l'ensemble du travail réalisé, et nous donnerons quelques perspectives majeures restant à explorer à la suite de nos recherches.

Chapitre 2

État de l'art

Sommaire

Ce chapitre a pour objectif de présenter les différents domaines et travaux liés aux champs de recherche de cette thèse. Il est organisé en 5 parties. Dans une première partie introductive, nous présenterons les domaines de la simulation multi-agents, de la simulation du comportement humain, et de la simulation urbaine, puisqu'il s'agit à la fois du cadre général et du cadre applicatif de nos recherches. Les trois suivantes sont liées aux pistes de recherche majeures de la thèse, c'est-à-dire tout d'abord les architectures d'agents, puis les mécanismes de composition comportementale, et enfin les processus d'anticipation. Enfin, la quatrième partie s'intéressera à la question du passage à l'échelle, qui est une contrainte très importante, et qui impacte l'ensemble de nos pistes de recherche.

2.1 Introduction

Le domaine de la simulation est vaste, nous allons nous intéresser en particulier à trois sous-parties de cet ensemble : la simulation multi-agents, la simulation de comportements humains, et la simulation urbaine.

2.1.1 Simulation multi-agents

La simulation multi-agents est un domaine de recherche très actif depuis de nombreuses années [Drogoul 1993, Drogoul 2003]. On nomme simulation la démarche scientifique qui consiste à réaliser une reproduction artificielle, appelée modèle, d'un phénomène réel que l'on désire étudier, à observer le comportement de cette reproduction lorsqu'on en fait varier certains paramètres, et à en induire ce qui se passerait dans la réalité sous l'influence de variations analogues [Drogoul 1993]. Grâce à sa capacité à traiter des types d'agents très différents, elle touche un large spectre de domaines d'application.

La simulation multi-agents met en jeu plusieurs types de problématiques. Par exemple celle liée à l'action, c'est-à-dire comment les agents agissent sur leur environnement. Mais aussi celle de la représentation du monde, et enfin celle de l'apprentissage et de l'adaptation des agents à leur environnement.

La recherche dans ce domaine a produit de nombreuses plates-formes de modélisation d'agents, dont le but est de fournir des outils à des modélisateurs pour leur permettre de créer une simulation mettant en relation plusieurs agents.

Netlogo [Tisue 2004], est un langage de programmation multi-agents et de modélisation de l'environnement pour la simulation de phénomènes (naturels ou sociaux) complexes. On peut également citer les approches **GAMA** [Amouroux 2009], **Swarm**, **Mason**, et **Repast** ([Railsback 2006]).

2.1.2 Simulation du comportement humain

La simulation du comportement humain met en jeu de nouvelles problématiques liées à la complexité des processus décisionnels humains. Cependant mener de telles études permet de répondre à des problématiques très variées.

Par exemple **SMACH** (Simulation Multi-Agents du Comportement Humain) [Amouroux 2013], s'intéresse par exemple à la simulation de l'activité humaine d'un foyer, dans le but de prévoir sa consommation électrique et ainsi éviter les pics de consommation. Ce travail aborde également la problématique de la simulation participative, qui permet à des humains de prendre le contrôle d'un avatar dans la simulation, ce qui permet de corriger son comportement ou de le valider.

On peut également citer des travaux portant sur d'autres types d'applications : simulation d'évacuation d'urgence de grands bâtiments [Pan 2006], simulation de

coopération d'agents au sein d'une équipe de travail [Miranda 2011], simulations militaires [Pew 1998], etc.

2.1.3 Simulation urbaine

Le domaine de la simulation urbaine met bien sûr en jeu à la fois des problématiques liées à la simulation multi-agents, et à la simulation de comportements humains. Derrière la problématique de la simulation multi-agents se cache souvent celle de l'auto-organisation des agents, une piste de recherche étudiée dans [Drogoul 1993]. Mais la simulation urbaine pousse plus loin cette problématique, puisqu'elle fait interagir des humains virtuels dans un environnement possédant lui-même une organisation complexe [Vanbergue 2002].

Cependant, et c'est une particularité du domaine, la granularité des simulations urbaines est fortement hétérogène. Notre approche dans laquelle chaque agent représente un humain virtuel évoluant dans une ville n'est pas la seule possible. Dans des domaines plus éloignés de l'Intelligence Artificielle, par exemple l'urbanisme, un agent peut représenter non pas un personnage virtuel, mais un ensemble beaucoup plus vaste (quartier, ville, région, etc.) [Batty 2007, Batty 2013]. Les interactions étudiées se produisent à une échelle beaucoup plus importante (par exemple entre des villes voisines), ce qui permet d'observer d'autres comportements (par exemple des flux migratoires, des densités de population, etc.).

2.2 Architectures d'agents

2.2.1 Introduction

Dans toute simulation multi-agents, l'architecture d'agents a un rôle central. La structure de l'architecture contraint les agents modélisés à respecter le formalisme utilisé. Il est donc primordial d'adapter l'architecture en fonction des objectifs que les agents devront remplir. De nombreux travaux ont déjà été menés dans ce domaine, et de grands types d'architectures d'agents ont été développés. Nous allons ici donner un aperçu de ces différents modèles, en indiquant ce qui, d'après nous, représente des points forts ou des faiblesses par rapport à nos contraintes et objectifs.

Il n'existe pas de classification parfaite des architectures d'agents, et aucune ne fait l'unanimité [Wooldridge 1995, Bryson 2000, Duch 2008]. En effet, selon le point de vue, une architecture peut être considérée comme faisant partie de telle ou telle catégorie d'architectures. Qui plus est, les définitions exactes de ces catégories varient selon les domaines de recherche. Nous n'allons donc pas chercher à proposer la classification la plus traditionnelle possible, ni la plus orthodoxe par rapport à un domaine en particulier, mais plutôt à axer cette classification en rapport avec notre problématique.

Nous rappelons les points essentiels sur lesquels nous souhaitons que notre architecture se positionne :

- Généricité du modèle afin de pouvoir intégrer des composants issus de théories ou systèmes hétérogènes.
- Flexibilité des comportements et des agents modélisables afin de permettre la simulation d'un grand nombre de scénarios appartenant à des domaines d'application divers.
- Crédibilité des comportements des agents.
- Capacité du modèle à passer à l'échelle, c'est-à-dire gérer un grand nombre d'agents en parallèle.

Nous commencerons par présenter les architectures considérées comme réactives permettant de modéliser simplement des comportements de plus en plus complexes. Puis nous présenterons celles plus cognitives, qui intègrent des processus qui tentent de représenter les spécificités du cerveau humain. Après avoir exposé les avantages et les inconvénients de chacune de ces architectures, nous nous intéresserons aux architectures hybrides, qui tentent de combiner les deux approches précédentes, et nous verrons qu'il existe de nombreuses manières différentes de réaliser cette hybridation.

2.2.2 Architectures réactives

Un agent réactif agit en réponse à des stimuli internes ou externes. Le cas le plus simple serait un agent sans état interne, ni but, qui ne ferait que réagir lors de la perception de certains événements. De simples règles dites de "perception-action" suffisent à modéliser un agent réactif trivial. Cependant un agent réactif peut être doté d'un état interne, évoluant avec des règles lui étant propres et prenant en compte un ensemble de données internes et externes. Cet état interne peut ainsi être le déclencheur d'un comportement, non directement lié à une perception ou un stimulus externe. Un agent réactif plus complexe peut même être doté d'une représentation du monde et de connaissances. Mais classiquement on considère qu'un tel agent ne peut raisonner sur lui-même ou son environnement, ni planifier un comportement.

2.2.2.1 Architectures hiérarchiques

Une des architectures réactives les plus célèbres est peut-être la **subsomption** de [Brooks 1985]. Dans cette architecture, chaque comportement qu'un agent peut adopter (par exemple, "éviter les collisions", "explorer", "suivre une route", etc.) est pris en charge par un module spécifique. Chacun de ces modules a accès à l'ensemble des informations qui lui est nécessaire afin de juger la pertinence de sa propre activation (perceptions, état interne, etc.). Ces modules de comportement sont ordonnés hiérarchiquement (les comportements des couches supérieures pouvant d'ailleurs être très complexes), et peuvent inhiber en cas de conflit les modules

de comportement situés en-dessous (voir figure 2.1). Ainsi, afin de sélectionner son comportement, un agent parcourt de haut en bas la liste de ses modules, et dès que l'un d'entre eux juge son activation comme pertinente, il s'exécute. Par exemple le module "éviter un obstacle" inhibe le module "suivre une route", ainsi un agent en train de suivre une route est capable d'interrompre son déplacement pour éviter un obstacle.

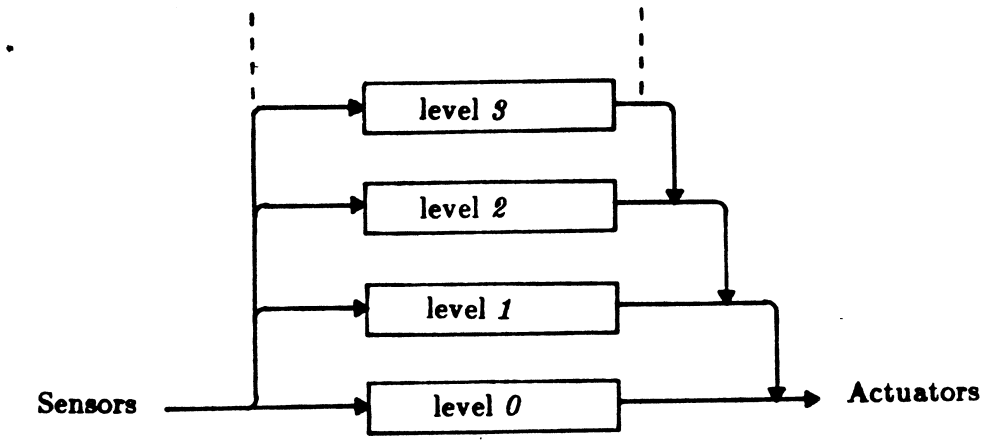

FIGURE 2.1 – Schéma de l'architecture de subsomption de Brooks

Cette architecture a été utilisée dans le domaine de la robotique, lorsque la complexité des tâches des robots entraînait des temps de calculs très importants et un manque de réactivité. Ce nouveau paradigme a alors permis de s'intéresser à des robots plus simples qui, plutôt que de construire une représentation du monde pour se déplacer et agir dans l'environnement, utilisaient le monde directement comme la meilleure représentation possible pour sélectionner leurs actions.

Cette architecture a été utilisée dans le domaine de la robotique, lorsque les robots n'avaient que des comportements simples à leur disposition. Une telle architecture, de part la simplicité du processus décisionnel est très peu consommatrice en ressources. De plus, elle est extrêmement modulaire, puisqu'il est facile d'insérer ou de supprimer de nouveaux comportements. Elle est également pratique pour complexifier au fur et à mesure les comportements activables par l'agent, en rajoutant des couches comportementales de haut-niveau par-dessus des couches plus simples.

Mais cette simplicité est également à la source de ses limitations : la rigidité de la hiérarchie interdit toute prise en compte du contexte dans l'arbitrage entre comportements, et les compromis entre comportements sont impossibles. L'architecture n'est absolument pas générique, puisqu'elle ne peut fonctionner que de manière ad-hoc dans un environnement donné et les comportements manquent de flexibilité.

2.2.2.2 Architectures de propagation de niveau d'activations

Si le côté trop hiérarchique d'une architecture est problématique, pourquoi ne pas tester des architectures décentralisées ? C'est ce que propose [Maes 1990], dans son architecture, chaque module de comportement décide lui-même de sa propre activation. Mais contrairement à l'architecture de subsomption, ces modules ne sont pas organisés hiérarchiquement, ils sont tous au même niveau. A chaque action correspond un *niveau d'activation*, qui varie dynamiquement, et qui indique la pertinence de l'action par rapport au contexte courant, ainsi qu'un *seuil d'activation*. Si le niveau d'activation d'une action dépasse son seuil, alors elle est exécutée. Les comportements sont reliés les uns aux autres à travers un réseau d'activation permettant de propager les niveaux d'activation en fonction des niveaux d'activations des comportements liés, du contexte et des comportements activés. Ce système de propagation/inhibition permet de faciliter les comportements permettant de réaliser plusieurs objectifs à la fois, d'encourager les comportements opportunistes, et d'éviter les conflits.

Cette architecture, entièrement décentralisée, est robuste et fortement réactive. Cependant la définition du réseau d'activation se complique avec l'augmentation de la complexité des comportement adoptables par l'agent. De plus aucune action à long terme ne peut être envisagée (ni planifiée), les actions immédiatement les plus prioritaires sont activées. Elle a été majoritairement utilisée pour modéliser des animaux, c'est-à-dire des agents virtuels assez simples, n'ayant pas un spectre comportemental trop important, dont le comportement est fortement prédictible et très peu planifié à l'avance.

Les travaux de [Rosenblatt 1989] combinent hiérarchie et propagation des niveaux d'activation. A la suite de ces recherches, Tyrrell a proposé la **hiérarchie à libre-flux** [Tyrrell 1993b]. La principale différence avec l'architecture précédente est que l'architecture est hiérarchisée, et que la propagation des niveaux d'activation se fera de haut en bas, à travers un graphe comportemental structuré, qui décompose chaque comportement activable en sous-comportements, puis sous-sous-comportements, et ainsi de suite jusqu'au niveau des actions élémentaires, c'est-à-dire les actions concrètement effectuées par les agents (voir figure 2.2). Lors de cette propagation tous les stimuli (internes et externes) sont pris en compte et leurs priorités sont propagées à travers le graphe, en étant modulées par les importances relatives de chaque comportement et sous-comportement. Ainsi, au niveau de décomposition maximal, toutes les actions sont dotées d'une priorité qui indique leur pertinence par rapport à la situation courante.

L'idée au cœur de ce travail est de réaliser la sélection d'actions le plus tard possible, c'est-à-dire au niveau des actions élémentaires concrètement effectuées par l'agent. Toutes les branches du graphe comportemental sont explorées, même les moins activées, ce qui permet d'obtenir des comportements de compromis d'une manière plus efficace que dans les précédentes architectures réactives.

L'idée de combiner hiérarchie et propagation des niveaux d'activation, tout en

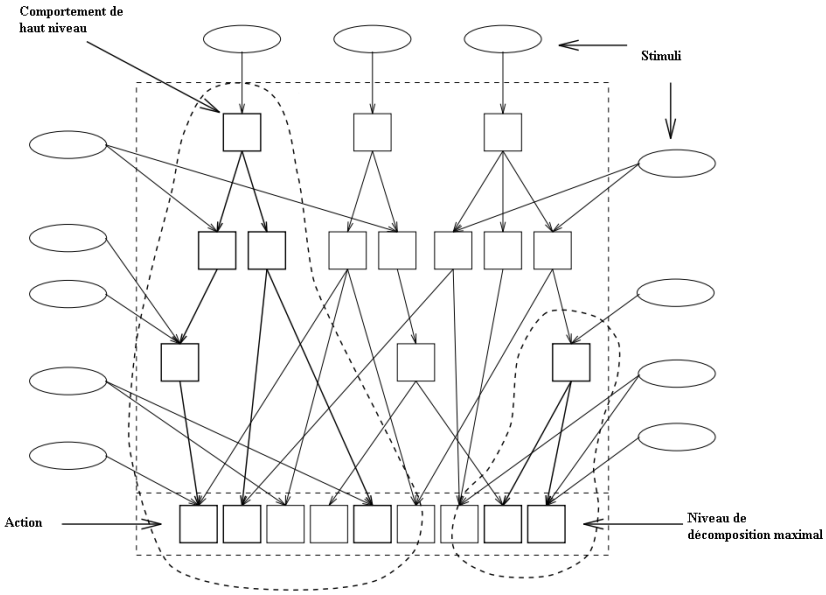

FIGURE 2.2 – Schéma de l'architecture de hiérarchie à libre-flux

retardant au maximum la prise de décision, afin d'explorer l'ensemble des comportements possibles, sans en mettre certains de côté est séduisante, et nous nous en inspirerons dans notre travail. Par contre, avec une architecture de ce type, aucune planification n'est possible, seule l'action possédant le plus grand niveau d'activation est activée, les comportements à long terme sont difficilement intégrables.

Cette architecture a surtout été utilisée pour modéliser des comportements d'animaux, ce qui explique que le manque de planification ne nuit pas à la crédibilité des comportements, mais pour la modélisation de comportements humains, une telle simplicité de décision pose problème.

2.2.2.3 Architectures basées sur un mécanisme de vote

DAMN [Rosenblatt 1995] donne un autre exemple d'architecture réactive classique non hiérarchique. Mais dans son cas, le mécanisme décisionnel repose sur un système de vote, dans lequel chaque module de comportement ne décide pas lui-même de sa propre activation, mais donne une note à chacune des différentes actions immédiatement activables par l'agent. Lors de cette étape de vote, les différents modules de comportement ont un poids dépendant de leur pertinence par rapport à la situation courante. L'action qui sera sélectionnée sera celle proposant le meilleur compromis entre tous les modules de comportement de l'agent (voir figure 2.3).

Cette architecture a été conçue pour la robotique. Dans son mécanisme décisionnel, chaque module de comportement a l'opportunité de donner son avis. Bien qu'elle soit limitée, une réelle collaboration entre modules permet de faire

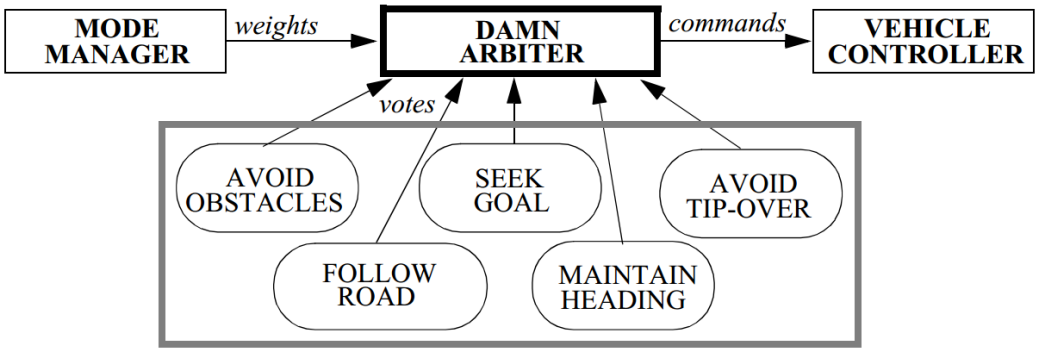

FIGURE 2.3 – Schéma de l'architecture DAMN

émerger le comportement. Cependant, le problème principal de cette approche est la prise en compte de la variabilité de l'importance des modules. En effet, déterminer pour chaque situation la pertinence relative de chaque module de comportement ne peut se faire que pour des cas simples. Le mécanisme de vote doit nécessairement être ad-hoc pour être convenablement équilibré. Cette approche est cependant pertinente en robotique afin d'obtenir un peu plus de souplesse par rapport aux architectures de subsomption. En relation avec notre problématique, l'idée d'intégrer un mécanisme de vote est intéressante, et permet de prendre en compte l'avis d'un ensemble de modules de comportement. Cependant les problèmes des approches précédentes (manque de vision à long terme, simplicité des comportements, etc.) restent valables.

2.2.2.4 Architectures avec tableau noir

Une approche complémentaire du mécanisme de vote de DAMN, est l'approche du **tableau noir** (*blackboard*). Le tableau noir permet de mettre en commun les connaissances et les avis de différents processus, dans le but de faire des recoupements et des déductions qui permettront de résoudre le problème courant du système. Un mécanisme de tableau noir comporte trois composants principaux :

- Des modules spécialistes, qui sont les sources de connaissance de l'agent. Spécialisés dans des domaines distincts, ils ont pour objectif de sélectionner, parmi l'ensemble de leurs informations, celles pertinentes pour répondre aux problèmes courants.
- Un tableau noir, qui est une sorte de mémoire de travail partagée, rassemblant l'ensemble des problèmes, et des informations publiées par les spécialistes. Chaque module spécialiste peut à son tour se servir d'une information partagée par un autre module pour l'utiliser dans son raisonnement.
- Un module de contrôle, qui est chargé de la modération entre tous les spécialistes et qui doit organiser les différentes informations de la manière la plus

cohérente possible.

Un des premières modèles, **Hearsay** ([Erman 1980]) est utilisé pour comprendre la parole. Il sert de socle à une architecture permettant de coordonner des processus indépendants afin de résoudre un problème de manière coopérative. En effet la compréhension du langage humain est un exemple typique de problème coopératif (voir exemple 2.4). Un nombre très important de processus hétérogènes et orthogonaux sont nécessaires au décryptage d'un texte, ou d'un dialogue.

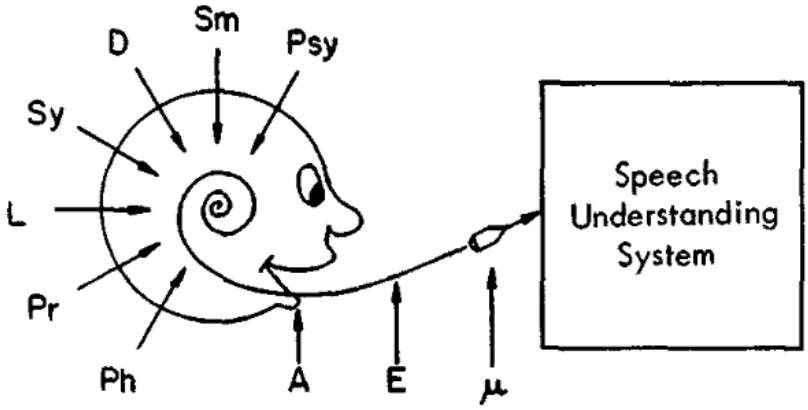

FIGURE 2.4 – Exemple de multiplicité des processus intervenant dans le dialogue

[Garcia 2000] dans le système **IBeNet** (*Internal Behavior Network*) propose une autre utilisation du tableau noir. Dans son mécanisme décisionnel, la "mémoire de travail" est itérativement mise à jour par différents spécialistes. Le processus commence par une spécification du problème à résoudre, et se termine par une solution. Chaque spécialiste met à jour la mémoire de travail en ajoutant une solution partielle, basée sur son domaine d'expertise. Ce mécanisme permet de prendre en compte des avis hétérogènes, tant sur le plan des mécanismes internes que des objectifs, et de les faire collaborer à l'élaboration d'une solution unifiant les points de vue.

Ce genre d'approche est très séduisante, et semble être une amélioration intéressante d'un simple mécanisme de vote, dans lequel les votes ne sont pas "justifiés". Mais toute la complexité du mécanisme repose sur le module de contrôle, qui doit être capable d'intégrer les connaissances de chaque spécialiste dans un même processus de résolution de problème. Si les spécialistes sont trop différents dans leur vision, leur système de référence, leur langage, etc. la mise en commun des informations devient un problème insolvable. La généricité du modèle est donc complexe à atteindre. De plus un tel processus de sélection de comportement demande du temps, et la question du passage à l'échelle est problématique.

2.2.2.5 Architectures motivationnelles et systèmes de classeurs

D'après [Nuttin 1985], un besoin, pour une entité, est une exigence inhérente au fonctionnement de cette entité. Les motivations peuvent être définies comme l'ensemble des raisons qui poussent un individu à agir. A la base d'une motivation se trouve toujours un besoin (voir les travaux de [Maslow 1943]). En réponse à une motivation, il faut qu'il existe des actions permettant de répondre à cette motivation. On distingue généralement 2 types de motivations, les motivations internes qui sont issues des besoins internes de l'individu (les besoins physiques/physiologiques/psychologiques) et les motivations externes, basées sur des besoins externes, c'est-à-dire issus de l'environnement.

Les architectures suivantes se réclament de l'approche **animat** [Meyer 1996, Guillot 2001], qui se concentre sur la synthèse du comportement de d'animaux virtuels ou de robots, grâce à des mécanismes inspirés de l'éthologie [Blumberg 1996].

Le modèle **MonaLysa** [Donnart 1996] utilise un **systèmes de classeurs hiérarchiques** (***HCS***) pour gérer l'architecture de contrôle des agents.

Un **système de classeurs** (*Learning Classifier System*) [Holland 1977] gère un ensemble de règles qui associent une ou plusieurs conditions à une action. Les classeurs peuvent avoir des poids indiquant leur importance, poids qui peuvent varier en cours de simulation, par exemple en fonction du nombre de fois qu'ils sont activés. Par la suite, des liens avec l'apprentissage par algorithmes génétiques ont été développés [Goldberg 1988, Wilson 1995].

Un des développements de ces systèmes de classeurs est le **système de classeur hiérarchique** (*Hierarchical Classifier System* [Dorigo 1993, Donnart 1998]. Ces systèmes possèdent deux groupes de classeurs, l'un s'occupant des actions internes, l'autre des actions externes, et utilisent des règles pouvant utiliser des symboles.

Un exemple de ce type d'architecture est l'architecture **MonaLysa** (voir figure 2.5), qui est composée de :

- Un **module réactif**, chargé de décider quelle action sera prochainement exécutée. Cette décision ne dépend pas uniquement des informations communiquées par les capteurs, mais également du but courant de l'agent. Ce module contient un ensemble de règles (ou classeurs) qui permettent à l'agent de réagir aux événements de l'environnement, ainsi qu'au contexte interne de l'agent.
- Un **module de planification** qui décompose les tâches en sous-tâches en fonction des informations venant de l'environnement via les capteurs de l'agent.
- Un **module de gestion du contexte** qui met à jour le contexte interne de l'agent. Ce module manipule donc une pile de tâches, avec la tâche courante

de l'agent au sommet. Le module de planification décompose la tâche courante en sous-tâches et les place sur la pile.

- Un **module de rétribution interne**, qui, grâce à un processus de renforcement, modifie les priorités des règles du module réactif et du module de planification.
- Un **module d'auto-analyse** qui analyse le comportement courant de l'agent afin de proposer de nouvelles tâches permettant d'améliorer les comportements futurs de l'agent.

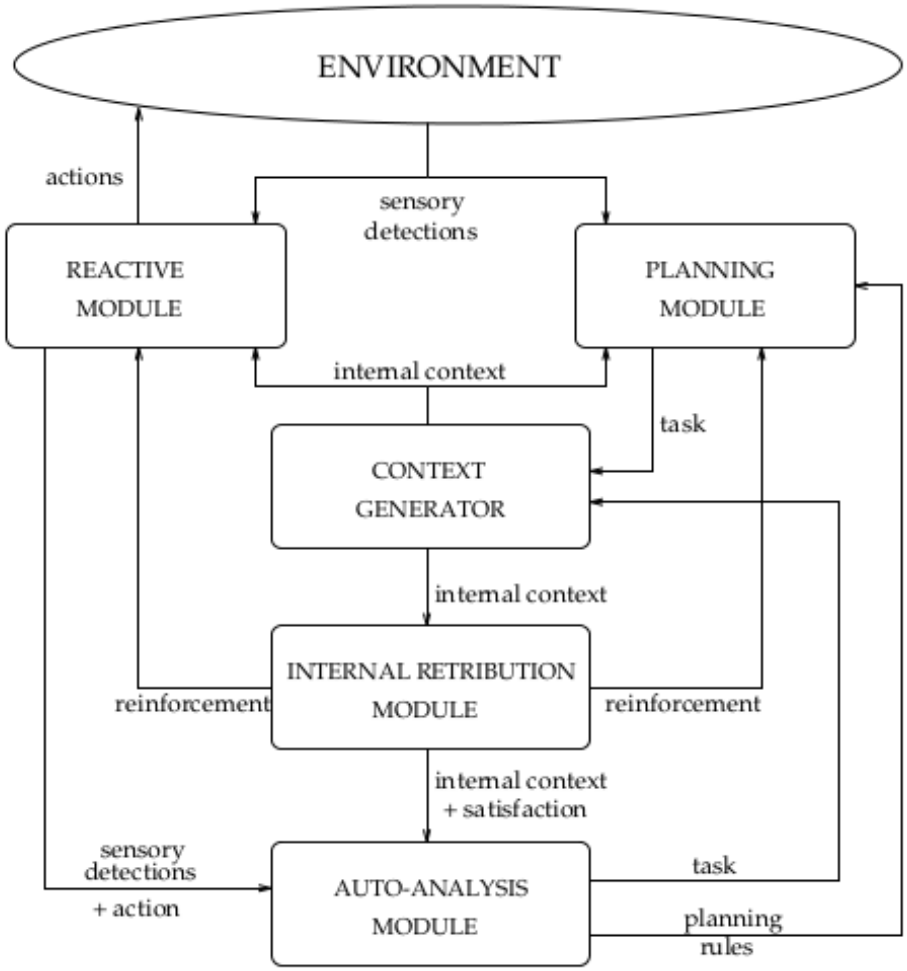

FIGURE 2.5 – Schéma de l'architecture MonaLysa

Les systèmes de classeurs ont également été utilisés pour implémenter une hiérarchie à libre-flux, comme dans le logiciel **DirectIA**, qui intègre en plus un mécanisme de planification réactive permettant de réaliser des comportements

humains complexes (notamment des missions militaires).

L'architecture **MHiCS** [Robert 2003] fonctionne également sur ce principe (voir figure 2.6). Elle possède 4 niveaux :

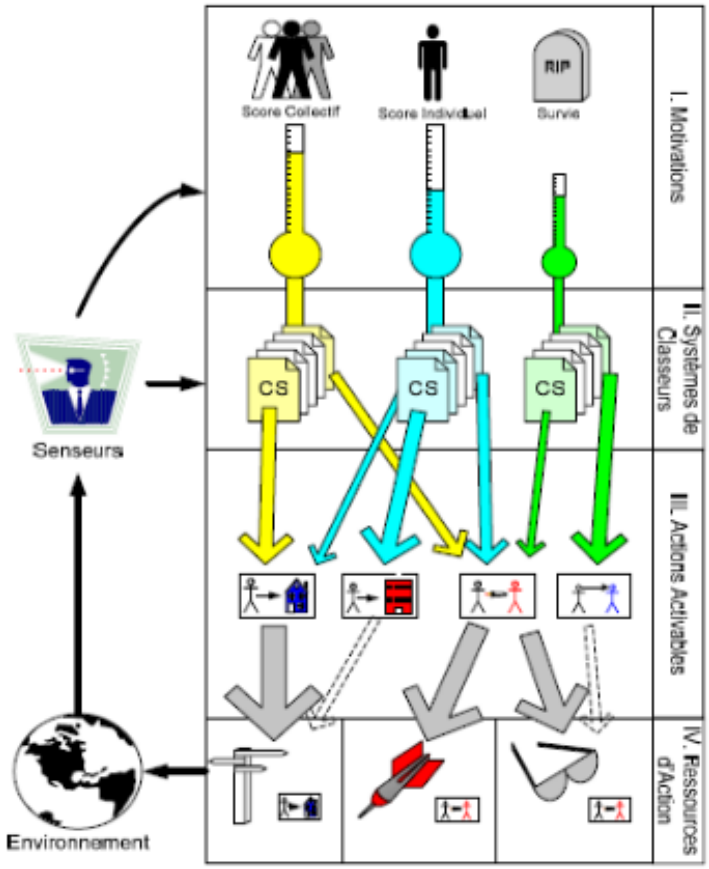

FIGURE 2.6 – Schéma de l'architecture MHiCS

- Le niveau des **motivations**, associées à deux valeurs : la puissance relative, et la valeur de motivation. La puissance relative sert à personnaliser l'agent. Elle est comprise entre 0 et 1 et indique l'importance de cette motivation pour l'agent. La valeur de la motivation varie avec le temps en fonction des comportements de l'agent.
- Le niveau des **systèmes de classeurs**. A chaque motivation correspond un système de classeurs. Le but de ces systèmes est de satisfaire la motivation correspondante.
- Le niveau des **actions activables**. Plusieurs classeurs peuvent être sélectionnés au même moment, donc plusieurs actions peuvent être exécutables en même temps. Pour déterminer laquelle sera exécutée, elles sont triées par priorité (calculée à partir de l'intensité d'exécution).

- Le niveau des **ressources d'actions** indique les ressources nécessaires à l'exécution des actions. L'action avec la plus forte intensité d'exécution a la priorité pour utiliser les ressources dont elle a besoin. Les autres actions exécutables ne pourront alors pas les utiliser. Le comportement de l'agent est donc une combinaison de toutes les ressources activées.

MHiCS a été développé dans le cadre du jeu vidéo, afin de modéliser des bots pour des jeux multi-joueurs, par exemple des scénario de capture de drapeau dans *Team Fortress CLassic*, un mod de *Half-Life*, un jeu de tir à la première personne.

L'architecture **MASLOW** (*Multi-Agents System based on Low-Needs* proposée par [Andriamasinoro 2003] repose sur la pyramide des besoins naturels proposée par [Maslow 1943]. L'auteur définit 3 catégories de besoins :

- Les *Low Needs* qui sont les besoins innés et naturels de chaque agent (par exemple la faim).
- Les *Medium Needs* qui sont les besoins permettant de combler les Low Needs (par exemple "aller au restaurant").
- Les *High Needs* qui permettent d'anticiper les Low Needs (par exemple, travailler pour gagner de l'argent).

L'architecture possède un "réseau d'actions", qui indique les relations entre toutes les actions possibles (parallélisme possible entre actions, inclusion dans un comportement, lien de succession, etc.). Le mécanisme de décision détermine l'action la plus prioritaire en fonction des besoins les moins satisfaits, et du réseau d'actions.

L'idée de placer une motivation à l'origine d'un comportement est intéressante, à la fois pour les animaux, les robots (une des applications de ce modèle est le problème des "robots fourrageurs") et les humains (une autre application est la simulation de l'activité quotidienne de paysans). Toute la difficulté de l'approche repose sur le réseau d'actions, qui peut devenir problématique à construire en cas de graphe comportemental complexe. Par ailleurs, le côté ad-hoc de ce réseau interdit au modèle d'être générique.

L'architecture développée par [de Sevin 2006], propose également un système basé sur les motivations (voir figure 2.7). Dans cette architecture, le module décisionnel effectue un choix à partir des perceptions de l'agent, de ses motivations, et d'un module de planification. Ce modèle prévoit que les émotions de l'agent puissent modifier ses variables motivationnelles.
Cette architecture a été développée afin de simuler le comportement d'un agent au cours de la journée et testée dans un environnement domestique.

En prenant en compte les émotions dans le processus décisionnel, cette architecture permet de simuler des comportements plus crédibles, mais cela augmente la

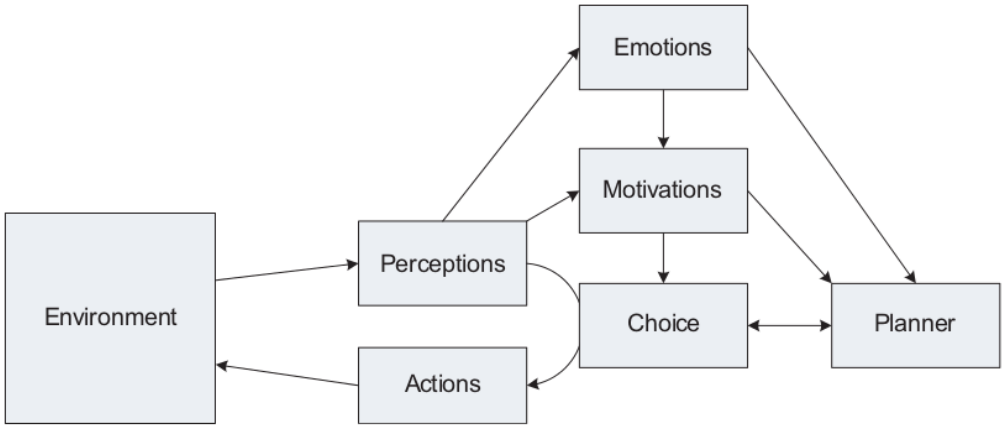

FIGURE 2.7 – Schéma d'une architecture motivationnelle basée sur des classeurs hiérarchiques

complexité du processus décisionnel, ce qui peut poser problème lorsqu'on cherche à passer à l'échelle. De plus, ces architectures ne sont pas conçues pour intégrer aisément de nouveaux composants.

2.2.2.6 Conclusion sur les architectures réactives

On a tendance à classer les architectures réactives au sein du paradigme ***bottom-up***. Les comportements sont construits du plus simple vers le plus complexe, et souvent l'intelligence des agents semble émerger de la collaboration de modules très simples à la base. Les avantages habituellement attribués aux agents réactifs sont leur simplicité de modélisation, leur rapidité de réaction, leur adaptation aux changements de l'environnement, et leur faible besoin en ressources de calcul. Ces capacités sont intéressantes par rapport à certaines de nos contraintes, à savoir la flexibilité des comportements et des agents, et le passage à l'échelle. Cependant certaines contraintes ne sont pas prises en compte, notamment la notion de crédibilité comportementale, qui dans le cas d'humains virtuels, pourrait bénéficier d'une planification des comportements sur le long terme, plutôt que de manière uniquement réactive. Le modélisation et l'intégration de processus cognitifs plus complexes permettraient également un gain de crédibilité intéressant. Ces notions sont traitées dans les architectures dites cognitives.

2.2.3 Architectures cognitives

Classiquement dans la littérature, l'approche réactive est opposée à l'approche dite "cognitive", ou symbolique. Au départ, l'approche cognitive part d'une étude du cerveau humain et cherche à l'imiter dans ses raisonnements et processus. Un

agent cognitif, ou *délibératif*, raisonne de manière logique à partir de représentations symboliques de lui-même et de son environnement. Dans ce type d'approche, plutôt que de faire émerger un comportement intelligent de la coopération de modules eux-mêmes peu intelligents, l'intelligence provient de théories permettant à l'agent de planifier un comportement sur le long terme afin d'atteindre ses objectifs, potentiellement complexes. Plutôt que de construire le comportement, à partir d'éléments simples et de le complexifier au fur et à mesure, l'approche cognitive essaye de décomposer des problèmes complexes en sous-problèmes, afin d'en réduire la complexité, et de les traiter séparément.

2.2.3.1 Architectures de planification délibérative

Dans ce type d'architecture, les actions ont des règles (règles procédurales, opérateurs, etc.) contenant au moins deux éléments : des préconditions qui décrivent les conditions requises pour accomplir l'action, et des effets qui décrivent l'état atteint après l'exécution de l'action. Les agents ont des buts à satisfaire (un état à atteindre, ou des actions à accomplir) et construisent des plans dynamiques (arbres comportementaux) pour les atteindre. Les actions exécutées sont choisies parmi celles leur permettant d'avancer dans leurs plans.

L'approche cognitive tire ses origines du paradigme symbolique développé par exemple par [Newell 1972] qui propose une analogie entre cerveau humain et système de traitement de l'information manipulant des symboles. Le modèle **GPS** (*General Problem Solver*)[Newell 1961] est basé sur une analyse des fins et des moyens (*means-ends analysis*). Il compare l'état courant avec l'état désiré, en déduit les étapes pour passer de l'un à l'autre, puis trouve les moyens d'atteindre ces fins en appliquant des règles de production (des règles du type SI <SITUATION> ALORS <ACTION>).

STRIPS (*STanford Research Institute Problem Solver*)[Fikes 1972] est également un précurseur. C'est un algorithme de planification classique, et d'un fonctionnement assez simple, également basé sur l'analyse des fins et des moyens. Les composants de cet algorithme sont :

- Les prédicats. Ils permettant de décrire l'environnement.
- Les états possibles du monde. Un état est constitué d'un ensemble de prédicats.
- Les buts. Un but est un état du monde, que l'agent souhaite atteindre.
- Les actions (ou opérateurs). Une action est définie par des préconditions (un état particulier du monde) et des effets (modifications de certains prédicats).
- Une pile de traitements. La pile contient l'ensemble des actions que l'agent doit effectuer, ainsi que les buts qu'il doit réaliser.

L'algorithme en lui-même est simple :

```
On initialise la pile en y plaçant le but initial de l'agent;
tant que La pile n'est pas vide faire
    Dépiler l'élément au sommet de la pile;
    si C'est un but alors
        si Il est vérifié alors
            On le supprime;
        sinon
            On choisit une action permettant d'atteindre ce but et on l'empile;
            On empile également les préconditions de cette action;
        fin
    sinon
        C'est une action;
        si Les préconditions sont vérifiées alors
            On exécute l'action et on la dépile
        sinon
            On revient en arrière jusqu'au dernier choix d'actions effectué, et
            on change d'action choisie;
            Si aucune autre action n'est possible, l'algorithme ne trouve pas
            de solution;
        fin
    fin
fin
```

Algorithme 1 : Algorithme STRIPS[Fikes 1972]

Cet algorithme souffre de nombreux problèmes, mais il est à la base de nombreux algorithmes de planification ultérieurs. Notons qu'il peut être amélioré en intégrant une méthode de recherche du type A^*.

Une des architectures cognitives les plus connues est l'architecture **SOAR** [Laird 1987, Laird 2012], qui a connu un développement continu depuis sa création. Cette architecture a pour objectif de modéliser l'ensemble des capacités cognitives d'un agent intelligent.

Le cycle standard de la décision est le suivant (voir figure 2.8 :

- Perception
- Élaboration : mise à jour de la mémoire à court terme, la mémoire de travail (*Working Memory*)
- Proposition d'opérateurs : les règles de production proposent des opérateurs qui correspondent au contexte
- Évaluation des opérateurs
- Sélection de l'opérateur courant
- Application de l'opérateur

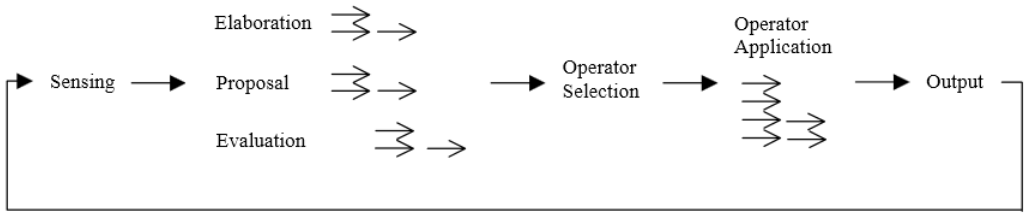

FIGURE 2.8 – Cycle de décision de SOAR

Un agent SOAR (figure 2.9) est doté d'une mémoire à long terme sous 3 formes distinctes :

- La mémoire procédurale, composée de règles de production, qui lui permet de savoir *comment* faire les choses. Il s'agit de routines connues de l'agent lui indiquant la manière de réaliser un objectif.
- La mémoire sémantique représente sa mémoire déclarative, c'est-à-dire ses connaissances objectives sur les *faits* et les *objets* de l'environnement.
- La mémoire épisodique retient les expériences passées de l'agent, c'est-à-dire d'anciennes mémoires de travail pouvant être réutilisées pour résoudre des buts similaires.

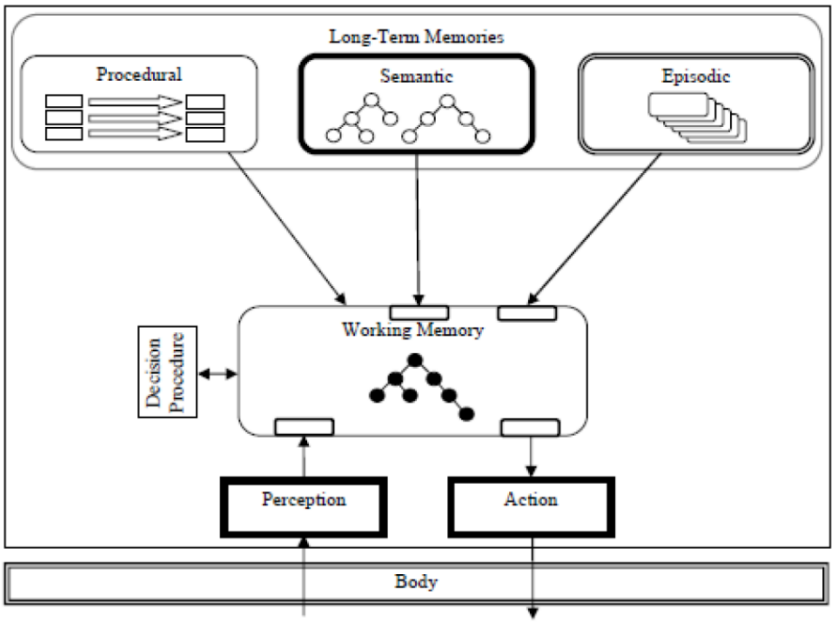

FIGURE 2.9 – Schéma de l'architecture SOAR

Une autre architecture influente est l'architecture **ACT-R** [Anderson 2004]. De

manière similaire à SOAR, ACT-R fonctionne grâce à des règles de production et à une mémoire séparée en deux parties : une mémoire procédurale et une mémoire déclarative. Ces deux mémoires communiquent à travers des buffers, et à chaque pas de temps le mécanisme de décision choisit la règle de production la plus adaptée.

Il est également intéressant de s'intéresser aux **HTN** (*Hierarchical Task Network*) [Erol 1996]. Dans un HTN, les actions et les états de l'environnement sont similaires à ceux utilisés dans STRIPS. Les actions sont appelées tâches primitives (*Primitive Tasks*) dans un HTN. Contrairement à une planification de type STRIPS, qui a pour objectif de de trouver une séquence d'actions permettant d'obtenir un certain état du monde, une planification HTN a pour objectif de trouver des plans qui accomplissent des réseaux de tâches (*task networks*). Un réseau de tâches est un ensemble de tâches qui doivent être traitées ensemble, de manière coordonnée, et qui n'a pas obligatoirement de conditions concernant son état final. Dans la plupart des cas, un réseau de tâches ne contient pas uniquement des tâches primitives, mais également des tâches non-primitives, c'est-à-dire des tâches ne pouvant pas être exécutées directement, pour lesquelles l'agent doit trouver un moyen de les réaliser. Afin de réaliser une tâche non-primitive les agents utilisent des méthodes, de la forme (α, d), avec α une tâche non-primitive et d un réseau de tâches. Pour accomplir la tâche α, l'agent doit accomplir toutes les tâches du réseau d sans violer aucune de ses contraintes. Par rapport aux buts de STRIPS, les réseaux de tâches sont beaucoup plus riches et la planification s'appuie sur une décomposition des buts complexes en sous-buts plus simples.

Ces architectures sont à l'heure actuelle largement utilisées. Les architectures de type SOAR ciblent principalement la modélisation et la simulation d'agents intelligents. Cependant elles restent focalisées sur la résolution de problèmes et l'optimisation des comportements, pas sur la crédibilité d'un comportement humain. Par ailleurs, en cherchant à imiter le fonctionnement du cerveau humain, ces architectures perdent en simplicité, et la complexité des mécanismes mis en œuvre sont difficile à faire passer à l'échelle. Les architectures de type HTN sont très utilisées dans le jeu vidéo[Kelly 2008], car elles permettent de réduire les coûts de calcul des algorithmes décisionnels.

2.2.3.2 Architectures de planification réactive

Dans les architectures de "planification réactive", les actions sont des modèles (*pattern*) de comportement, contenant au moins 3 éléments : un but, un contexte devant être vrai pour que le pattern puisse être activé, et un corps contenant toutes les actions exécutées par le pattern. Contrairement aux architectures de planification délibérative, dans lesquelles les plans sont construits automatiquement par le système depuis les actions, les plans sont statiques et écrits par les humains comme pattern de comportements. Un pattern de comportement consomme un but de l'agent (le but du pattern) et produit des sous-buts (quand le corps du pattern

est exécuté).

Procedural Reasoning System (PRS) [Georgeff 1987], permet à un agent de choisir une intention ou un plan avant qu'ils ne soient complètement élaborés. Cela permet d'éviter des attentes trop fortes concernant l'environnement, ainsi qu'un travail de planification trop précis dans un futur incertain. Le point central ici est de pouvoir interrompre et abandonner un plan en cours, en fonction des circonstances. Le système est composé de (voir 2.10) :

- **Une base de connaissances** contenant des croyances et des faits concernant le monde.
- Un ensemble de **buts** ou **désirs** que l'agent cherche à réaliser.
- Un ensemble de **procédures** décrivant comment atteindre un but grâce à l'exécution d'une suite d'actions ou comment réagir par rapport à une perception. Une procédure est constituée d'un corps, qui décrit les étapes de la procédure, et d'une condition qui indique dans quelles situations la procédure est utile. Le corps est un plan, consistant non pas en une suite d'actions à réaliser, mais en une suite de sous-buts à accomplir. A un instant donné, les différents procédures en cours d'exécution peuvent être considérées comme représentant les intentions de l'agent. Ces procédures sont également appelées aires de connaissances (*Knowledge Areas*).
- Un **interpréteur** (ou mécanisme d'inférence) permettant de manipuler ces différents composants.

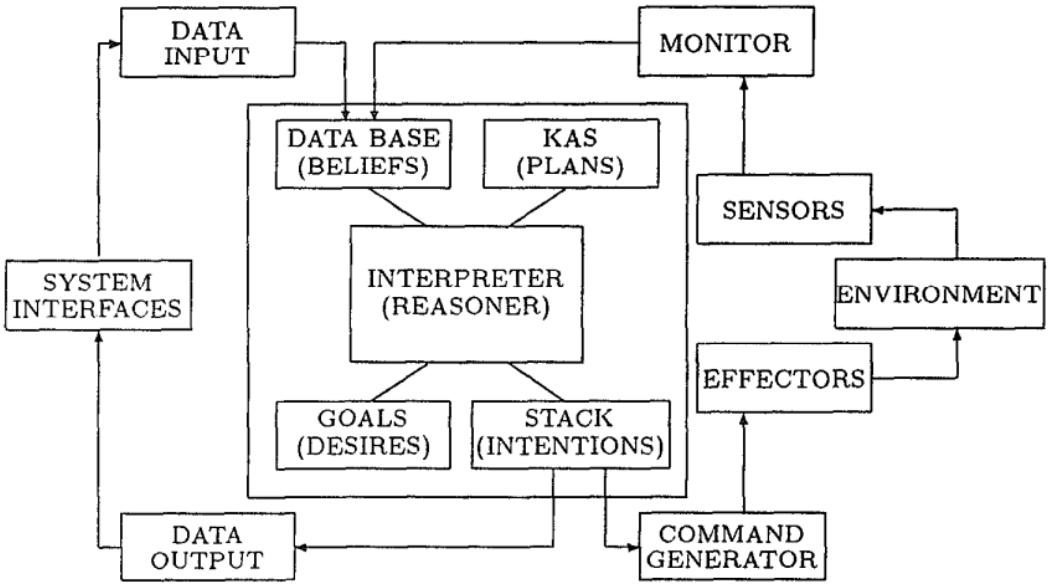

FIGURE 2.10 – Schéma de l'architecture PRS

PRS est utilisé pour le contrôle de robots mobiles, étant situés dans un

environnement réel, et devant réagir en temps réel, pour lesquels la réactivité est un élément clé. De nombreuses architectures ultérieures reprennent les bases posées par cette architecture. Dans le principe de la planification réactive nous apprécions tout particulièrement l'idée de planification sans raisonnement, c'est-à-dire de coupler des comportements planifiés sur le long terme (gain de crédibilité) mais sans raisonnement, puisque la planification a été réalisée hors ligne (gain de temps de calcul, et donc facilitation du passage à l'échelle). Par contre le problème de la généricité du modèle n'est pas traité.

2.2.3.3 Le modèle Belief-Desire-Intention

Le modèle **BDI** (*Belief-Desire-Intention*) de [Georgeff 1999], inspiré de PRS, est un autre exemple d'agent délibératif. Un agent BDI raisonne à partir de 3 types de connaissances : ses croyances, ses désirs, et ses intentions. A partir de ses croyances, l'agent va mettre à jour ses désirs, et en fonction du contexte va tenter de réaliser ses intentions en les traduisant sous forme d'actions (voir figure 2.11).

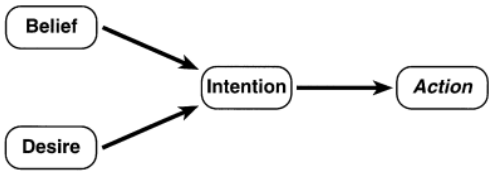

FIGURE 2.11 – Schéma de l'architecture BDI

- Les **croyances** sont les informations qu'un agent possède sur le monde, mais aussi sur ses propres états internes. Les croyances sont subjectives, et induisent une vision personnelle de l'environnement de l'agent.
- Les **désirs** représentent les souhaits de l'agent. Ils n'ont pas à être réalisables immédiatement (certains peuvent même ne jamais se réaliser), et certains désirs peuvent être conflictuels (par exemple *vouloir de l'argent* et *vouloir se reposer*).
- Les **intentions**, qui sont créées lorsqu'un agent décide de satisfaire un désir. Il planifie alors son comportement pour tendre vers cet objectif, ce qui peut comprendre la réalisation de sous-objectifs intermédiaires.

La principale caractéristique de cette architecture, est de séparer la phase de planification de la phase d'exécution. Les agents sont ainsi libres de séparer leur temps de calcul entre la réflexion (sélection du meilleur plan) et l'action (sélection de la meilleure manière d'effectuer le plan).

La boucle décisionnelle est la suivante :

- Génération des options comportementales

- Sélection de l'option après délibération selon les options proposées
- Mise à jour des intentions en fonction de l'option choisie
- Exécution de l'intention
- Attente d'un nouvel événement

Depuis sa création, l'architecture BDI a servi de socle à de nombreux modèles, par exemple **Jadex** de [Braubach 2005] (voir 2.12). Les messages entrants (stimuli externes) et les événements internes (stimuli internes), ainsi que les nouveaux buts sont les entrées du mécanisme interne de réaction et de délibération. Basés sur les résultats du processus de délibération, ces événements sont transformés en plans sélectionnés depuis la bibliothèque de plan. Au cours de l'exécution de ces plans, la base de croyances peut être modifiée, des messages peuvent être envoyés, des sous-buts peuvent être créés et peuvent causer des événements internes. Un agent Jadex est déterminé par ses croyances, ses buts, et sa librairie de plans à l'état initial. Le mécanisme de réaction et de délibération est le seul module commun à tous les agents Jadex, les autres composants sont des *capacités* pouvant être modifiées et réutilisées.

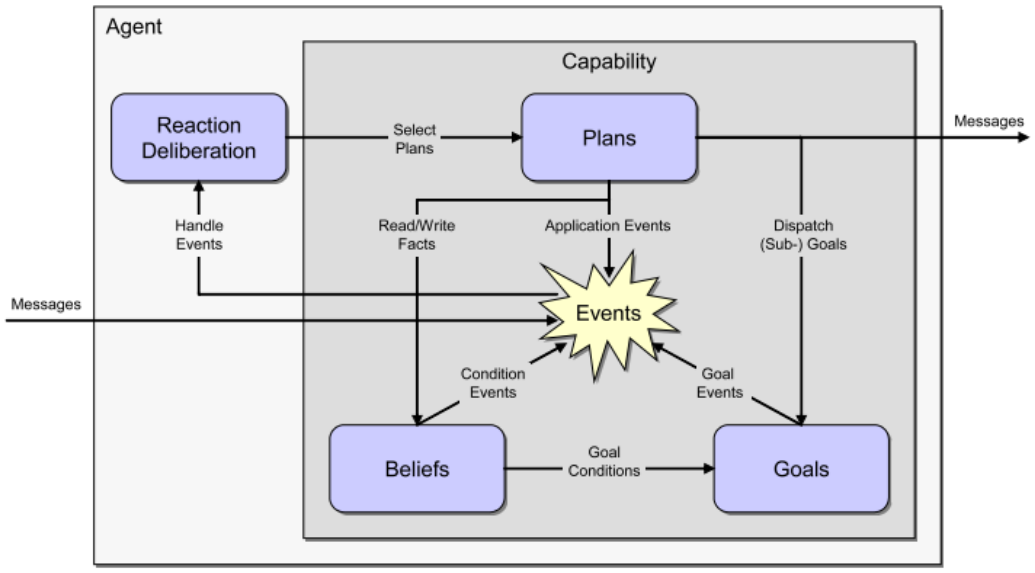

FIGURE 2.12 – Schéma de l'architecture Jadex

Un agent **JAM** ([Huber 1999]) est quant à lui composé de 4 modules :

- Un modèle du monde, qui est une base de connaissance qui représente les croyances de l'agent.
- Une bibliothèque de plans, qui contient un ensemble de plans que l'agent peut utiliser pour accomplir ses objectifs.

- Un interpréteur, qui est le "cerveau" de l'agent et qui raisonne sur ce que l'agent doit faire, quand et comment il doit le faire.
- Une structure d'intentions, qui est une structure interne rassemblant les buts courants de l'agent, ainsi que les progrès et succès de chacun.

Une des architectures BDI les plus complètes est peut-être l'architecture **PEP → BDI** de [Jones 2009, Edward 2010], qui ajoute des composantes d'émotions (peur/espoir, énervement/gratitude, honte/fierté et défiance/confiance), de personnalité (par exemple, empathie, altruisme, curiosité, bravoure, nervosité, etc.), et de physiologie (notamment stress, faim/soif, fatigue, température, blessures) au modèle standard BDI. Le mécanisme décisionnel est le suivant :

1. Perception des nouvelles informations venant de l'environnement.
2. Génération des émotions par rapport aux nouvelles perceptions.
3. Modification des croyances à cause des nouvelles perceptions et des émotions.
4. Mise à jour des variables physiologiques.
5. Sélection des désirs et des intentions.
6. Mise à jour des émotions (impact des intentions).
7. Si les nouvelles émotions diffèrent de celles de la phase 2, nouvelle mise à jour des croyances, variables physiologiques, désirs et intentions.
8. Planification.
9. Exécution du plan.

Le modèle BDI, et ses nombreuses extensions offrent un paradigme qui s'est largement imposé dans le paysage des architectures d'agents. La qualité du modèle ainsi que l'élégance du design ont conquis de nombreux modélisateurs. Cependant nous ne trouvons pas dans un tel modèle, la modularité recherchée. En effet, bien que le paradigme BDI soit générique, il n'est pas possible pour un tel agent de prendre en compte autre chose que ses désirs et ses croyances (et ses émotions pour un agent PEP → BDI) — comme par exemple ses motivations, ses obligations, ses prédictions, etc.

2.2.3.4 Conclusion sur les architectures cognitives

Les avantages habituellement attribués aux agents cognitifs sont leur capacité à planifier des comportements sur le long terme, résoudre des problèmes complexes, et exhiber des comportements plus proches de ceux d'un humain cherchant à optimiser son comportement. Malheureusement ils sont plus coûteux en terme de ressources, et ont plus de mal à réagir rapidement à des changements dans l'environnement. Cependant, la capacité à planifier à l'avance un comportement en vue de réaliser

un objectif complexe, pouvant nécessiter la réalisation de plusieurs sous-objectifs intermédiaires, est une capacité indispensable que nous devons fournir à nos agents. Il est donc nécessaire de trouver un compromis entre les deux approches, réactive et cognitive.

2.2.4 Architectures hybrides

Une architecture hybride est une architecture qui combine les approches réactive et cognitive présentées précédemment. En effet, de nombreux chercheurs soutiennent l'idée que ni un comportement entièrement réactif, ni un comportement entièrement délibératif ne s'avèrent suffisants pour modéliser un humain virtuel. Agir uniquement de manière réactive à l'environnement, interdit toute réflexion menant à un plan permettant de réaliser des buts complexes, ou à long terme. A l'inverse, agir uniquement en suivant un plan préparé à l'avance, empêche de prendre en compte des changements inattendus dans l'environnement, ou de réaliser des actions de manière opportuniste. Il parait donc intéressant de coupler les deux approches. Cependant cette combinaison peut se réaliser de nombreuses manières. Nous allons ici en présenter certaines qui nous semblent particulièrement pertinentes.

2.2.4.1 Winner-take-all

Une première manière d'organiser ce couplage est de séparer les comportements réactifs et cognitifs dans des parties isolées de l'architecture et de leur donner le contrôle de l'agent de manière alternée, en fonction du contexte. Ce genre d'architecture est appelé ***Winner-take-all*** (le gagnant emporte tout)[Öztürk 2009], c'est-à-dire que le but ou le module le plus important prend le contrôle total de l'agent.

L'architecture **InteRRaP** [Müller 1993], inspirée de **Touring Machine**[Ferguson 1992], est composée de 3 couches comportementales hiérarchiques (voir figure 2.13). La première couche est celle des comportements réactifs, elle contient une bibliothèque de comportements simples. La deuxième couche peut être assimilée à celle des comportements cognitifs. Elle permet à l'agent de planifier un comportement en servant des briques comportementales du niveau inférieur, afin de réaliser un objectif plus complexe. La troisième couche est dédiée à la coopération, elle autorise l'agent à demander de l'aide à d'autres agents afin de réaliser des buts qu'il ne peut remplir seul.

De plus, une séparation claire est faite entre contrôle (partie gauche de la figure 2.13) et connaissance (partie droite). La partie *contrôle* comporte la représentation du monde, ainsi que les comportements réactifs, cognitifs, et coopératifs. La partie *connaissance* contient le modèle du monde, les modes de comportements, les plans locaux, et les connaissances coopératives.

Le processus décisionnel des agents InteRRaP est décomposé en 2 phases. Tout d'abord une phase ascendante de traitement de l'information, puis une phase des-

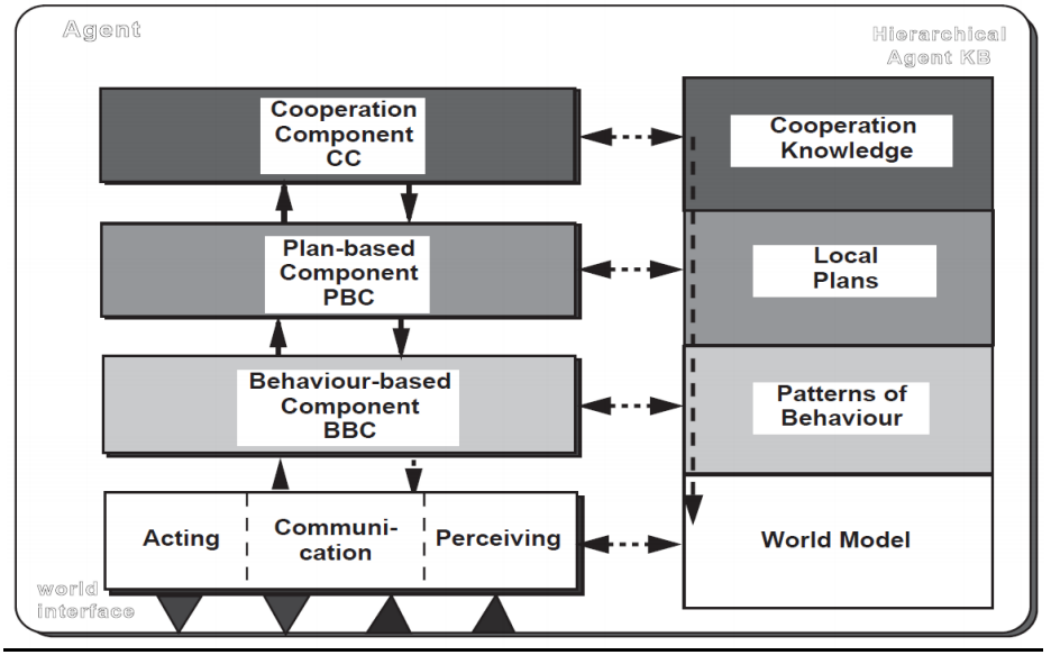

FIGURE 2.13 – Schéma de l'architecture InteRRaP

cendante de traitement de la décision. La phase ascendante opère de manière itérative : chaque couche, en partant de la plus basse, essaye de traiter l'information. Si elle réussit, elle envoie ses directives vers les couches inférieures jusqu'aux actions élémentaires situées dans le modèle du monde. Si elle n'arrive pas à traiter l'information, elle les fait passer au niveau supérieur, qui essaye à son tour. Ainsi, l'agent va tout d'abord tenter de trouver dans sa bibliothèque de comportements de la couche réactive, un comportement lui permettant d'accomplir son but. S'il en trouve un, alors ce comportement est immédiatement exécuté. S'il n'en trouve aucun, alors il passe à la couche cognitive, et cherche un plan lui permettant de remplir ses objectifs. Si un plan est trouvé, il est immédiatement exécuté ; sinon, l'agent passe au stade de coopération et demande de l'aide à l'extérieur.

Cette architecture a été conçue pour la robotique et les systèmes multi-agents. Les processus cognitifs viennent suppléer les processus réactifs, uniquement dans le cas où ces derniers ne parviennent pas à gérer la situation courante. On ne peut pas réellement parler de coopération entre réactif et cognitif, mais plutôt d'adaptation de la profondeur de réflexion à la complexité du problème. Par ailleurs, une fois qu'une couche parvient à traiter l'information, elle inhibe toutes les autres (à la fois les couches supérieures qui ne seront pas atteintes, et les couches inférieures qui ne feront qu'exécuter les ordres). Il n'y a donc qu'une seule couche à la fois qui s'exprime. Cependant le principe de pouvoir adapter la profondeur de réflexion en fonction du contexte est une idée intéressante.

L'architecture **PECS** [Schmidt 2005] n'est pas hiérarchique. Elle est constituée de 4 composants situés au même niveau (voir figure 2.14). Le premier module s'occupe du physique de l'agent et gère des variables homéostatiques internes. Selon le niveau de ces variables, le module peut demander le déclenchement d'un comportement associé (manger, dans le cas d'une variable de faim importante par exemple). Le deuxième module est un module en charge des émotions. Selon son état émotionnel, l'agent pourra envisager l'activation de tel ou tel comportement. Le troisième module représente les mécanismes cognitifs, tels que la gestion des connaissances. Un quatrième module "social" gère la coopération entre agents. Ces 4 modules sont en conflit permanent pour prendre le contrôle de l'agent. En effet, lors de chaque itération du processus décisionnel, un module décisionnel est chargé de déterminer quel module est le plus pertinent par rapport à la situation courante. Ce module prend alors le contrôle de l'agent et inhibe tous les autres. Ce type d'architecture fonctionne également selon le principe du Winner-take-all, car le module jugé le plus apte à gérer la situation courante prend le contrôle total de l'agent sans prendre en compte les avis des modules "perdants".

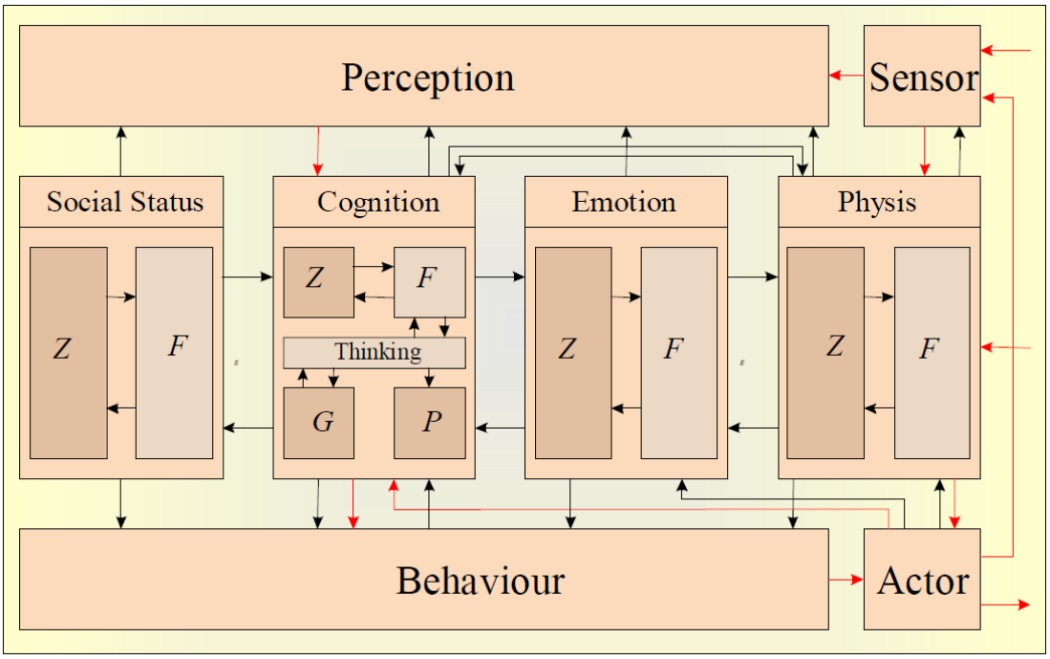

FIGURE 2.14 – Schéma de l'architecture PECS

Cette architecture a été utilisée dans 3 domaines d'application différents, une simulation de distribution de nourriture dans une zone de guerre permettant de tester des stratégies de coordination en environnement incertain, tout en devant prendre en compte des aspects émotionnels et psychologiques en compte ; une si-

mulation de l'impact de préventions médicales sur une population ; un programme décrivant l'importance de l'émotion dans le processus décisionnel.

Dans le cas de systèmes *Winner-take-all* il est difficile de parler de coopération entre modules, puisqu'un seul module peut s'exprimer à la fois. De plus, la structure de l'architecture est prédéfinie (nombre de modules, types des modules, etc.), elle est donc très peu flexible. Cependant, l'organisation des modules est intéressante, et nous nous sommes inspirés de ce modèle pour construire le squelette de notre architecture.

2.2.4.2 Architectures maître-esclave

Une autre manière de gérer les relations entre cognition et réaction est de rendre l'un maître de l'autre. Ainsi, certaines architectures placent un module cognitif à la tête du modèle, et lui permettent de dicter sa conduite à un module réactif.

Le modèle **3T** de [Bonasso 1997] est un parfait exemple de ce genre d'architecture utilisée en robotique. Cette architecture est découpée en 3 couches :

- Une couche de comportements réactifs, reprogrammables dynamiquement.
- Un séquenceur qui active ou désactive des ensembles de comportements pour créer un réseau comportemental dynamique en fonction du contexte et des actions de l'agent.
- Un planificateur délibératif qui raisonne en profondeur sur les buts, les ressources et les contraintes temporelles de l'agent.

Chaque couche donne des ordres à la couche inférieure, qui décompose ces ordres en ordres compréhensibles pour la couche inférieure. Ainsi le planificateur synthétise l'ensemble des buts de l'agent sous forme d'un plan partiellement ordonné qui est transmis au séquenceur. Ce dernier décompose ce plan en comportement réactifs simples, correctement ordonnancés, qui sont envoyés à la couche réactive qui réalise leur exécution.

Being-in-the-world de [Depristo 2001], est un modèle conçu pour modéliser un agent dans un monde virtuel multi-agents, de type "donjon". Il est composé de 2 modules asyncrones et indépendants contrôlant le comportement des agents : *Descartes* et *Heidegger*.

- Descartes est un module qui raisonne à partir des états internes et des connaissances mises à jour par Heidegger, et qui décompose les buts de haut niveau de l'agent en buts plus simples (grâce à un mécanisme de chaînage arrière).
- Heidegger est un module qui met à jour les connaissances à partir des perceptions et essaye de satisfaire les buts immédiats de l'agent (envoyés par Descartes). Il gère également les actions réactives, c'est-à-dire celles en réponse immédiate à un stimulus.

Ce modèle se rapproche de l'architecture **ICARUS** par [Langley 2006] qui présente une forme d'hybridation dont l'objectif est de proposer un système de traitement de l'information qui soit psychologiquement plausible. ICARUS (voir figure 2.15) est composé de 4 modules : *Argus* perçoit son environnement de manière sélective. *Daedalus*, en planifiant le comportement, est en charge de la partie cognitive de l'agent. *Maender*, lui, est en charge des comportements réactifs et exécute les plans transmis par *Daedalus*. Enfin *Labyrinth* s'occupe de la gestion des connaissances.

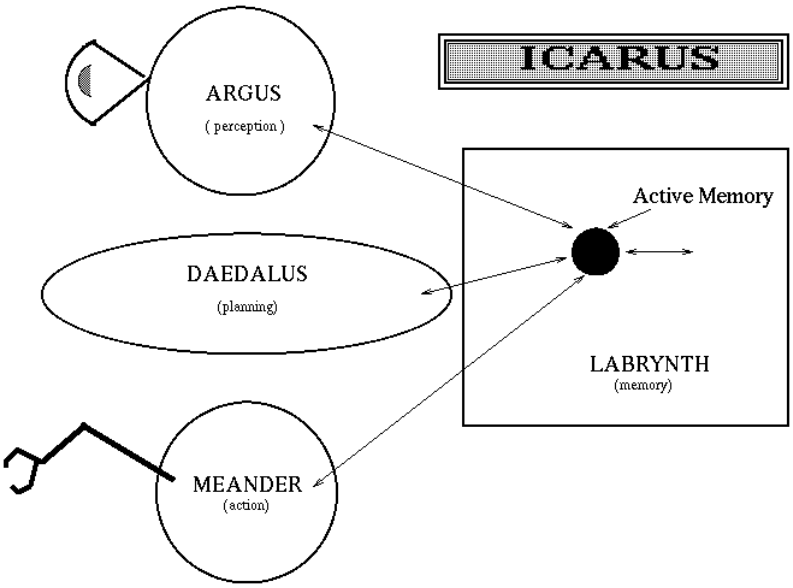

FIGURE 2.15 – Schéma de l'architecture ICARUS

Dans toutes ces architectures, bien que des comportements réactifs et cognitifs soient présents dans l'architecture, on ne peut pas réellement parler de coopération, puisque le réactif ne fait que suivre les ordres avec un minimum d'autonomie, ou alors "court-circuite" le cognitif, par exemple pour répondre à des situations d'urgence. Par ailleurs, l'organisation hiérarchique des modules ne laisse que peu de place à une généralisation du modèle.

2.2.4.3 Approche centrée sur les interactions

Une autre sorte assez originale d'hybridation est l'approche centrée sur les interactions. **CoCoA** (*Cognitive Collaborative Agents*)[Devigne 2007] est une plateforme de simulation pour agents virtuels. Dans les travaux de [Dujardin 2010b] une

architecture centrée sur les interactions est proposée ; elle est basée sur l'approche **IODA** [Kubera 2011]. Le concept clé de cette approche, est que les agents sont définis par les interactions qu'ils peuvent effectuer ou subir. Les interactions sont définies par une condition, une garde de distance, et des actions (les conséquences). Les agents sont dotés d'une base de connaissances à partir de laquelle un module de planification détermine les actions à effectuer pour atteindre leurs buts. Les plans sont de 2 types différents :

- Les plans abstraits, qui sont constitués d'une suite d'actions visant à la réalisation d'un but.
- Les plans concrets, qui sont des plans abstraits, enrichis des déplacements nécessaires à la réalisation des actions.

Le modèle est fondé sur la dualité raisonnement-individualité. Le raisonnement étant commun à tous les agents, il détermine les possibilités d'un agent selon ce qu'il peut faire et ce qu'il veut faire. L'individualité, elle, est personnelle à chaque agent, et permet à l'agent de décider comment il exécute ce qu'il veut faire.

Le processus décisionnel est décomposé en 5 modules travaillant en série :

1. Perception.
2. Mise à jour des croyances.
3. Planification : chaînage arrière à partir des buts, chaque alternative proposée étant notée dans sa globalité.
4. Détermination de la prochaine interaction
5. Exécution.

Un nouveau processus de décision est lancé à la fin de chaque exécution d'interaction, ou lors d'une nouvelle perception. Des fonctions d'affinement des évaluations de chaque alternative, permettent au processus décisionnel de prendre en compte un certain nombre de critères :

- Opportunisme : création d'un rayon autour de chaque interaction possible dans lequel la motivation associée augmente.
- Accomplissement en espace : bonus accordé aux interactions proches de l'agent.
- Accomplissement en temps : bonus accordé aux buts dont l'accomplissement est rapide.
- Revalorisation multi-buts : bonus accordé aux actions permettant la réalisation de plusieurs objectifs.
- Inertie : bonus accordé à l'action courante.

Cette architecture est utilisée principalement pour modéliser des agents dans des systèmes multi-agents (potentiellement de grande envergure), dans des environnements assez simples. Elle est très pertinente quand on cherche à simuler un grand nombre d'agents simples. En effet, la matrice d'interaction indiquant l'ensemble

des interactions possibles entre tous les types d'agents, permet d'avoir une excellente vue de l'ensemble du système. CoCoA est donc parfaite dans le cadre d'un jeu vidéo par exemple. Cependant la matrice d'interactions peut devenir très complexe si de nombreux types différents d'agents sont en relation, s'ils possèdent un large spectre comportemental, et si l'environnement offre de nombreuses manières de réaliser les différentes interactions. Par ailleurs, cela suppose une connaissance parfaite de l'environnement, puisque lors de la création d'un nouveau type d'agent, il faut spécifier l'ensemble des interactions avec le reste du système. Il devient donc impossible d'interagir avec des agents inconnus.

2.2.4.4 Intégration de multiples fonctions cognitives

Récemment, une idée provenant des sciences cognitives a eu un impact le domaine des architectures décisionnelles. Cette idée est que la cognition humaine, lorsqu'elle est confrontée à un problème, met en jeu de nombreux processus cognitifs bien distincts. Plutôt que de chercher à modéliser le plus précisément possible un processus en particulier, il est intéressant de chercher à en *intégrer* le plus possible dans un même modèle.

Polyscheme de [Cassimatis 2005], s'intéresse à la simulation de la cognition humaine. Cette architecture utilise de nombreux modules "spécialistes" fonctionnant sur des modèles différents (règles de production, réseaux bayésiens, etc.). Un mécanisme de focalisation sélectionne un thème d'intérêt et chaque spécialiste donne son opinion sur la question. Ensuite, un module décisionnel intègre tous ces avis et sélectionne un comportement.

Cette architecture est particulièrement pertinente par rapport à nos problématiques. L'idée de faire collaborer différents spécialistes, fonctionnant sur des modèles hétérogènes, sur un même problème fait partie de nos buts. Polyscheme permet effectivement une étroite collaboration entre des modèles distincts. Cependant l'expression des avis des spécialistes est également hétérogène, et dépend du modèle utilisé par le spécialiste. Ce qui fait que la fonction d'intégration des avis est dépendante des spécialistes utilisés. Ainsi, l'intégration dans le processus décisionnel de nouveaux spécialistes impose une modification de la fonction de décision. Il est ainsi nécessaire de trouver un moyen de rendre le modèle plus générique afin de pouvoir intégrer d'autres spécialistes sans toucher au reste de l'architecture.

L'architecture CoSy [Hawes 2007] cherche à créer un "robot cognitif", ce qui nécessite l'emploi de capacités cognitives variées. Cette architecture (voir figure 2.16) est composée d'un ensemble de sous-architectures faiblement couplées. Chacune contenant un certain nombre de composants de traitement d'informations, qui partagent des informations via une mémoire de travail (*working memory*) et un manager de tâches (*task manager*). Il existe deux types de composants : les non-managés et les managés. Les composants non-managés réalisent des traitements de données

assez simples, et tournent en continu, envoyant leurs résultats dans la mémoire de travail. Par opposition, les composants managés contrôlent les changements opérés dans la mémoire de travail et suggèrent des tâches à réaliser. Comme ces tâches sont coûteuses et ne peuvent pas être toutes réalisées en même temps, elles sont sélectionnées en fonction des besoins du système. Par ailleurs, chaque sous-architecture possède une mémoire de travail qui lui est propre et qui est accessible en lecture à toutes les autres sous-architectures, mais seuls les processus de la sous-architecture peuvent y écrire.

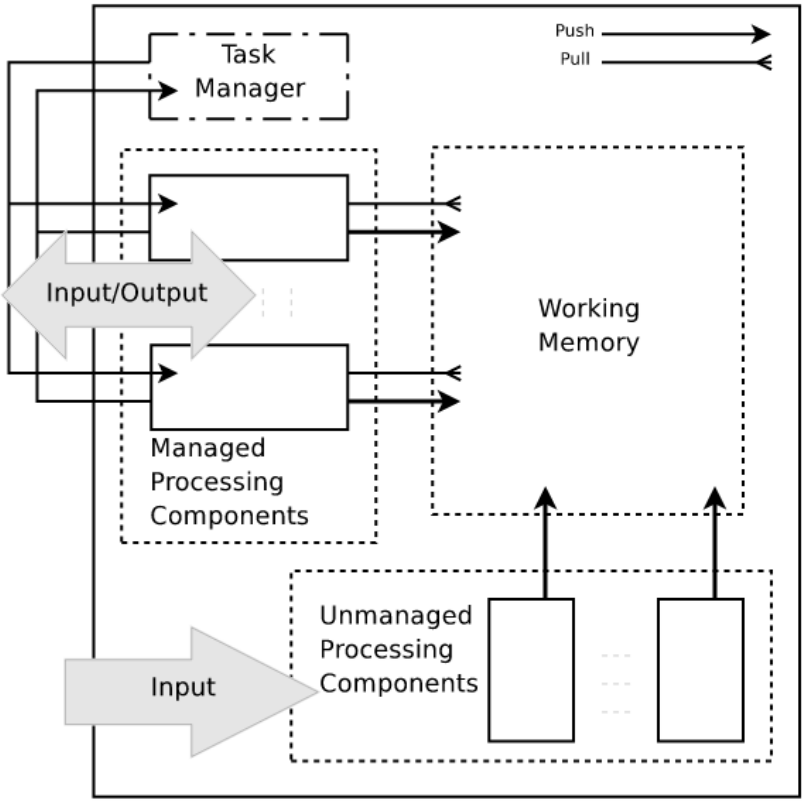

FIGURE 2.16 – Schéma de l'architecture CoSy

Cette architecture permet presque de répondre à l'ensemble de nos contraintes. Elle est capable d'intégrer des sous-architectures hétérogènes, elle est générique et flexible, permet de modéliser un large spectre comportemental ainsi que des agents différents, etc. Le seul problème par rapport à nos objectifs est le coût computationnel d'un tel processus. En effet, le processus décisionnel est long. Non seulement les différentes sous-architectures partagent des informations via la mémoire de travail, mais en plus certains processus prennent en entrée des informations depuis cette mémoire de travail, et certains autres vont manipuler des données depuis d'autres sous-architectures. Cette complexité organisationnelle rend le mécanisme

de sélection de tâche très performant. Mais elle le ralentit considérablement (et d'autant plus avec l'augmentation du nombre et de la complexité de chaque sous-architecture). Ce coût n'est pas pénalisant dans le cadre d'utilisation de CoSy (la gestion d'un robot). Mais dans notre cadre de simulation fortement multi-agents, nécessitant une forte réactivité de chaque agent, cela est problématique. Nous allons donc être contraints de simplifier le mécanisme.

2.2.4.5 Conclusion sur les architectures hybrides

Les architectures hybrides offrent une importante variété d'organisations entre modules, ce qui est pour nous une source d'inspiration considérable. Comme nous le verrons dans le chapitre 3, l'architecture que nous proposons est résolument hybride, et elle reprend différentes théories, idées, et modèles présentés ici.

2.3 Composition de comportement

Les contraintes de flexibilité et généricité que nous imposons à l'architecture, mènent à se poser la question de la composition de comportements, à savoir coupler au sein d'un même processus décisionnel des comportements fortement hétérogènes, pouvant venir de modules différents, basés sur des modèles distincts. Nous soulignons que ce thème de recherche est différent de celui de la *Behavior Composition* [Yadav 2011], qui a pour objectif de synthétiser un comportement virtuel cible, grâce à la coordination d'un ensemble de comportements connus en entrée. L'objectif de ce champ de recherche est d'atteindre un objectif en l'absence de plan connu. Cela est très différent de notre problématique : coupler en entrée du processus décisionnel des comportements hétérogènes afin de produire en sortie un comportement hybride, bénéficiant de l'apport de l'ensemble des comportements d'entrée.

2.3.1 Les critères de Tyrrell

Un précurseur sur le thème de la composition de comportements est [Tyrrell 1993a]. Il a indiqué une liste des caractéristiques que doit posséder un mécanisme de sélection d'action de bonne qualité. Voici ces points :

1. **Gérer tous les types de comportements.** Les agents doivent être capables de traiter tous les types de comportements activables dans la simulation.
2. **Persistance des actions.** Les agents doivent avoir tendance à ne pas interrompre leur action en cours, en raison du "coût" associé à un changement d'action.
3. **Priorité des comportements proportionnelle aux états (internes et externes) courants.** Lors de la sélection d'actions, l'agent doit prendre en compte l'état courant des stimuli. Ce critère s'applique uniquement dans le cas de systèmes homéostatiques, ce qui n'est pas le cas dans notre travail.

4. **Préférence des actions consommatoires [1] par rapport aux actions appétitives [2].** A priori, toutes choses étant égales par ailleurs, un agent devrait préférer une action consommatoire à une action appétitive, en raison de la simplicité de l'action permettant d'atteindre de manière plus sûre son objectif.
5. **Concurrence équilibrée entre comportements.** Les comportements permettant la réalisation de plusieurs objectifs et ceux utiles à un seul, ainsi que les comportements déclenchés par de nombreux stimuli et ceux sensibles à un seul doivent être en concurrence de manière équilibrée. On ne doit pas introduire de déséquilibre dans le système de préférences, tel qu'un type de comportement soit toujours préféré à un autre, quel que soit le contexte.
6. **Persistance du comportement courant.** D'une manière similaire au point 2, lors de la sélection d'actions, l'agent doit avoir tendance à poursuivre le comportement, ou la séquence d'actions, en cours de réalisation, plutôt que d'entamer une nouvelle séquence d'actions, en raison du coût associé à l'inefficacité de la séquence d'actions abandonnée.
7. **Interruption possible si nécessaire.** Le comportement ou l'action en cours doivent pouvoir être interrompus en cas de besoin.
8. **Opportunisme.** L'agent doit être capable de choisir des comportements opportunistes.
9. **Pas de système en *Winner-take-all*.** Lorsqu'un agent décide de satisfaire un but, il doit continuer à prendre en compte ses autres buts, afin de ne pas perdre d'informations et de garder la possibilité de choisir un comportement de compromis entre plusieurs objectifs.
10. **Combinaison des préférences.** La priorité d'une action doit pouvoir prendre en compte une combinaison des préférences des comportements associés (dans le cas ou cette action permet la réalisation de plusieurs tâches).
11. **Compromis.** Un agent doit être capable de choisir des actions qui ne sont pas le meilleur choix pour la réalisation d'un objectif, mais qui le sont lorsque tous les objectifs sont pris en compte simultanément.
12. **Combinaison flexible des stimuli.** Un agent doit être capable d'intégrer une combinaison de stimuli en parallèle.

Tous ces points sont intéressants, et seront pris en compte dans notre mécanisme décisionnel. Si les points 2, 3, 4, 6, 7, 8 et 12 sont des critères généraux caractérisant un mécanisme de sélection d'actions, les points 1, 5, 9, 10, et 11 sont directement liés au problème de la composition de comportements. En effet, le point 1 indique que le mécanisme doit gérer tous les comportements activables, c'est-à-dire quels que soient les comportements hétérogènes proposés en entrée, la décision doit être

1. Se dit d'un comportement qui mène directement à la satisfaction d'un besoin interne (par exemple la faim ou la soif)

2. Se dit d'un comportement qui tend à augmenter la probabilité que l'agent soit capable se satisfaire un besoin (par exemple le déplacement vers une source de nourriture)

capable de les traiter. Le point 5, la concurrence équilibrée entre comportements poursuivant un seul objectif, et ceux aidant à la réalisation de plusieurs, prennent tout leur sens lorsque un même comportement peut être utile à plusieurs modules différents, pour des raisons différentes, et en vue d'accomplir des objectifs différents. Idem pour le point 9, le mécanisme ne doit pas fonctionner en *winner-take-all*. Un comportement entamé ne doit pas prendre le dessus sur tous les autres, tout comme un module proposant des comportements ne doit pas négliger les avis des autres. Le point 10 rejoint le 5, l'importance d'une action à un moment donné dépend de l'avis de l'ensemble des comportements importants, ce qui permet à l'agent de choisir des actions de compromis (point 11).

2.4 Processus d'anticipation

Afin d'augmenter la crédibilité des comportements des agents, nous souhaitons disposer dans l'architecture un processus d'anticipation. Parmi l'éventail des processus généralement considérés comme cognitifs et dans lesquels la maîtrise humaine est indéniable, l'anticipation tient une place majeure. [Rosen 2012] définit un système d'anticipation de la manière suivante : "un système contenant un modèle prédictif de lui-même et/ou de son environnement, qui lui permet de changer d'état à un instant, en accord avec les prédictions du modèle pour un instant futur".

Une définition plus précise est également donnée : "un système d'anticipation S_2 est un système contenant un modèle d'un système S_1 avec lequel il interagit. Ce modèle est un modèle prédictif ; son état présent donne des informations sur des états futurs de S_1. De plus, l'état présent du modèle peut causer un changement d'état dans d'autres sous-systèmes de S_2 ; ces sous-systèmes sont (a) impliqués dans l'interaction de S_2 sur S_1, et (b) ils n'affectent pas le modèle de S_1 (c'est-à-dire qu'ils n'y sont pas liés). En général, on peut considérer les changements d'états de S_2 découlant du modèle comme une adaptation, ou une pré-adaptation, de S_2 par rapport à ses interactions avec S_1".

D'une manière plus pragmatique, l'anticipation est généralement considérée comme la capacité à prédire et raisonner sur des événements futurs.

Ces définitions sont en accord avec la notion d'anticipation intelligente (*intelligent anticipation*) développée par [Riegler 2001]. Bien que ces définitions ne soient pas unanimement reconnues [Riegler 2003], nous les considérerons dans notre travail comme standards, et partirons du principe que l'anticipation est un processus cognitif intentionné, basé sur des modèles prédictifs habituellement préexistants.

En intelligence artificielle, avant d'être utilisés dans la modélisation d'agents intelligents, des mécanismes d'anticipation ont tout d'abord été introduits dans les domaines de l'apprentissage par renforcement [Sutton 1991, Kaelbling 1996], des systèmes de classeurs (*Learning Classifier System*) [Stolzmann 1997], et des réseaux de neurones [Carpenter 1991].

[Butz 2003a] définit 4 types principaux d'anticipations : l'anticipation implicite, l'anticipation de récompense, l'anticipation sensorielle, et l'anticipation d'états.

2.4.1 Anticipation implicite

L'**anticipation implicite** réfère à des comportements pré-programmés, c'est-à-dire qu'aucune prédiction n'est réalisée. Dans un système d'anticipation implicite, les stimuli internes et externes sont directement pris en compte dans le processus décisionnel. Les informations d'anticipation sont directement incluses dans les comportements de l'agent, dans les structures et les interactions entre les capteurs, les actuateurs, et les algorithmes qui les relient. C'est-à-dire que c'est dans le module décisionnel lui-même, que le modélisateur a introduit des notions permettant à l'agent de réaliser une certaine forme d'anticipation. A titre d'exemple, il est possible de comparer ce type d'information à celles contenue dans le code génétique d'un animal (considéré comme un agent purement réactif).

Ce type d'anticipation nous intéresse peu, puisqu'il ne repose pas sur la prise en compte de prédictions dans le processus décisionnel.

2.4.2 Anticipation de récompense

Si un agent prend en compte dans son processus décisionnel des prédictions concernant les récompenses possibles des différentes actions qu'il exécute, alors il réalise une **anticipation de récompense** (*payoff anticipation*). Un système d'anticipation de récompense utilise dans son mécanisme décisionnel des anticipations des bénéfices de chaque action possible, mais ne prend pas en compte d'autres types de prédictions. Les systèmes de classeurs par anticipation (*anticipatory classifier system*) [Holmes 2002], ainsi que les systèmes d'apprentissage par renforcement [Sutton 1998, Kaelbling 1996] sont des exemples d'utilisation typique de ces prédictions.

Ce type d'anticipation est pertinent par rapport à notre approche, bien que son intérêt soit limité. En effet, nous désirons que nos agents puissent prendre en compte d'autres prédictions que des prédictions de récompense. D'ailleurs ces récompenses peuvent être délicates à manipuler, étant donné que les récompenses espérées des actions ne sont pas nécessairement identiques pour toutes les théories ou modèles utilisés pour modéliser le comportement de l'agent.

2.4.3 Anticipation sensorielle

Dans une **anticipation sensorielle** (*sensory anticipation*), les prédictions ne sont plus restreintes aux récompenses des actions, mais sont étendues aux états futurs de l'environnement, ainsi qu'aux futurs stimuli. Cependant, ces

prédictions n'impactent pas directement le comportement des agents, mais modifie la manière dont les senseurs traitent les perceptions [Fleischer 2003]. Les stimuli attendus peuvent ainsi être traités plus rapidement que ceux qui sont imprévus. L'anticipation sensorielle est fortement reliée aux mécanismes d'attention sélective et permet de simuler des états mentaux tels que la curiosité, ou la distraction (par exemple en filtrant un certain nombre de perceptions) [Castelfranchi 2006]. Dans la même direction, certains travaux se focalisent sur les phénomènes de surprise et prévision pour gérer l'anticipation [Piunti 2007].

L'anticipation sensorielle est un champ de recherche intéressant, mais qui semble complexe à mettre en place dans le cas d'une simulation fortement multi-agents.

2.4.4 Anticipation d'états

Dans un processus d'**anticipation d'états**, une prédiction des états futurs de l'environnement est utilisée directement dans le processus de décision. Comme pour l'anticipation sensorielle, le système doit disposer d'un modèle prédictif (inné ou appris), mais dans ce cas les prédictions modifient directement les choix de l'agent (sans passer par l'intermédiaire des senseurs). Souvent, un tel mécanisme est utilisé pour éviter des états futurs de l'environnement considérés comme indésirables [Davidsson 2003, Doniec 2008].

Dans [Capdepuy 2007], l'auteur utilise un processus d'anticipation d'état afin de construire un modèle interne de prédictions. Deux types de relations sont prises en compte : les événements dont les délais d'apparition temporelle sont proches, et ceux dont les apparitions sont proches temporellement. Ce modèle oblige à une simplification de taille : les agents ne perçoivent pas des flux continus d'informations à valeurs réelles, mais des événements en temps discret (de 0 à n événements peuvent être perçus à un pas de temps donné). En étudiant les flots d'événements et leur date d'apparition, un mécanisme repère les événements étant liés entre eux, et construit une méthode permettant de prédire, avec un certain degré de certitude, l'apparition d'un événement lorsqu'un événement qui lui est lié est observé. Un mécanisme d'oubli est également ajouté, afin de supprimer les prédictions erronées, ou obsolètes.

Ce modèle est intéressant, mais les simplifications effectuées ne peuvent être réalisées dans la nôtre, ce qui influence beaucoup le fonctionnement du processus d'anticipation.

Un processus d'anticipation d'états est également utilisé dans les **Quakebot** proposés par [Laird 2001]. Ces bots, conçus pour jouer à *Quake II*, un jeu vidéo de type FPS (*First Person Shooter*), sont basés sur l'architecture SOAR. L'objectif est de doter ces bots de la capacité d'adopter des comportements proches de ceux des joueurs humains, typiquement tendre des embuscades, ce qui implique d'être

capable de prévoir le comportement d'un ennemi, et agir en fonction de nos prévisions. Pour cela, le plus souvent, les développeurs ajoutent de manière spécifique les comportements voulus en fonction du contexte. L'idée dans Quakebot est d'ajouter aux bots une capacité générale d'anticipation des actions des adversaires.

Le Quakebot construit tout d'abord une carte de son environnement qui peut être réutilisée ultérieurement. Leurs capacités sensorielles ont été conçues pour ressembler à celle d'un humain (vision bloquée par les obstacles, bruits perçus dans un certain rayon). Le cycle de décision du Quakebot est identique à celui d'un agent SOAR (voir figure 2.8). L'approche de l'anticipation développée dans ce modèle consiste à "se mettre à la place de l'ennemi". C'est-à-dire que le Quakebot se crée une représentation interne de l'état interne de ses opposants. Évidemment, cette représentation n'est qu'une estimation, basée sur les observations de l'agent. Ensuite, afin de prédire le comportement de ses opposants, il va utiliser son propre processus décisionnel pour déterminer ce qu'il ferait lui, s'il était dans l'état dans lequel il suppose que son adversaire se trouve. En quelque sorte, l'agent utilise son propre processus décisionnel comme simulateur du processus décisionnel de l'autre. L'intérêt d'un tel modèle est qu'il est fortement générique. Il ne dépend pas du jeu dans lequel le bot évolue, ni des tactiques employées. Cependant pour que cela fonctionne, il est nécessaire que les objectifs et les comportements (ici les tactiques) de tous les agents soient proches.

L'auteur s'intéresse à 3 questions fondamentales concernant l'anticipation. Quand anticiper ? Comment anticiper ? Et quoi faire avec les anticipations ?

1. Il est contre-productif d'essayer d'anticiper continuellement, de le faire lorsque nos connaissances sur l'ennemi ne sont pas suffisantes, ou lorsque l'agent sait déjà ce qu'il doit faire. C'est pourquoi un Quakebot n'essaye d'anticiper que lorsqu'il perçoit un ennemi (ses connaissances sur lui ne sont donc pas nulles), que cet ennemi ne le voit pas et qu'il est loin (dans le cas contraire, le bot doit attaquer immédiatement).
2. Dans le cas de Quake II, la projection du bot sur un adversaire est globalement assez simple. En effet, lorsqu'un bot perçoit un ennemi il connaît immédiatement sa position, sa santé, son niveau d'armure, et son arme. Ce qui est suffisant pour induire une estimation plausible de son comportement.
3. Dans le cas du Quakebot, les auteurs se sont concentrés sur un seul cas dans lequel l'anticipation est utile : lorsque le bot prédit qu'un adversaire va se rendre dans une salle, et que lui-même peut s'y rendre en premier. Le bot pourra ainsi tendre une embuscade ou empêcher l'adversaire de s'emparer d'un bonus (en le prenant avant). Par ailleurs, afin de comparer le temps qu'il lui faut pour se rendre dans une salle, et le temps que mettra l'adversaire, le bot considère le nombre de salles à traverser pour l'un et l'autre. Ainsi, les conditions de terminaison d'un processus d'anticipation sont simples : l'agent continue à anticiper tant qu'il est capable de prédire la suite du comportement de son adversaire avec une certaine confiance (s'il y a trop de possibilités pour la suite d'un comportement, le bot abandonne l'anticipa-

tion), mais s'arrête dès qu'il prédit que l'agent se rendra dans une salle dans laquelle lui-même peut se rendre en premier.

Une extension intéressante de ce mécanisme est sa capacité d'apprentissage. En effet, afin de prédire le comportement d'un agent, le Quakebot peut avoir besoin d'un peu de temps (quelques secondes au maximum), or le temps est précieux dans un FPS. Dans SOAR il est possible de retenir la situation courante, et d'apprendre une nouvelle règle liée à ces conditions, qui permet d'obtenir directement l'anticipation associée. Il devient dont possible d'entraîner des bots au préalable, afin de leur faire apprendre des règles spécifiques liées à l'environnement ou au contexte général de la partie pour les rendre plus efficaces et plus rapides.

Ce type d'anticipation est très intéressant, cependant l'environnement n'a rien de comparable avec celui dans lequel nos agents sont appelés à évoluer. En effet, l'environnement d'un Quakebot est de petite taille, statique (les armes, bonus, et autres points d'intérêt sont fixes), le nombre des adversaires est très faible, et un seul bot doit être géré en temps réel. De plus un certain nombre de mécanismes de ce processus d'anticipation sont dépendants du jeu, notamment ceux permettant de définir quand un bot doit essayer d'anticiper le comportement d'un adversaire, ou pour déterminer dans quels cas une prédiction est utile, et ce que l'agent doit faire pour en profiter (dans ces cas là, des connaissances ad-hoc doivent être apportées). Par ailleurs, le contexte de ce bot est un jeu concurrentiel, ce qui explique que l'anticipation des comportements des adversaires est cruciale. Dans le cas de la simulation urbaine, les agents ne sont habituellement pas en concurrence, ni en coopération. La prédiction de leurs comportements est moins critique que dans un jeu de combat. Les capacités de prédictions les plus importantes sont donc plutôt au niveau de l'environnement et du propre comportement de l'agent.

2.5 Passage à l'échelle

La capacité de notre modèle à gérer un grand nombre d'agents en parallèle est primordiale pour deux raisons. D'une part parce que nous souhaitons que notre architecture puisse être utilisée pour des applications variées et d'autre part parce que son domaine applicatif immédiat, la simulation urbaine, demande la simulation d'un très grand nombre d'agents afin d'être crédible. De nombreux domaines de recherche traitent de ce côté fortement multi-agents.

2.5.1 Simulation de foules

Le domaine de la simulation de foules (*Crowd Simulation*) est très actif et s'intéresse, en particulier, à la simulation urbaine. L'objectif d'un simulateur de foule est de représenter de la manière la plus crédible possible, un ensemble de groupes d'humains virtuels.

Dans [Thalmann 2009], les auteurs identifient les problèmes à résoudre afin de simuler des foules en temps réel dans un environnement virtuel. Leurs contraintes sont la simulation de foules composées au minimum d'une centaine d'individus, dont la représentation graphique est de bonne qualité. Nous passerons sur les aspects modélisation graphique, génération automatique, animation, apparence, locomotion, recherche et suivi de chemin, évitement de collision, etc... qui sont loin de nos centres d'intérêt, afin de nous concentrer sur les aspects comportementaux.

A ce sujet, les auteurs expliquent que le comportement humain est modifié à l'intérieur d'une foule. C'est pourquoi ils essayent de grouper les agents ayant les mêmes buts, afin de représenter l'aspect grégaire de l'humain. Dans leur approche, les ressources computationnelles sont en majorité dépensées ailleurs (rendu graphique). Cela oblige les auteurs à devoir simuler des comportements intelligents tout en restant peu coûteux en temps de calcul. Pour ce faire, ils utilisent l'environnement en lui attachant de l'information, et s'en servent pour affiner les comportements des agents. Cependant les comportements exhibés restent très simples, il n'y a pas de réelles interactions entre agents (intra et extra groupe), les comportements sociaux sont inexistants, et les émotions collectives ne sont pas présentes.

Autonomous pedestrians proposé par [Shao 2007] s'intéresse à l'animation d'un piéton virtuel en environnement urbain. Leur première piste de travail concerne l'apparence humaine, le déplacement et le contrôle de la motricité. La deuxième concerne les problèmes de perception, basée sur une décomposition de l'environnement en cellules typées, contenant les informations utiles aux agents. La troisième piste correspond à la modélisation comportementale. Elle consiste en un mécanisme décisionnel *bottom-up* basée sur 6 règles de type perception-actions, pouvant être activées de manière séquentielle. Ces règles comportementales sont complétées par des comportements navigationnels, permettant aux agents d'éviter les collisions, les obstacles, de choisir leur chemin (*path finding*), etc. La question de la navigation est centrale dans cette approche, car leur objectif est de maximiser le réalisme des déplacements des agents. Ainsi, l'objectif des règles comportementales est d'améliorer la crédibilité des mouvements des agents. Les composantes motivationnelles intégrées au modèle permettent d'augmenter la variabilité des comportements plutôt que de gérer avec précision un état interne réaliste.

A un niveau plus élevé de décision, les *autonomous pedestrian* sont dotés d'un niveau de contrôle cognitif leur permettant de devenir des agents délibératifs, prenant en compte leurs connaissances, raisonnant sur l'environnement, et concevant et exécutant des plans à long terme. Cependant, même à ce niveau décisionnel, seule la crédibilité navigationnelle est prise en compte. Un plan à long terme se réfère donc à un *déplacement* à long terme et la planification d'un agent fait référence à sa planification de chemin (*path planning*) utilisant les données topologiques de la carte. Par ailleurs, ces agents sont dotés d'une mémoire leur permettant de retenir leurs différentes activités en cours, et les résultats d'une planification antérieure.

Alors que pour [Shao 2007] seule la crédibilité navigationnelle importe, dans notre cas toutes ces questions sont prises en charge dans d'autres parties de l'architecture globale de nos agents (le module de navigation, le module de physique, et le module graphique notamment (voir chapitre 3).

Dans **YaQ** ([Maim 2009]), il est possible de peupler de grands environnements avec des foules d'humains virtuels "en bandeau" (*Crowd Patches*), ce qui est à la fois moins gourmand en ressources que de simuler des foules interactives, et nécessite moins de mémoire que de scripter l'ensemble des déplacements des agents. Toutes les tâches coûteuses (évitement de collision et recherche de chemin principalement) sont pré-calculées dans des petites zones de l'espace, et ces zones sont interconnectées les unes avec les autres pour couvrir l'ensemble de l'environnement. D'un autre côté, dans YaQ il est également possible de simuler des foules sans l'aide de bandeaux. Pour cela, YaQ possède une architecture hybride qui permet de gérer la planification de déplacement de centaines d'agents en temps réel, grâce à une approche liée au niveau de détail (voir 2.5.3). C'est-à-dire que dans les régions intéressantes (proches de la caméra ou d'un événement important) les déplacements seront réellement simulés, alors que dans les régions moins importantes une approche statistique sera utilisée.

Il est également intéressant de souligner le logiciel **MASSIVE** de [Lind 1999] utilisé notamment pour simuler des foules pour le cinéma.

Toutes ces approches apportent des éléments intéressants en vue de simuler un grand nombre d'agents, mais elles se focalisent sur l'aspect graphique et les déplacements plutôt que le processus décisionnel.

2.5.2 Jeux

Le jeu vidéo est également un domaine très actif s'intéressant à la simulation de personnage virtuels, que ce soit :

- en tant que foule et élément du décors (par exemple dans les jeux de simulation de ville comme *Simcity*),
- en tant que personnage secondaire pouvant interagir avec le héros et avoir un impact dans le jeu (notamment dans les jeux de type aventure dans lesquels le joueur joue le rôle du héros, par exemple *Grand Theft Auto*, *The Elder Scrolls*, *World of Warcraft*, etc.),
- en tant que général commandant une grande armée (dans les jeux de simulation militaire, type *Total War*),
- en tant qu'ennemi du joueur (dans les jeux de tir à la première personne (*First Person Shooter* par exemple *Call of Duty*),
- ou en tant qu'allié qui aide le joueur (jeux de tir à la première personne au sein d'une équipe dont les membres sont gérés par l'ordinateur, typiquement

F.E.A.R.),

- etc.

Quels que soient les rôles qui leur sont dévolus, les personnages virtuels de jeu vidéo ont rarement besoin de capacités cognitives étendues. En effet, dans ce domaine la majorité des ressources est utilisée ailleurs (graphisme notamment), et l'objectif principal du personnage virtuel est généralement d'être cohérent, et réaliste. Les techniques développées et utilisées dans le jeu vidéo ne sont pas toujours utilisées afin de passer à l'échelle (et simuler un plus grand nombre de personnages non-joueurs), mais bien souvent elles permettent de gérer les agents de manière légère afin d'économiser les ressources pour d'autres besoins. Ces techniques sont donc intéressantes pour nous, dans l'optique du passage à l'échelle.

Les scripts restent très utilisés. Très peu flexibles, un script consiste à définir à l'avance, le comportement de l'agent. Ils sont fréquemment utilisés comme guides pour maintenir le scénario dans la direction souhaitée. Un peu plus flexible que les scripts, les machines à états finis (*Finite State Machine*) sont très souvent utilisées également. Une machine à états finis est composée d'un ensemble d'états, et de transitions entre ces états. Dès que les conditions de changement d'état sont validées, le personnage change d'état. A chaque état correspond un type de comportement ou un ensemble d'actions. Ainsi un agent de ce genre sera capable de prendre en compte le contexte pour modifier son comportement.

Depuis 2004, avec la sortie du jeu Halo 2, un nouveau concept apparaît dans le jeu vidéo : les arbres comportementaux (*Behavior Trees*). A l'heure actuelle, cette technique est l'une des plus utilisées. Un arbre comportemental est un arbre dont chaque feuille est une action que l'agent peut réaliser, et à chaque nœud correspond une méthode permettant de calculer si la branche correspondant à ce nœud est intéressante à explorer ou non. Ainsi, en fonction de son contexte courant, l'agent explorera la branche de l'arbre la plus intéressante et réalisera l'action la plus pertinente.

Dans SpirOps AI [spi 2005], une autre approche est utilisée, elle est appelée approche orientée objectifs (*drive oriented*). Dans ce type d'approche, les agent ne changent pas d'état comme dans une machine à états finis, mais sont continuellement dirigés par plusieurs objectifs, pouvant être satisfaits en parallèle (ce qui est impossible avec les machines à états finis ou les arbres comportementaux).

L'approche **GOAP** (*Goal-Oriented Action Planning*) par [Orkin 2006] est notamment utilisée dans des jeux tels que FEAR2, Fallout3, Empire : Total War, Just Cause 2, Deus Ex : Human Revolution, etc... GOAP est une architecture possédant un module de planification de type STRIPS, conçu spécialement pour le le contrôle en temps réel des personnages non joueurs du jeu.

[Evans 2009] propose également dans Les Sims3, une approche assez originale de l'intelligence des personnages non-joueurs, notamment en ce qui concerne le niveau de détail de l'intelligence des agents. Les personnages non-joueurs non présents à l'écran ne sont pas simulés de manière précise. Des fonctions statistiques "d'auto-satisfaction" sont créées, ce qui permet par exemple d'avoir de fortes chances de croiser des Sims ayant faim lorsque l'heure du repas approche, alors que ces mêmes sims seront assez probablement rassasiés après l'heure du repas. De plus, des fonctions de plus haut niveau gèrent les mécanismes de grande échelle tels que les naissances, les décès, les déménagements, etc.

Toutes ces approches permettent de simuler un grand nombre d'agents, grâce à une simplification de la modélisation de chacun. Cette solution n'est pas envisageable dans notre cas, puisque nous devons garder une forte crédibilité comportementale.

Dans le domaine du jeu vidéo, un certain nombre de travaux s'intéressent à l'équilibre de la difficulté du jeu, par rapport aux performances du joueur [Andrade 2005]. Ce domaine de recherche présente un intérêt fort par rapport à notre problématique, surtout lorsqu'il est mis en relation avec le domaine de la **narration interactive** (*Interactive Storytelling*) [Cavazza 2002]. Ces travaux permettent d'adapter le déroulement d'un scénario (que ce soit dans un jeu vidéo, ou dans un autre domaine d'application) aux actions du joueur afin qu'il respecte un certain nombre de contraintes imposées par le modélisateur. Une grande partie de la difficulté de l'approche repose sur le fait que les actions du joueur peuvent être imprévisibles, ou non rationnelles, et que le scénario doit tout de même suivre son cours, sans pour autant le rendre trop rigide. Des algorithmes de planification peuvent notamment être intégrés à ces travaux [Porteous 2010], permettant ainsi de mieux gérer les problèmes liés à la scalabilité et à la réactivité en temps-réel.

Ces recherches pourront être d'une grande aide pour les modélisateurs de simulations. En effet, même en l'absence de joueur, la grande autonomie de nos agents peut rendre le déroulement des scénarios imprévisible. Pour cette raison, avoir des outils permettant l'adaptation de l'histoire en fonction des actes protagonistes est primordial.

2.5.3 Niveau de détail

Quand on cherche à augmenter le nombre d'agents modélisés et simulés en parallèle, plutôt que de simplifier l'ensemble des agents, ou la manière dont est gérée l'environnement, une solution possible est de modifier dynamiquement la complexité des agents en fonction de leur *importance*. [Wissner 2010] donne un compte-rendu des diverses approches liées au niveau de détail utilisées en intelligence artificielle, et notamment dans la simulation de comportements. Les premiers travaux dans cette direction datent de 2002 [Brockington 2002].

Les travaux de [Navarro 2011] se focalisent sur le lien entre niveau de détail et simulation urbaine. Le point de départ de leur réflexion est la constatation que dans une simulation il y a un compromis à faire entre niveau de détail de la représentation de chaque agent, et passage à l'échelle (c'est-à-dire nombre d'agents simulés). Plus les agents sont finement modélisés et simulés, plus leur niveau de détail est élevé, et plus leur simulation est coûteuse en temps de calcul, et donc moins d'agents il sera possible de simuler. Le système proposé prend en compte le niveau de détail comme un paramètre intégré à part entière dans le modèle, qui peut varier dynamiquement au cours de la simulation en prenant en compte les événements de l'environnement, mais également les centres d'intérêts des observateurs. Ainsi, le comportement des agents sera modélisé avec un haut niveau de détail dans les zones proches des points d'intérêt, et avec de moins en moins de précision, au fur et à mesure qu'ils s'éloignent de ces zones. Les points d'intérêts peuvent être définis avant le lancement de la simulation en fonction des objectifs de celle-ci, par exemple des zones particulièrement importantes, ou des lieux clés de la simulation. Mais ces points d'intérêt dépendent également de ce que l'observateur est en train d'observer. Par exemple s'il zoome à un endroit précis du monde, tous les agents situés à cet endroit doivent être modélisés avec un haut niveau de détail, même si ce n'était pas le cas l'instant précédent.

Dans cette approche, la baisse du niveau de détail se traduit par la capacité à agréger des agents. Pour cela, le système calcule des distances spatiales et psychologiques entre agents d'une même zone. La distance psychologique représente la proximité psychologique de deux agents, c'est-à-dire à quel point ils partagent les mêmes buts, les mêmes émotions, ou les mêmes intentions. Si ces deux distances sont suffisamment faibles (i.e. si les agents sont suffisamment "proches" physiquement et psychologiquement), le système les agrège en une nouvelle entité. Plusieurs agents peuvent ainsi être agrégés dans un même groupe, et ce groupe sera piloté par un seul cerveau, ce qui permet un gain CPU non négligeable. Bien sûr, si un groupe entre dans une zone d'intérêt, il sera désagrégé.

La solution d'un niveau de détail dynamique de la modélisation décisionnelle des agents parait être une solution très prometteuse. Elle permettrait de garder une forte complexité décisionnelle (avec notamment un processus d'anticipation) pour les agents importants, et de la simplifier pour les agents moins importants, ce qui permettrait de passer à l'échelle sans perdre de crédibilité.

2.6 Synthèse

L'étude des différents travaux existants va permettre de nous situer par rapport à l'état de l'art. Les tableaux 2.1, 2.2, et 2.3 regroupent les principaux systèmes, architectures, ou théories présentées précédemment et les classent suivant nos objectifs qui sont : généricité de l'architecture, flexibilité des comportements, crédibilité des comportements, et facilité de passage à l'échelle.

En ce qui concerne les architectures réactives (voir tableau 2.1), la subsomption manque de généricité et de flexibilité, mais limite fortement la complexité du processus décisionnel et permet facilement de passer à l'échelle.

Architectures / Caractéristiques	Subsomption	Propagation d'utilités	Hiérarchie à libre-flux	Architecture avec mécanisme de vote	Architecture avec tableau noir	Architecture motivationnelle
Généricité du modèle	Hiérarchie prédéfinie	Réseau d'actions complexe	Graphe comportemental réactif	Modules simples	Module de contrôle complexe	Uniquement des motivations
Flexibilité du processus décisionnel	Winner-take-all	Réseau d'activation prédéfini	Réseau d'activation prédéfini	Vote simple	Pas de langage commun	Compromis entre motivations
Crédibilité des comportements	Manque de souplesse	Architecture purement réactive	Architecture purement réactive	Architecture purement réactive	Comportements crédibles	Uniquement des motivations
Facilité de passage à l'échelle	Simplicité du processus décisionnel	Simplicité du processus décisionnel	Simplicité du processus décisionnel	Simplicité du processus décisionnel	Processus décisionnel très complexe	Processus décisionnel complexe

TABLE 2.1 – Tableau de synthèse des architectures réactives

Les réseaux de propagation, en n'utilisant pas de structure hiérarchique, permettent de rendre le processus décisionnel un peu moins rigide, mais ce réseau peut vite devenir complexe à organiser et modifier.

La hiérarchie à libre-flux utilise une hiérarchie, mais permet d'éviter le problème du *winner-take-all* grâce à la recherche de comportements de compromis. Cependant son caractère uniquement réactif pose des problèmes de crédibilité lorsqu'il s'agit de modéliser des comportements humains. La solution proposée dans DirectIA permet de combler ce manque, grâce à un mécanisme de planification réactive.

Les architectures réactives avec mécanisme de vote font un premier pas vers plus de flexibilité et de généricité, mais les processus et modules mis en œuvre sont trop simples pour être suffisants dans le cas de comportements complexes.

Les architectures avec tableau noir font un pas supplémentaire vers la généricité et la flexibilité, tout en se positionnant clairement par rapport à l'objectif de crédibilité. Seulement la gestion d'un tableau noir ralentit considérablement le processus décisionnel, et n'est pas compatible avec la nécessité de simuler un grand nombre d'agents.

Enfin, les architectures motivationnelles offrent un cadre intéressant, mais la restriction des sources comportementales aux seules motivations interdit toute généricité.

Par rapport aux architectures cognitives (tableau 2.2), les architectures de planification délibérative visent une bonne crédibilité comportementale, mais le processus décisionnel est un peu lourd, et l'utilisation d'un planificateur impose l'unicité du modèle utilisé.

Dans ce domaine, les architectures SOAR et ACT-R proposent une avancée significative en prenant en compte différents types de mémoires, mais elles restent

Architectures / Caractéristiques	Architecture de planification délibérative	SOAR/ACT-R	Architecture de planification réactive	Architecture BDI	PEP → BDI
Généricité du modèle	Modèle unique	Différents types de mémoires possibles	Modèle unique	Modèle unique	Prise en compte des émotions
Flexibilité du processus décisionnel	Planification délibérative	Planification délibérative	Planification réactive	Planification réactive	Planification réactive
Crédibilité des comportements	Comportements crédibles	Comportements crédibles	Utilisé en robotique	Comportements crédibles	Comportements crédibles
Facilité de passage à l'échelle	Processus décisionnel complexe	Processus décisionnel complexe	Processus décisionnel complexe	Processus décisionnel complexe	Processus décisionnel complexe

TABLE 2.2 – Tableau de synthèse des architectures cognitives

fermées à l'ajout de modèles externes.

Les architectures de planification réactive allègent légèrement le processus décisionnel, mais ne permettent pas non plus l'ajout de nouveaux modèles.

Les architectures BDI offrent une autre possibilité de modélisation, visant également la crédibilité des comportements. Mais le modèle est également fermé, et l'ajout de théories supplémentaires n'est pas gérée.

Le modèle PEP → BDI, en permettant d'intégrer les émotions des agents dans la décision augmente la généricité de l'architecture, mais ne permet toujours pas d'ajouter des modèles externes.

Les architectures hybrides permettent de se rapprocher plus près de nos 4 critères (voir tableau 2.3). INTERRAP, grâce à sa hiérarchie organisée en couches, permet d'obtenir des processus décisionnels simples. Mais cette architecture manque de souplesse, et a un fonctionnement assez proche d'un maître-esclave.

Architectures / Caractéristiques	INTERRAP	PECS	ICARUS	MhiCS	Polyscheme	CoSy
Généricité du modèle	Architecture 3 couches	Modules prédéfinis	Modules prédéfinis	Modules prédéfinis	Besoin d'un traducteur	Modèle générique
Flexibilité du processus décisionnel	Architecture maître-esclave	Winner-take-all	Architecture maître-esclave	Processus flexible	Processus flexible	Processus flexible
Crédibilité des comportements	Manque de souplesse	Manque de souplesse	Comportements crédibles	Manque de complexité	Comportements crédibles	Comportements crédibles
Facilité de passage à l'échelle	Processus décisionnel simple	Processus décisionnel simple	Processus décisionnel simple	Processus décisionnel simple	Processus décisionnel complexe	Processus décisionnel très complexe

TABLE 2.3 – Tableau de synthèse des architectures hybrides

PECS, malgré une organisation de ses modules entièrement différente, et non hiérarchique, souffre des mêmes limitations. L'architecture manque de généricité puisque les modules utilisés sont prédéfinis, et son fonctionnement en *winner-take-all* interdit toute flexibilité comportementale.

ICARUS se positionne plus clairement que les deux architectures précédentes sur l'objectif de crédibilité des comportements, tout en gardant un processus décisionnel assez simple. Mais là encore les modules sont prédéfinis, et les différentes couches comportementales ne coopèrent pas réellement, mais fonctionnent plutôt sur des relations maître-esclave.

MhiCS garde toujours la forte limitation de généricité due à l'utilisation de modules prédéfinis, mais le processus décisionnel utilisé est intéressant en raison de sa flexibilité et de sa simplicité. Il n'est cependant pas utilisé pour modéliser des comportements crédibles.

Polyscheme se positionne de manière intéressante sur le plan de la généricité, puisque cette architecture est capable de prendre en entrée des modules divers. Cependant, afin que leur avis soient correctement pris en compte dans le processus décisionnel, chacun de ces modules a besoin d'un traducteur, ce qui rend le processus de décision un peu plus complexe, et ne simplifie pas la tâche de l'ajout de modèles.

CoSy, par contre, possède un fonctionnement entièrement générique et flexible, et les comportements modélisés sont crédibles. Seulement la satisfaction de ces 3 critères se fait au détriment du 4°, à savoir la complexité du processus décisionnel.

2.7 Conclusion

Comme le soulignent les tableaux précédents, aucune architecture préexistante ne regroupe l'ensemble des caractéristiques désirées. Notre objectif principal dans cette thèse sera de développer une architecture satisfaisant l'ensemble des points présentés.

Pour cela nous réutiliserons certaines caractéristiques de plusieurs des modèles présentés. Le principe de décomposition des comportements et de sélection retardée, utilisé dans la hiérarchie à libre flux, sera une source d'inspiration pour notre module décisionnel. Afin d'indiquer l'importance qu'un comportement a, pour un module comportemental quelconque, nous garderons en tête le mécanisme de vote utilisé dans DAMN. Les architectures motivationnelles seront utilisées en tant que source comportementale de base des agents, c'est-à-dire qu'en l'absence d'autres modèles, nos agents seront dirigés par leurs motivations.

Afin d'augmenter la pertinence des comportements, en particulier sur le moyen terme, nous utiliserons un mécanisme dérivé de la planification réactive, permettant aux agents d'enchaîner de manière cohérente leurs actions, sans non plus chercher à les optimiser. L'architecture SOAR, qui est peut-être un des standards les plus reconnus dans le domaine des architectures cognitives nous inspirera, en particulier en ce qui concerne la gestion des connaissances. Les différentes architectures BDI ont également servi de modèle, notamment dans la construction du processus

décisionnel.

L'idée que nous retenons d'INTERRAP est la distinction qui est faite entre les situations ne nécessitant pas l'intervention de la partie cognitive, et celles dont la partie réactive ne suffit pas à résoudre les problèmes. Seulement dans notre architecture nous souhaitons que, quelle que soit la complexité de la situation, l'ensemble des parties (ou plus précisément l'ensemble des modules) participent à l'élaboration du comportement, afin que chaque décision prise par l'agent reflète l'ensemble de ses désirs et réflexions, et pas seulement une partie d'entre elles.

L'architecture PECS est particulièrement importante, car c'est peut-être l'architecture qui a le plus inspiré la structure de la nôtre. Les deux principales différences tiennent au fait que nous ne fixons pas les modules comportementaux qui seront utilisés, et que ces modules ne fonctionnent pas en *winner-take-all*.

Polyscheme et CoSy ont, quant à elles, donné des pistes intéressantes par rapport aux mécanismes de communication entre l'ensemble des modules comportementaux et le module de décision, ainsi que sur la manière d'ouvrir le système à de nouveaux modules.

Aucune architecture de l'état de l'art ne satisfait l'ensemble de nos contraintes. De nombreuses architectures sont très pertinentes par rapport à certaines de nos problématiques, mais ne permettent pas de répondre à certaines autres.

Chapitre 3

Une architecture pour agents virtuels en environnement urbain

Sommaire

3.1 Introduction

La première contribution de cette thèse repose sur le développement d'une architecture décisionnelle pour agents virtuels. Elle est adaptée à la modélisation et à la simulation d'humains virtuels en environnement urbain, mais grâce à ses caractéristiques de généricité et de flexibilité, elle peut être utilisée dans des domaines très variés pour des agents très différents. Ce travail de modélisation de l'architecture est une contribution réalisée dans le cadre d'un travail collectif, piloté au LIP6 par Étienne de Sevin et Vincent Corruble pendant le projet Terra Dynamica.

Dans ce chapitre nous donnerons tout d'abord quelques définitions de base, qui permettront de clarifier les concepts utilisés, puis nous préciserons le formalisme que nous avons utilisé. Nous verrons ensuite la place de cette architecture décisionnelle au sein de l'architecture d'agents globale utilisée dans le projet. Et nous nous attacherons à identifier avec précision les entrées et les sorties de notre module, dans le but de les rendre les plus génériques possible, en vue de l'intégration de cette décision dans d'autres architectures agents. Dans un troisième temps nous nous intéresserons aux intérêts et contraintes de l'approche choisie, et aux raisons qui nous ont poussés à développer une telle approche. Nous présenterons précisément le modèle et le formalisme utilisés, ainsi que les mécanismes qui sous-tendent cette architecture décisionnelle. Une fois cette base établie, nous détaillerons l'instanciation de cette architecture réalisée dans le cadre du projet Terra Dynamica, les différents modules incorporés et leurs utilités.

3.2 Définitions

Afin de clarifier les concepts abordés ainsi que le formalisme utilisé, nous donnons ici un résumé du vocabulaire utilisé. Il est assez classique, majoritairement tiré de [Russell 2010].

Nous appelons **comportement**, un ensemble organisé d'actions qu'un agent peut adopter au sein d'une simulation. *Organisé* signifie que ces actions ne peuvent pas toujours être réalisées dans n'importe quel ordre, mais qu'elles doivent parfois être ordonnées (utilisation d'un graphe de comportements "et/ou", voir partie 3.3.1).

Notation : Nous notons $C = \{C_1, ..., C_n\}, n$ étant le nombre total de comportements, l'ensemble des comportements existants dans la simulation.

Un comportement peut être décomposé en **sous-comportements**.

Exemple : Soit le comportement "manger". Ce comportement peut être décomposé dans les sous-comportements "manger au restaurant", "manger chez soi", ou "grignoter".

Les sous-comportements peuvent également être décomposés en sous-sous-comportements, et ainsi de suite jusqu'au niveau des **actions élémentaires** ou **actions**. Une action est un comportement particulier qui ne peut pas être décomposé. Les actions représentent le niveau atomique des comportements, et sont effectivement exécutées dans la simulation.

Exemple : Reprenons l'exemple précédent. Le comportement "manger" peut être décomposé dans les sous-comportements "manger au restaurant", "manger chez soi", ou "grignoter". Le sous-comportement "manger au restaurant" étant composé des actions "choisir un restaurant", et "manger dans un restaurant". Le sous-comportement "manger chez soi" étant lui composé des actions "faire les courses", "préparer à manger", et "manger chez soi". Le comportement "grignoter" pour sa part ne peut pas être décomposé : c'est une action élémentaire.

En fonction de la simulation, des objectifs visés, de la complexité désirée des agents et de leurs comportements, la **complexité** des comportements peut varier. Dans cette thèse, nous appelons complexité d'un comportement sa hauteur dans le graphe décisionnel, c'est-à-dire le nombre maximal de décompositions entre le comportement et n'importe quelle action qui le compose.

Exemple : Dans une simulation portant sur les habitudes alimentaires des habitants d'un quartier, l'action "manger au restaurant" utilisée dans les exemples précédents, pourrait devenir un sous-comportement lui-même comportant des actions beaucoup plus précises comme "choisir un menu", "manger vite", "déguster son repas", "prendre un café", "boire du vin", etc.

L'ensemble des comportements, ainsi que leurs ensembles respectifs de sous-comportements, forment le **graphe comportemental** de la simulation. Ce graphe comportemental et les relations entre comportements qui le composent, est à la base du processus décisionnel. Le graphe comportemental, ainsi que les notions de complexité, seront détaillés avec précision dans la partie 3.3.1.

Une **action concrète** est l'instanciation d'une action élémentaire dans l'environnement, c'est-à-dire la prise en compte, en pratique, de la réalité environnementale concrète dans une action abstraite. Pour la majorité des actions,

l'environnement proche influence la réalisation.

Exemple : Soient un agent A, l'action élémentaire a_e = "Manger au restaurant", et le restaurant r situé dans l'environnement. Alors l'action concrète a_c associée à a_e et r est : a_c = "Manger dans le restaurant r".

Dans un environnement donné, il existe donc autant d'actions concrètes que de manière différentes de les réaliser.

Un **but** est un ensemble d'états du monde, que l'agent souhaite atteindre. Notons que ce but peut également être la réalisation d'un comportement ou d'une action, et non les effets de cette action ou comportement.

Exemple : Un agent peut avoir le but de se trouver à un endroit précis de l'environnement à un moment donné. Ou d'accomplir le comportement "envoyer du courrier" avant une certaine heure.

Un **plan** est un ensemble organisé d'actions menant à la réalisation d'un but. Un plan peut prendre en compte plusieurs buts en parallèle, chacun pouvant être décomposés en sous-buts.

Exemple : Soit A un agent devant envoyer une lettre et aller au travail. Le but "envoyer une lettre" correspond à un comportement composé des sous-comportements "acheter un timbre" et "poster une lettre". Le but "aller au travail" est réalisé lorsque l'agent est situé sur son lieu de travail. Un plan pour réaliser ces 2 buts serait : se rendre à la poste pour acheter les timbres et poster la lettre, puis se rendre au travail. Notons qu'il existe des plans alternatifs : se rendre dans un bureau de tabac pour acheter un timbre, puis chercher une boite aux lettres sur le chemin du travail. Ou : se rendre au travail, et poster la lettre plus tard.

L'**instanciation d'un plan** est un ensemble organisé d'actions concrètes, c'est-à-dire la prise en compte dans un plan du contexte de l'agent et de son environnement.

Exemple : Reprenons l'exemple précédent. Supposons que A connaisse l'emplacement d'une poste ($poste_1$), d'un bureau de tabac ($tabac_1$) et de deux boîtes aux lettres ($boite_1$ et $boite_2$). Des instanciations de plan permettant de remplir le but "envoyer une lettre" sont : "acheter un timbre à $poste_1$", puis "poster une lettre à $poste_1$" ou "acheter un timbre à $tabac_1$" puis "poster une lettre à $boite_1$" ou encore "acheter un timbre à $tabac_1$" puis "poster une lettre à $boite_2$".

Notons que les instanciations de plan du type "acheter un timbre à $poste_1$", puis "poster une lettre à $boite_x$" ou "acheter un timbre à $tabac_1$", puis "poster une lettre à $poste_1$" existent également, même s'ils sont clairement sous-optimaux.

3.3 Formalisme utilisé

3.3.1 Graphe comportemental

Comme cela a été expliqué dans la section précédente, nous nous basons sur un **graphe comportemental**, regroupant l'ensemble des comportements que les agents peuvent sélectionner dans une simulation donnée. Ce formalisme est très utilisé à l'heure actuelle et possède de nombreux avantages [Millington 2009]. Il permet notamment de regrouper l'ensemble des comportements et des actions réalisables dans la simulation dans un même ensemble, et d'indiquer les relations entre ces comportements. Ce graphe est enrichi de nombreuses relations qui permettent de tenir compte des spécificités des comportements.

Dans notre modèle, un unique graphe comportemental est généré lors du lancement d'une simulation. Ce graphe sera noté G. Chaque nœud du graphe est un comportement, et chaque feuille une action. Ce graphe est conçu de telle sorte qu'il permet de déterminer l'ensemble des plans possibles pour réaliser chaque comportement. Les agents n'auront donc pas besoin de passer du temps à chercher ces plans, ils sont connus lors du lancement de la simulation. Cela permet un gain de temps important.

Notons que nous enrichissons ce graphe d'un ensemble de relations, permettant d'ajouter des contraintes entre les comportements.

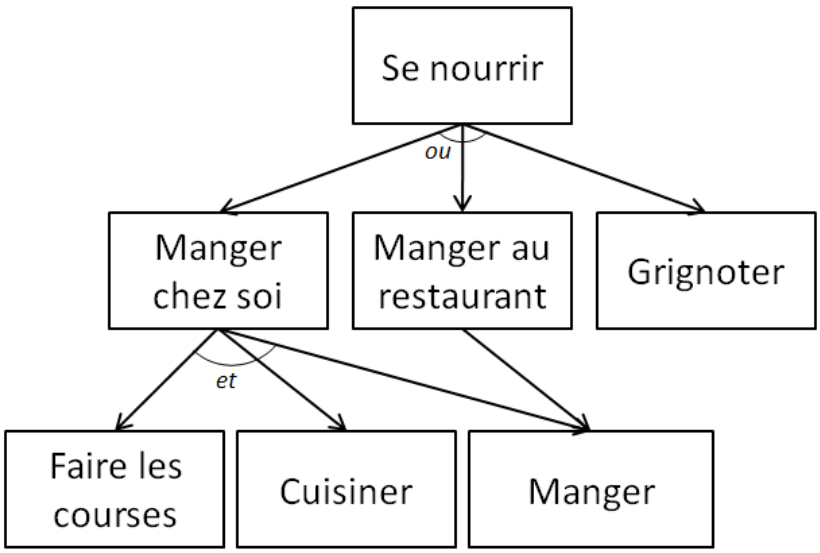

FIGURE 3.1 – Exemple de graphe comportemental

On parle souvent dans la littérature d'**arbre comportemental** (*behavior tree*). Un arbre est "un graphe non orienté, acyclique et connexe". Si cette appellation peut s'avérer correcte dans certains modèles, ce n'est pas le cas ici, puisqu'une action peut faire partie de plusieurs comportements, créant ainsi un treillis (exemple figure 3.1,

l'action "manger" appartient aux comportements "manger chez soi" et "manger au restaurant").

Ce graphe est un graphe *et/ou*. En effet, chaque nœud peut se décomposer de deux manières différentes. Soit le comportement nécessite la réalisation d'un ensemble ordonné de sous-comportements, dans ce cas c'est un nœud *et*, soit il nécessite la réalisation d'un sous-comportement parmi un ensemble possible, dans ce cas c'est un nœud *ou*. Notons qu'un arbre "et/ou" est transposable en simple arbre "ou", mais que ces derniers sont beaucoup plus lourds et peu explicites.

Exemple : Afin d'accomplir le comportement "se nourrir", un agent peut soit "manger chez soi", soit "manger au restaurant" : c'est un nœud *ou*. Par contre, pour réaliser le sous-comportement "manger chez soi", un agent doit effectuer les actions "faire les courses", et "cuisiner", et "manger", c'est un nœud *et*.

Dans ce graphe, le comportement "se nourrir" a une complexité de 2 car il existe des actions permettant de réaliser ce comportement qui se trouvent "2 niveaux de décomposition" plus bas. Notons qu'une action est un comportement dont la complexité est nulle.

Un exemple complet de graphe décisionnel est donné dans l'annexe B.

3.3.2 Actions

Afin de définir les actions, nous choisissons un modèle assez classique. Les actions ont les caractéristiques suivantes :

- **Des critères de coût**. Les critères de coût sont dépendants de la simulation, ce qui permet d'augmenter la généricité du modèle en permettant au modélisateur de choisir et de définir les coûts pertinents à prendre en compte en fonction de la simulation visée (voir partie 3.3.3).
- **Des effets**. A priori, toutes les actions ont un effet sur l'environnement ou sur l'agent qui l'exécute. Il existe de nombreux effets différents (plus de détails dans la partie 3.3.4).
- **Des actuateurs**. En fonction de la simulation, le modélisateur définit un certain nombre d'actuateurs, dont les agents sont dotés. Chaque action qu'un agent peut effectuer utilise un certain nombre de ces actuateurs. Le nombre d'actions simultanées qu'un agent peut accomplir est donc limité par ses actuateurs disponibles (voir partie 3.3.5).
- **Des préconditions**. Pour qu'une action puisse être exécutée, il peut être nécessaire qu'un certain nombre de conditions soient vraies. Ces préconditions peuvent être de plusieurs types, voir partie 3.3.6.
- **Des liens entre actions**. Les actions sont enrichies de deux types de relations permettant des articulations entre actions plus complexes. Ces liens sont détaillés dans la partie 3.3.7.

- **Des animations**. Les animations sont la représentation visuelle des actes des agents dans la simulation. Leur objectif est de faire passer des informations à un observateur. Plus de détails dans la partie 3.3.8.

3.3.3 Critères de coût

Une dimension importante du modèle est sa volonté de généricité. Il nous semble primordial de limiter le moins possible les domaines d'applications, ou les types de simulations pouvant être modélisées. Évidemment, une généricité absolue n'est pas possible. Tous les modèles présentent des contraintes, mais nous allons essayer d'augmenter cette généricité autant que possible. C'est pourquoi la définition même des actions n'est pas figée. En effet, les actions sont à la base de toute la pyramide comportementale, ce sont les particules élémentaires du comportement. Toute contrainte à leur encontre implique une contrainte pour le modèle tout entier.

Partant du principe que les actions peuvent avoir des importances et rôles différents selon les simulations, nous avons laissé la possibilité au modélisateur de rajouter des critères de coût sur les actions. En effet, si des coûts tels que le prix de l'action (argent) ou le temps nécessaire (durée de l'action) sont des critères classiques pour une simulation urbaine, d'autres peuvent être importants dans d'autres types d'applications. Par exemple dans des simulations liées à des scénarios de gestion de risque ou de combat, la notion de dangerosité des comportements peut prendre une place primordiale, et nécessiter d'être considérée comme un critère de coût à part entière. La liste des critères utilisés lors d'une simulation est donc entièrement paramétrable (voir annexe A.1.2).

Par défaut seuls deux critères de coût sont utilisés : le coût monétaire et le coût temporel. Ces deux grandeurs semblent naturelles lorsqu'il s'agit de caractériser et de comparer des actions typiquement liées à une activité humaine. D'autres critères, par exemple le coût énergétique, sont envisageables, mais dans le cadre du projet et des diverses simulations réalisées, nous n'avons jamais eu besoin de plus de détails.

Les critères de coût doivent être objectifs puisqu'ils sont relatifs aux comportements, et non aux agents. Les appréciations subjectives des actions ou des comportements qu'un agent peut avoir (par exemple la qualité) seront traitées par un autre biais (voir la section sur les préférences des agents : 3.5.4).

3.3.3.1 Coût monétaire

Le coût monétaire d'une action représente la somme d'argent nécessaire à son accomplissement. Dans notre modèle nous choisissons de l'indiquer sous forme d'une fourchette de prix. Lorsqu'un agent exécute cette action, un prix est tiré aléatoirement dans la fourchette, et cette somme est retirée de l'argent possédé par l'agent. Lorsqu'un agent planifie son comportement, s'il veut être certain d'avoir suffisamment d'argent pour réaliser ses buts, il est nécessaire qu'il s'assure d'avoir les sommes maximales requises par l'ensemble des actions qu'il prévoit d'effectuer.

Ce tirage aléatoire parmi une fourchette de prix, nous a semblé la manière la

plus générique de fonctionner. De cette façon il est possible d'introduire un peu de variabilité dans les actions, tout en gardant la possibilité de définir une fourchette dont les prix minimum et maximum sont identiques, ce qui revient à fixer un prix exact. De cette manière il devient possible de gérer grâce à la création d'une seule action une grande variété de comportements. Par exemple une action "faire du shopping" peut avoir une très large échelle de prix, ce qui permet de modéliser des comportements extrêmement variés.

L'argent est bien souvent absent des simulations multi-agents, mais nous avons décidé d'en faire une des composantes principales de notre modèle, car la plupart des actions effectuées par des humains en milieu urbain sont comparables à des services, mettant en jeu, même très indirectement, un échange payant entre personnes.

3.3.3.2 Coût temporel

Le critère de coût le plus universellement utilisé est sûrement celui de la durée. C'est un critère quasi indispensable, puisqu'il est impossible de réaliser une simulation sans connaître la durée des différentes actions pouvant être exécutées (à moins de ne simuler que des agents ne réalisant aucune "action", mais seulement des déplacements par exemple).

Nous avons également choisi de la donner sous forme d'une plage temporelle. Lorsqu'un agent exécute une action, la durée nécessaire pour la terminer est tirée aléatoirement dans la plage de temps.

Pour les mêmes raisons que pour le prix, l'aspect aléatoire de la durée permet une plus grande liberté de modélisation. Par exemple l'action "faire du shopping", si elle est caractérisée par une plage temporelle très large, peut durer très peu ou très longtemps, de manière pas toujours planifiée (même une action que nous réalisons tous les jours ne nous demande pas toujours le même temps).

3.3.4 Effets

C'est uniquement par le biais des actions que les agents sont capables d'agir avec leur environnement. S'ils veulent modifier un état du monde, ou un état interne ils sont obligés d'en passer par la réalisation d'un certain nombre d'actions, sélectionnées en fonction de leurs effets. Dans cette thèse, nous partons du postulat qu'une action peut avoir deux types d'effets : les effets externes, et les effets internes [Donnart 1998].

3.3.4.1 Effets externes

Tout d'abord, les actions peuvent modifier l'environnement. Dans notre modèle, l'environnement est composé d'un ensemble d'objets, dont les états sont paramétrables, et modifiables en cours de simulation. Ce sont ces objets qui seront touchés par les effets environnementaux des actions. Notons que nous considérons ici les autres agents comme faisant partie de l'environnement.

Exemple : Soit l'action "mettre le feu à une poubelle". Elle peut être exécutée par un agent sur n'importe quel objet de type "poubelle" situé dans l'environnement. Ses conséquences sur l'environnement pourraient être dépendantes du contenu de la poubelle. Par exemple, si la poubelle contient du papier, alors la poubelle passe dans l'état "en feu" pour une durée dépendante de la quantité de papier. Sinon rien ne se passe.

3.3.4.2 Effets internes

Le deuxième type d'effets que peuvent avoir les actions est celui affectant directement l'agent, via ses **variables internes** ou son **inventaire** (voir partie 3.5.5). En effet, les agents doivent avoir un inventaire spécifiant leurs possessions matérielles actuelles. La majorité des préconditions des actions reposent également sur ces variables d'inventaire, et souvent un agent se verra obligé de réaliser une action lui permettant d'obtenir une ressource d'inventaire nécessaire à la réalisation de l'action qui l'intéresse vraiment.

Exemple : Soit A un agent désirant "se désaltérer". Pour se faire, plusieurs possibilités s'offrent à lui. Il peut en particulier "acheter une boisson dans un distributeur", ce qui aura 2 effets sur son inventaire : lui retirer une certaine somme d'argent, et lui donner une canette de boisson. Grâce à cette canette il pourra réaliser l'action "boire une canette" lui permettant réellement de se désaltérer.

3.3.5 Actuateurs

La question de la parallélisation des actions est une question intéressante dans le cadre des humains virtuels. En effet, les humains sont capables de réaliser plusieurs actions en parallèle, par exemple respirer et écrire. Mais ils ne sont pas capables de réaliser n'importe quelle action en même temps que n'importe quelle autre. Différentes approches se sont développées pour répondre à cette question, mais nous nous baserons sur celle des **actuateurs** [Ferrell 1993]. Le principe est de considérer que les agents ont un certain nombre d'actuateurs, indépendants les uns des autres, qu'ils peuvent utiliser pour réaliser des actions. Les actions quant à elles, nécessitent un ensemble précis d'actuateurs pour être réalisées. Ainsi les agents peuvent réaliser autant d'actions en parallèle que souhaité, tant qu'aucun actuateur n'est utilisé par plusieurs actions en même temps.

Exemple : Supposons que les agents possèdent 3 actuateurs : un actuateur "tête", un actuateur "bras" (un seul actuateur pour les deux bras), et un actuateur "jambes". Soit une action a_1 : "boire un café" utilisant l'actuateur "bras" ; une action a_2 "marcher", utilisant l'actuateur "jambes" ; une action a_3 "discuter" utilisant l'actuateur "tête" ; et une action a_4 "téléphoner" utilisant les actuateurs "tête" et "bras".

Selon cette répartition, un agent pourra boire un café en marchant et en

discutant (les actions a_1, a_2 et a_3 ne partagent aucun actuateur commun), ou téléphoner en marchant (idem pour les actions a_2 et a_4), mais ne pourra pas téléphoner en buvant un café (actuateur "bras" utilisé 2 fois) ni téléphoner en discutant (actuateur "tête" utilisé 2 fois).

Évidemment, en fonction de la simulation désirée, les actuateurs pertinents ne sont pas les mêmes. Dans certain cas, le modélisateur peut avoir besoin d'augmenter le nombre des actuateurs (et créer par exemple un actuateur pour chaque bras de l'agent), alors que dans d'autres un unique actuateur peut suffire. Les actuateurs disponibles pour les agents sont donc paramétrables.

Notons qu'il est également possible d'affiner l'utilisation des actuateurs, en fractionnant leur utilisation ("boire un café" utilise 30% de l'actuateur "bras" et 20% de l'actuateur "tête"), ainsi les agents peuvent cumuler les actions en parallèle, tant qu'aucun actuateur n'est utilisé à plus de 100%.

3.3.6 Préconditions

Certaines actions ne peuvent être réalisées directement, elles ont besoin qu'un certain nombre de préconditions soient remplies avant de pouvoir être réalisées [Maes 1989].

Ces préconditions peuvent porter sur des objets devant être présents dans l'inventaire de l'agent. Ces objets peuvent être retirés lors de l'action (par exemple utiliser un ticket de métro pour entrer dans une station, ou payer pour un service). Ils peuvent également être uniquement des prérequis, et ne âs être retirés à l'agent lors de l'exécution de l'action (par exemple avoir un badge pour accéder à un bâtiment). Dans certain cas, la précondition peut être de ne pas avoir certains objets sur soi (exemple : ne pas avoir de baskets pour entrer dans une discothèque).

Les préconditions peuvent également porter sur la réalisation préalable d'un ensemble d'actions, ou de comportements (par exemple l'action "manger chez soi" nécessite l'exécution de l'action "mettre la table").

Les préconditions sont le principal élément entraînant le chaînage des actions en vue de réaliser un but. Si un agent veut obtenir les effets d'une action, mais qu'il ne satisfait pas les préconditions de cette action, il va chercher un ensemble d'autres actions lui permettant de les satisfaire, et ainsi de suite.

3.3.7 Liens entre actions

Le comportement humain est généralement considéré comme plus complexe que le comportement animal. Cependant, ce dernier possède déjà une grande diversité, et un certain nombres de particularités intéressantes. L'une d'entre elles, est le **schème d'action spécifique** (*Fixed Action Pattern*) [Lorenz 1985, Alcock 2001]. C'est une séquence comportementale, composée de plusieurs actions, qui se déroule toujours jusqu'à son terme, quel que soit le contexte. Si ces comportements spécifiques sont plus rares, voire absent chez l'homme, le comportement humain possède toutefois

des caractéristiques culturelles ou de "savoir-vivre" qui créent des liens très forts entre actions, de telle sorte que des schèmes d'actions spécifiques se forment.

Afin de retranscrire ces cas particuliers comportementaux, des liens entre actions sont ajoutés. Ces liens sont de deux types :

- **Antériorité**. Une action peut être dotée d'une relation d'antériorité par rapport à une deuxième action. Cette relation implique que la deuxième action doit être effectuée immédiatement après la première. L'action dotée du lien d'antériorité est donc antérieure à la seconde.
- **Postériorité**. Une action peut être dotée d'une relation de postériorité par rapport à une deuxième action. Cette relation implique que la deuxième action doit être effectuée immédiatement avant la première. L'action dotée du lien de postériorité est donc postérieure à la seconde.

Remarque : comme le montrent les exemples suivants, ces deux liens ne sont pas redondants.

Exemple : Il est possible de doter l'action "manger au restaurant" d'un lien d'antériorité par rapport à l'action "payer l'addition", afin d'interdire qu'un comportement tierce ne vienne s'intercaler (par exemple "retirer de l'argent"). De cette manière un agent désirant payer en liquide au restaurant prendra en compte le fait qu'il doit déjà avoir l'argent sur lui avant d'aller au restaurant. L'action "manger au restaurant" sera toujours antérieure à l'action "payer l'addition".

Dans cet exemple, l'action "payer l'addition" n'est pas forcément dotée d'un lien de postériorité par rapport à l'action "manger au restaurant". En effet, un agent peut effectuer l'action "payer l'addition" sans avoir effectué l'action "manger au restaurant" au préalable (par exemple s'il a bu un café).

Exemple : Pour compléter l'exemple, il est possible de considérer que l'action "manger au restaurant" est dotée d'un lien de postériorité par rapport à l'action "choisir son menu". Attention, dans ce cas un agent désirant "aller aux toilettes" devra le faire avant de commander son menu !

Par contre, après avoir effectué l'action "choisir son menu", un agent peut décider que le restaurant ne lui plaît pas, et décider de partir (ce qui est impossible si l'action "choisir son menu" est dotée d'un lien d'antériorité par rapport à l'action "manger au restaurant").

3.3.8 Animations

Grâce à des effets visuels et/ou sonores, les animations permettent de faire comprendre aux observateurs ce que les agents sont en train de faire : elles représentent donc un aspect important de la crédibilité des comportements. Cependant, pour le modélisateur d'une simulation, le panel des animations disponibles est une réelle contrainte, puisque seules les actions pouvant être associées à une animation existante peuvent être exécutées dans la simulation. En effet, si dans une simulation les

agents peuvent effectuer une action A, mais que cette action ne peut pas être associée à un signal sonore ou visuel correctement interprétable par les observateurs, de gros problèmes de compréhension risquent de se produire. Les observateurs peuvent ne pas comprendre ce que les agents sont en train de faire, croire qu'ils sont en train de réaliser une autre action, voire ne pas se rendre compte du tout qu'ils sont en train de faire quelque chose. C'est un grave échec pour la simulation. Lors de l'élaboration du graphe comportemental, il est donc important de veiller à la corrélation entre actions réalisables et animations existantes.

Exemple : si l'animation correspondant à "parler au téléphone" n'existe pas, l'action "appeler quelqu'un au téléphone" peut difficilement faire partie du graphe comportemental, sauf s'il existe un autre moyen de faire comprendre à un observateur ce que l'agent est en train de faire (texte explicatif, bulle informative, etc.). Dans notre cas, comme le réalisme de la simulation est important (graphisme de bonne qualité, en 3 dimensions) nous ne pouvons pas recourir à ces subterfuges et nous sommes limités aux animations disponibles.

3.4 Architecture décisionnelle générique

3.4.1 Introduction

Dans cette partie nous allons présenter l'architecture générique développée dans le cadre de cette thèse. Comme nous l'avons présenté dans la partie 2.6, notre objectif est de développer une architecture d'agents répondant à 4 critères principaux, qui sont la généricité (1) et la flexibilité (2) du modèle, et sa capacité à simuler des comportements crédibles (3) pour un grand nombre d'agents dans un environnement complexe (4).

L'objectif de généricité est né d'une constatation simple. De nombreux domaines de recherche s'intéressent de très près à la modélisation agent. On peut citer l'intelligence artificielle, la robotique, la psychologie cognitive, la simulation multi-agents, la planification, etc. Tous ces domaines ont développé des techniques très différentes les unes des autres afin de modéliser des agents répondant à leurs besoins propres. Classiquement, lorsqu'une architecture d'agents est développée, elle se place dans l'un de ces courants de recherche et adopte les techniques associées qui permettent d'obtenir des agents adaptés à leurs objectifs.

Plutôt que de choisir l'un des paradigmes proposés dans l'état de l'art, nous cherchons à offrir une structure permettant l'incorporation de n'importe quelle théorie, système, ou sous-architecture appartenant aux divers domaines. Avec une modélisation de la sorte, il devient possible de développer un large panel d'agents très différents les uns des autres, qui sont adaptés à un large panel d'applications. Nous allons même plus loin, et considérons que chacune de ces théories peut être considérée comme une manière de modéliser un processus cognitif spécifique ou un aspect du comportement/raisonnement d'un agent. Si un ensemble de ces théories sont accessibles à partir d'une même architecture, il est

envisageable d'en sélectionner certaines, et d'en ignorer d'autres, afin de modéliser un type d'agent bien précis couplant plusieurs de ces aspects. Non seulement la diversité des agents modélisables augmente, mais leur complexité augmente également [de Sevin 2012a]. Cet objectif a pour effet de transformer l'architecture d'agents en une véritable **méta-architecture**, dotée d'une bibliothèque de modèles.

L'objectif de flexibilité découle en quelque sorte de la généricité désirée. En effet, intégrer dans une même architecture d'agents des modèles hétérogènes n'est pertinent que si le processus décisionnel est capable de traiter des comportements eux-mêmes hétérogènes. Pour cela il faut que le processus décisionnel soit flexible, capable de prendre en entrée des comportements de complexité, durée, importance, et source différents.

Comme nous l'avons vu dans l'état de l'art (voir partie 2.6), aucune architecture pré-existante ne rassemble l'ensemble de ces 4 critères. Il est donc nécessaire de développer un nouveau modèle.

3.4.1.1 Choix du type d'architecture

Pour cela, le paradigme des agents hybrides semble être le plus approprié. En effet, alors que l'objectif lié à la simulation d'un grand nombre d'agents incite à s'intéresser aux agents réactifs, celui lié à la maximisation de la crédibilité des comportements pousse à ajouter des processus cognitifs plus complexes. La coexistence de ces deux objectifs orthogonaux est un premier argument pour le choix d'une architecture hybride, c'est-à-dire pouvant coupler des processus réactifs et cognitifs. Un deuxième argument vient de l'étude des tableaux de synthèse de la partie 2.6, qui montrent que les architectures les plus propices à rassembler les 4 critères sont les architectures hybrides.

3.4.1.2 Présentation et inspirations de notre approche

Une grande partie de l'originalité de notre approche repose sur la séparation en deux de son processus décisionnel. En effet, en amont de la phase de décision proprement dite, nous ajoutons une phase de génération des comportements potentiellement intéressants, et c'est seulement à partir de ces comportements que la décision travaillera. Cette idée vient en partie de l'architecture DAMN [Rosenblatt 1995], qui propose un mécanisme de vote. Dans celui-ci, plusieurs modules simples, chacun en charge d'un seul comportement, attribuent une note à chacune des actions immédiatement réalisables. Cependant ce mécanisme souffre de nombreux problèmes : le contexte est difficile à prendre en compte, sa complexité augmente énormément avec l'augmentation du nombre de comportements adoptables, et aucune planification sur le long terme n'est possible. Une autre source d'inspiration est le modèle BDI [Rao 1995], dans lequel la première phase du processus décisionnel consiste en une génération des options comportementales.

Cependant cette phase est très simple. Il s'agit uniquement de faire un tri parmi les désirs de l'agent, entre ceux qui seront immédiatement traités, et les autres. Par ailleurs, une seule intention est manipulée à la fois. La dernière architecture dont nous nous sommes inspirés sur ce point est l'architecture PECS [Schmidt 2005]. Dans cette architecture, différents modules comportementaux luttent pour prendre le contrôle de l'agent. En fonction du contexte, le module le plus adapté décidera du comportement de l'agent. Le problème principal de cette approche est son caractère *Winner-take-all*, c'est-à-dire l'impossibilité de ces modules de travailler de manière coopérative : un module unique sera toujours en charge, inhibant tous les autres.

Notre idée est de combiner ces différentes approches, en reprenant l'organisation de PECS. Cela en mettant en place un processus de proposition de comportements potentiellement intéressants, dérivé du mécanisme de vote de DAMN afin que la décision prennent en compte l'ensemble des avis des modules, et non un seul. Ensuite, pendant la phase de décision, un mécanisme inspiré de la hiérarchie à libre-flux [Tyrrell 1993a] permettra de combiner ces différents votes afin de choisir le comportement offrant le meilleur compromis entre toutes les propositions.

Ce fonctionnement permet de se positionner par rapport à nos deux premières problématiques :

- Le modèle devient générique, puisqu'il est possible de rajouter des modules comportementaux, pouvant traiter d'aspects différents du comportement des agents, sans augmenter la complexité de l'architecture (le mécanisme de proposition de comportements potentiels reste le même).
- Le processus décisionnel gagne en flexibilité puisque tous les modules comportementaux donnent leur avis, et qu'ils sont tous pris en compte lors de la décision.

En ce qui concerne la problématique liée à la crédibilité des comportements, elle sera traitée dans le chapitre 5 traitant des capacités d'anticipation. Quant à la problématique du passage à l'échelle, elle touche l'ensemble des composants du modèle, elle sera donc discutée à la fois dans la section 3.5.8 (au niveau de l'architecture d'agents), dans la section 4.7 (au niveau décisionnel), et dans la section 5.5 (concernant les capacités d'anticipation).

Ce fonctionnement est très générique, puisque les deux parties de l'architecture ne sont pas couplées, les seules informations devant transiter entre les deux parties étant les **propositions de comportement**. On constate qu'avec une architecture de cette forme, on retrouve une organisation rappelant celle de PECS.

3.4.1.3 Modèle et schéma de principe

Toutes ces caractéristiques donnent le modèle, qui a été appelé **FlexMex** [de Sevin 2012a], et qui se compose d'un point de vue très haut niveau des composants suivants (voir 3.2) :

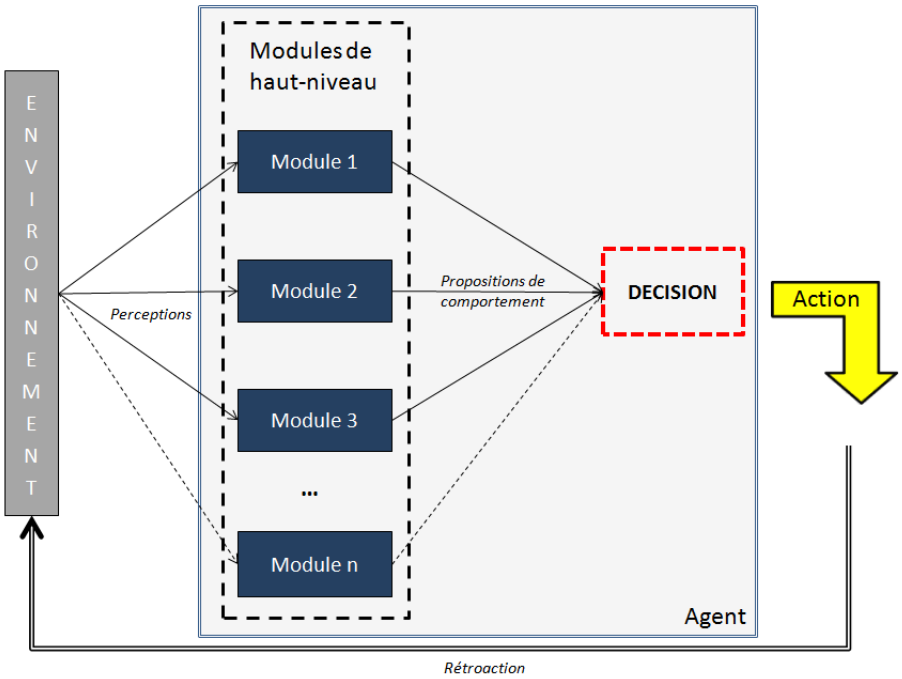

FIGURE 3.2 – Schéma de principe de FlexMex

- **Les modules de comportement** qui génèrent des comportements intéressants (selon leurs propres critères) et qui les envoient sous forme de propositions de comportements potentiels à un module décisionnel chargé de l'arbitrage.

- **Le module décisionnel** qui reçoit les propositions de comportements, qui génère les différents plans permettant de réaliser les comportements proposés, qui choisit la meilleure instanciation de plan, et qui sélectionne les actions immédiatement exécutées.

Comme nous l'avons vu, le déroulement du processus de décision est inspiré du modèle BDI. La première étape est la génération des comportements potentiels, qui est effectuée par les modules de haut-niveau. Du point de vue de l'architecture BDI, ces propositions de comportement peuvent être vues comme les *désirs* de l'agent. Ensuite, le module décisionnel va chercher l'ensemble des plans permettant de réaliser ces désirs, puis à partir des connaissances de l'agent (ou de ses *croyances*), il va chercher l'ensemble des instanciations possibles des plans générés. Ces instanciations de plan sont assimilables à ses *intentions*. Finalement, en fonction des préférences de l'agent, le module décisionnel va sélectionner une ou plusieurs intentions, pour les exécuter immédiatement.

3.4.2 Les modules de comportements

L'architecture présentée ici suppose donc différents **modules de comportements**, en charge de comportements spécifiques, proposés par des spécialistes du domaine en question. Ces modules sont chargés de proposer des comportements à un module décisionnel. Ces modules pourront être actifs, ou inactifs, en fonction de l'individualité des agents considérés. Ainsi il devient possible de modéliser un large spectre d'agents divers, exhibant des comportements hétérogènes, et pouvant s'adapter à des situations ou contextes variés. Lorsqu'ils seront actifs, ces modules participeront à la génération du comportement des agents, en proposant des possibilités de comportements au module chargé du raisonnement. Ils interviennent donc en amont du processus de décision, puisque ce sont eux qui sont chargés d'indiquer les actions envisageables. Pour cette raison, ces modules seront également appelés **modules de haut-niveau**.

Une grande liberté de modélisation est donnée pour ces modules, ce qui permet d'atteindre les objectifs de généricité de l'architecture. La seule contrainte que doit respecter un module de haut-niveau est d'envoyer correctement des propositions de comportements qui circuleront en direction du module décisionnel. La modélisation interne des modules est libre, ce qui permet de choisir n'importe quelle composant ou système et de l'incorporer dans l'architecture. Cette totale liberté de modélisation entraîne cependant un certain nombre de contraintes, notamment au niveau de l'interprétation des informations entrantes (des modèles différents peuvent interpréter différemment une même information). Les deux principales sources de divergence de point de vue sont les perceptions et les effets des actions.

3.4.2.1 Modules de comportement et perceptions

Toutes les perceptions qu'un agent reçoit de son environnement sont fournies à l'ensemble de ses modules de haut-niveau. En effet, il est impossible de savoir a priori quelles perceptions seront intéressantes pour le module, et quelles autres ne le seront pas. C'est donc au module lui-même de faire le tri dans ses perceptions de l'agent. En plus de s'intéresser à des stimuli différents, par ailleurs, les modules ne vont pas interpréter un même stimulus de la même manière : chaque module s'intéresse à des particularités différentes du monde.

Exemple : Une odeur de madeleine sera interprétée différemment par un module gérant la faim (par exemple en augmentant de manière épisodique la faim de l'agent), et par un module gérant les émotions (par exemple en rappelant à l'agent de vieux souvenirs et en le rendant nostalgique).

3.4.2.2 Modules de comportement et effets des actions

Le constat est le même en ce qui concerne les effets des actions. Dans la partie 3.3.4 nous avons parlé des différents effets internes et externes des actions, mais il existe des effets qui ont un impact directement les modules de haut-niveau.

Certains modules peuvent désirer l'accomplissement d'une même action, mais pour des raisons différentes, ou à l'inverse, à partir d'une même raison, désirer l'accomplissement d'actions différentes. Par ailleurs, un module n'a pas à *comprendre* l'ensemble des actions possibles. Certains modules peuvent n'avoir aucun rapport avec certains comportements (en particulier si les modules traitent de comportement bien spécifiques) et donc ne pas être modifiés par l'exécution de certaines actions. Un module de gestion de crise (pour un décideur politique) n'est pertinent qu'en cas de crise, en dehors de ces situations bien précises, il n'offre aucun intérêt ou besoin de l'informer que l'agent prend son café.

Exemple : Soit un agent A, possédant deux modules de comportements, un module gérant les motivations de base (faim, soif, etc.) et un module traitant des émotions. Une même action "manger du chocolat", aura 2 significations complètement orthogonales pour ces deux modules. Pour le premier, cette action impactera la faim de l'agent (elle diminuera ses besoins ultérieurs en nourriture), alors que pour le deuxième cela impacterait la production de sérotonine et l'humeur générale de l'agent (par exemple sa tendance à être joyeux).

On peut compléter l'exemple avec un module gérant l'emploi du temps d'un agent. Pour ce dernier module, l'action "manger du chocolat" n'a aucun impact.

Que ce soit pour les perceptions ou les effets des actions, la solution la plus modulaire et générique de fonctionner est de paramétrer les différents modules grâce à des fichiers de paramètres décrivant la manière dont les modules interprètent les différentes informations entrantes. La structure et le fonctionnement de ces fichiers de paramètres sera décrit dans le chapitre 7.

3.4.3 Propositions de comportements

La généricité de l'architecture provient de sa capacité à prendre en compte n'importe quel module de haut-niveau, quels que soient ses mécanismes internes ou ses domaines de compétences. Les modules de haut-niveau prennent en entrée les stimuli externes et internes et formulent en sortie des propositions de comportements. Ces propositions de comportement sont les seules informations qui sortent des modules de haut niveau, et qui transitent vers le module décisionnel. Elles doivent respecter un formalisme précis qui sera décrit dans la partie 7.4.1. Tant que ce formalisme est respecté, le contenu et les traitements effectués dans les modules de haut-niveau sont libres. D'ailleurs, pour le reste de l'architecture, les modules de haut-niveau sont traités comme des boîtes noires.

Notation : Notons $P(A, t, M) = \{P_1, ..., P_n\}$ l'ensemble des propositions de comportements envoyées par le module M, au temps t, pour l'agent A.

Les propositions de comportements portent sur un comportement donné du graphe comportemental, et possèdent une priorité indiquant l'importance de ce

comportement relative au module qui le propose. Cette priorité appartient à l'intervalle $]-1;1[$, les priorités positives indiquant un comportement d'autant plus important que la priorité est élevée, et les priorités négatives indiquant une répulsion à l'égard du comportement, répulsion d'autant plus forte que la priorité est faible (voir 7.4.1.2 pour plus de détails).

Nous notons $P = (C, p)$ une proposition de comportement, portant sur le comportement C, avec la priorité p.

Les propositions sont d'une complexité quelconque, c'est-à-dire qu'elles peuvent à fois porter sur des comportements très complexes, tout comme sur des actions élémentaires.

Exemple : La proposition de comportement "se nourrir" avec une priorité de 0.8 indique une très forte faim. La proposition de comportement portant sur l'action "travailler" avec une priorité de -0.3 indique une légère répulsion à l'idée de travailler.

3.4.4 Décision

Le module décisionnel est chargé de choisir à quel moment un nouveau processus de génération comportementale doit être lancé. En effet, les modules de haut-niveau n'ont pas les capacités d'appréhender la situation courante de l'agent de manière globale, ils ne peuvent donc pas décider par eux-même du moment où leur avis est nécessaire. C'est le module décisionnel qui est en charge du lancement de la réflexion de l'agent, ainsi les modules de haut-niveau ne font des propositions que lorsque cela est nécessaire. Le **processus décisionnel** de l'architecture englobe l'envoi des propositions de comportement par les modules de haut-niveau, la sélection d'un plan par le module décisionnel, et l'envoi d'un certain nombre d'actions immédiatement exécutées vers un module d'exécution des actions.

Par ailleurs, il est chargé de l'intégration de toutes les propositions de comportement dans un seul et même processus décisionnel. Il doit également être capable de les comparer entre elles, et de planifier un comportement les prenant toutes en compte, en fonction de leur importances. En sortie, ce module envoie une liste d'actions immédiatement exécutées. Il garde également en mémoire le plan actuellement suivi par l'agent.

Le fonctionnement exact du module décisionnel sera détaillé dans le chapitre 4.

3.5 Agent et individualité

Dans notre modèle les agents sont définis par un ensemble de caractéristiques, nous allons ici les détailler, sans nous attarder sur les caractéristiques physiques (âge, taille, apparence, etc.) qui n'impactent pas directement la décision.

3.5.1 Type d'agent

Tout d'abord chaque agent appartient à un **type d'agent**.

Un type d'agent est une abstraction qui permet de regrouper les agents par groupes d'agents qui partagent un certain nombre de paramètres communs. Le regroupement par type, le nombre de types, ainsi que les paramètres servant de points communs sont à la discrétion du modélisateur.

Lorsqu'un agent est créé, son individualité (c'est-à-dire ses préférences), est déterminée en fonction du type auquel il appartient. On peut ainsi assimiler le type à un cadre général, ou un profil typique, autour duquel les agents du type se répartissent. Sans jamais être identiques, ils partagent un même modèle.

Exemple : Soit deux types d'agents "touriste" et "homme d'affaire" dans une simulation urbaine. Les agents du type "touriste" auront plutôt tendance à flâner dans les rues ou visiter la ville, alors que les "hommes d'affaires" risquent plutôt d'être pressés.

Comme l'a démontré [Dujardin 2010a], il est possible de séparer le **raisonnement** de l'**individualité**, c'est-à-dire créer une distinction claire entre la partie du comportement qui décrit les actions que pourra exécuter l'agent (en fonction de ses capacités et de ses connaissances), de la partie qui sélectionne parmi plusieurs actions possibles celles qui correspondent le mieux à ses traits de caractères et sa personnalité.

Nous proposons de rajouter une troisième composante toute aussi distincte et séparable que les deux premières dans le processus décisionnel : le choix des modules comportementaux actifs (c'est-à-dire le choix des mécanismes ou théories de génération de comportements potentiels qui seront utilisés pour modéliser l'agent). Il est possible d'imaginer deux agents ayant les mêmes préférences, la même individualité, mais n'ayant pas les mêmes modules de haut-niveau actifs.

3.5.2 Modules de haut niveau actifs

Ce type va notamment définir les modules de haut-niveau qui seront actifs. Tous les agents d'un même type partageront les mêmes modules comportementaux, ils évoluent dans un même espace comportemental, ce qui n'est pas forcément le cas entre agents de types différents (on appelle espace comportemental l'ensemble des comportements pouvant être sélectionnés).

Exemple : Soit un type "pompier". Les agents de type "pompier" seront dotés de deux modules comportementaux : un module de gestion de motivations très classique, et un module traitant spécifiquement de la gestion des incendies.

3.5.3 Poids des modules

Le type d'agent va également avoir un impact sur le **poids** des modules de haut-niveau. En effet, tous le agents ne vont pas porter le même intérêt à leurs modules actifs. Un agent peut être très sensible à un module et très peu sensible à un autre. En pratique cela revient à attribuer plus ou moins d'importance aux propositions de comportements provenant d'un module. Cette notion d'intérêt relative aux modules est traitée en attribuant un poids à chaque module actif. Ce poids viendra modifier les priorités des propositions de comportement. Les poids sont définis dans l'intervalle [0,1]. Notons qu'un module doté d'un poids nul est un module inactif.

Notation : Nous notons $Pds(A, M)$ le poids du module M, pour l'agent A.

En plus de définir l'espace comportemental des agents, le type définit également la liste des actions exécutables par les agents. Cette liste est forcément un sous-ensemble de l'espace comportemental, et permet d'interdire à un type d'agents certaines actions.

3.5.4 Préférences

Les agents ont également des **préférences** concernant tous les comportements qu'ils peuvent adopter. Ces préférences sont des valeurs numériques, définies dans l'intervalle [0,1]. Elles indiquent les comportements que les agents aiment adopter, et ceux qu'ils n'aiment pas, sans prendre en compte le contexte.

3.5.5 Inventaire

Les agents possèdent un inventaire, qui contient l'ensemble de leurs possessions matérielles. Les objets contenu dans cet inventaire dépendent de la simulation, il n'y a pas de liste d'objets prédéfinie à l'avance. L'inventaire des agents est appelée à être modifiée en cours de simulation, puisque les effets des actions modifient souvent les inventaires des agents.

Lors de leur création, les agents ont déjà des objets en leur possession, cet inventaire initial dépend de leur type, qui indique les objets possédés, et une fourchette indiquant leur nombre potentiel.

Exemple : Soit A un agent du type "touriste". Ce type indique que lors de leur création, les agents possèdent de l'argent liquide (entre 10 et 100 €), un appareil photo (0 ou 1), un plan de la ville (0 ou 1), etc.

3.5.6 Sensibilité aux critères

Lorsque le graphe comportemental a été défini par le modélisateur, un certain nombre de **critères** ont également été spécifiés (voir 3.3.3). Les agents ne sont pas tous sensibles de la même manière face à ces critères. Certains agents peuvent être plus sensibles que d'autres à certains critères, ou avoir une opinion plus ou moins positive des comportements en lien avec ces critères.

Les agents ont donc une valeur de sensibilité pour chaque critère pris en compte dans la simulation. Cette valeur appartient à l'intervalle [-1,1]. Une valeur de -1 indiquant une répulsion totale envers ce critère, une valeur de 1 une attirance totale, et une valeur de 0 indiquant une parfaite indifférence. Le type d'un agent définit une fourchette dans laquelle chaque valeur de sensibilité sera choisie aléatoirement.

Dans nos simulations, seuls deux critères sont pris en compte : le temps et le coût. Un agent sensible au coût mais peu sensible au temps aura tendance à choisir des comportements dont le coût est faible, quitte à ce qu'ils lui prennent plus de temps.

Exemple : Considérons le critère "dangerosité". Un agent sensible à ce critère sera considéré comme peureux, et aura tendance à éviter les comportements considérés comme dangereux, alors qu'un agent très peu sensible à ce critère sera considéré comme téméraire, et aura tendance à les rechercher.

3.5.7 Niveau d'insatisfaction

Il est intéressant de pouvoir indiquer à tout instant le niveau de satisfaction global d'un agent, c'est-à-dire son contentement moyen. Étant donné le fonctionnement de l'architecture, à savoir des modules de haut-niveau spécialistes qui envoient des propositions de comportement "important" à un module décisionnel, nous choisissons comme définition de l'insatisfaction "le fait pour quelqu'un de ne pas voir satisfait un souhait, un désir" [1]. Ce qui veut dire que l'insatisfaction peut être assimilée à un état de frustration (lié au fait qu'un désir n'est pas réalisé). Ainsi, plutôt que de chercher à caractériser la satisfaction d'un individu, nous préférerons nous intéresser à son **niveau d'insatisfaction**, calculé à partir de l'insatisfaction de ses modules de haut-niveau actifs. En effet, ce sont eux qui sont à l'origine des comportements. L'origine d'une proposition de comportement est toujours une insatisfaction du module de haut-niveau. Si un module est entièrement satisfait de l'état de l'environnement et de l'agent, il n'aura aucune raison de proposer des comportements (ou alors les comportements proposés auront des priorités faibles). De la même manière, un module de haut-niveau fortement insatisfait proposera de nombreux comportements, avec des priorités élevées. L'insatisfaction d'un agent peut donc être estimée à partir de la somme des priorités des comportements proposés par l'ensemble des modules de haut-niveau actifs pour un agent [2].

1. www.larousse.fr
2. Cette approche de la satisfaction rappelle la vision stoïcienne ou bouddhiste du bonheur, nous sommes pleinement conscient que cette vision n'est pas la seule envisageable, mais elle se

Notation : Soit $P(A,t,M) = \{P_1,...,P_n\}$ la liste des propositions de comportements venant du module M, au temps t, pour l'agent A, avec n le nombre total de propositions de comportement. A partir de cette liste il est possible de calculer le niveau d'insatisfaction relatif au module M, de l'agent A, au temps t $I(A,t,M) = \sum_{i=1}^{n} p_i$, avec p_i la priorité de la proposition de comportement P_i.

Partant du principe que l'insatisfaction est une frustration liée à l'attente d'un désir, plus le désir sera important et plus la frustration sera importante. Les priorités des propositions de comportement sont donc à prendre en compte. Mais les poids des modules de haut-niveau ont également leur importance. En effet, si un module a une grande importance pour un agent, le moindre de ses désirs a une forte importance, même si les priorités associées sont faibles. Le calcul du niveau d'insatisfaction globale de l'agent est donc :

Notation : $I(A,t) = \sum_{i=1}^{n} Pds(A,M_i).I(A,t,M_i)$, avec $Pds(A,M_i)$ le poids du module M_i pour l'agent A, et $I(A,t,M_i)$ le niveau d'insatisfaction relatif au module M_i au temps t pour l'agent A.

Nous soulignons le fait que d'autres approches de la satisfaction des agents sont possibles. Le mécanisme d'anticipation qui sera présenté dans le chapitre 5 nécessite l'existence d'une méthode de calcul de la satisfaction instantanée d'un agent. Ce mécanisme est générique, il peut être utilisé dans d'autres types d'architectures utilisant d'autres méthodes de calcul de cette satisfaction.

3.5.8 Importance des agents

3.5.8.1 Introduction

L'architecture que nous proposons doit également pouvoir passer à l'échelle, c'est-à-dire pouvoir simuler un grand nombre d'agents en parallèle. En effet, si pour certaines simulations, un petit nombre d'agent peut suffire, pour d'autres en revanche il peut être important de simuler un très grand nombre d'agents. Si les ressources computationnelles ne sont pas suffisantes, il faut trouver un moyen d'économiser du temps de calcul afin de pouvoir simuler tous les agents sans ralentir la simulation.

Pour cela nous allons nous servir des travaux réalisés en intelligence artificielle portant sur le niveau de détail [Wissner 2010], grâce à un parallèle réalisé avec le domaine de la représentation graphique. En effet, depuis longtemps, le domaine du rendu graphique en 3 dimensions s'intéresse à la question d'un **niveau de détail dynamique**. Le principe de base étant que si un objet est situé loin de l'observateur,

prête particulièrement bien au modèle, et permet de caractériser simplement l'état de bonheur global d'un agent.

sa qualité graphique (i.e. la plupart du temps, le nombre de ses polygones) peut être diminuée. Ainsi l'objet sera plus rapide et moins coûteux en temps de calcul à afficher à l'écran, et la perte de qualité de sa modélisation est compensée par le fait que, étant situé loin de l'observateur, l'œil ne peut pas distinguer autant de détails. De plus, l'économie réalisée en simplifiant l'arrière-plan peut permettre d'augmenter la qualité du premier plan, ce qui est visuellement très appréciable.

Un raisonnement similaire peut être effectué en intelligence artificielle, et dans la modélisation agent tout particulièrement, avec deux complexités supplémentaires. A savoir : la distance à l'observateur (qui n'est pas forcément suffisante à caractériser l'importance d'un agent), et le fait que l'observateur se déplace, mais également les agents. La première étape quand on travaille sur un niveau de détail en intelligence artificielle, est donc de définir la notion d'**importance des agents**.

L'importance d'un agent peut dépendre de deux choses : son rôle et sa situation. La notion de rôle est donc une nouveauté par rapport au niveau de détail en représentation graphique, qui ne s'intéresse qu'à la situation des objets.

3.5.8.2 Rôle

Nous proposons la notion de rôle d'un agent, défini dans le scénario qui régit la simulation. En fonction de ce qui est simulé, un certain nombre d'agents peuvent avoir une importance particulière dans les événements. Par exemple si le but de la simulation est d'observer un scénario d'évacuation de bâtiment dans le cas d'une panique, les agents de sécurité en charge de l'évacuation vont avoir une importance beaucoup plus grande que les civils constituant la foule.

Il est très facile de faire un parallèle avec le cinéma, où, en fonction du scénario d'un film, il est possible de faire la distinction entre les acteurs principaux, et les figurants. Lors d'une scène, les agents importants (les acteurs principaux) sont très importants, ils doivent concentrer une grande partie des ressources de calcul afin de simuler leur comportement. Par opposition les agents peu importants (les figurants), peuvent avoir un comportement très basique, très simplifié, demandant moins de ressources, car l'attention de l'observateur n'est pas focalisée sur eux.

Cependant le rôle d'un agent n'est pas absolu, il est relatif au déroulement de la simulation. Imaginons par exemple un scénario visant à simuler des policiers, répondant à des situations de gestion de crise en cas d'attaque terroriste. Les policiers et les terroristes de la simulation ont un rôle important, et tous les habitants du quartier ont un rôle, a priori, non important. Seulement si un terroriste décide de prendre quelqu'un en otage, cet otage qui avait jusqu'alors un rôle négligeable devient très important. Son rôle doit s'adapter à sa nouvelle importance.

3.5.8.3 Situation

Quel que soit son rôle, la situation d'un agent modifie également fortement son importance. En effet, si un agent est au premier plan devant la caméra, il est, de fait, plus important que s'il était complètement invisible. La notion de situation nous

amène à discuter de la notion de **zone d'intérêt**, pour laquelle nous reprenons les travaux de [Navarro 2013a]. La zone d'intérêt la plus évidente est celle visible par l'observateur. En effet, si un observateur regarde une zone, c'est qu'elle est importante, les agents qui s'y trouvent gagnent donc en importance. Mais la position de la caméra n'est pas suffisante à décrire les zones importantes. En effet, dans une simulation il peut être intéressant de considérer certaines zones comme importantes, même si elles ne sont pas tout le temps observées, afin que les comportements des agents dans cette zone soient toujours modélisés avec soin. Par exemple, dans le cas d'une simulation d'évacuation de bâtiment, si l'environnement est plus vaste que le bâtiment à évacuer, le bâtiment doit être une zone importante. Dans le cas de l'évacuation d'un immeuble à cause d'un feu, l'immeuble et ses alentours sont intéressants à considérer comme une zone d'intérêt. Par contre le reste du quartier est moins *intéressant*.

3.5.8.4 Définition de l'importance

L'importance d'un agent dépend donc de son positionnement par rapport aux deux axes de son rôle et de sa situation. En fonction de son importance, il est possible de diminuer la précision de sa modélisation, afin de rendre son comportement plus grossier, plus simpliste, moins intéressant, mais plus rapide à produire.

Il est important qu'un agent dont le rôle est important soit tout le temps modélisé avec soin, même s'il n'est situé dans aucune zone d'intérêt. Par contre un agent dont le rôle n'est pas important, qui se trouve loin de toutes les zones d'intérêt, peut devenir extrêmement simple.

De nombreuses catégorisations de l'importance sont possibles, car elles dépendent du type de simulation visé, du nombre d'agents à simuler, des ressources disponibles, etc. Nous allons cependant donner un exemple.

Exemple : Lors de la définition du scénario, les agents peuvent avoir deux sortes de rôles : un rôle important ou non-important. Les zones d'intérêt sont définies par un point (le centre de la zone), tous les agents situés à moins de 20 mètres de ce point sont considérés comme étant dans la zone. Le point situé à la verticale de la caméra est considéré comme le centre d'une zone d'intérêt. A un instant donné de la simulation les agents peuvent appartenir à l'une des catégories d'importance suivantes :

- Catégorie 1 : Les agents dont le rôle est important.
- Catégorie 2 : Les agents dont le rôle est non-important, mais situés dans une zone d'intérêt.
- Catégorie 3 : Les agents dont le rôle est non-important, et situés hors de toutes zone d'intérêt.

3.5.8.5 Prise en compte de l'importance

Une fois que les catégories ont été définies, il y a plusieurs manières complémentaires de simplifier la modélisation des agents comme nous le verrons dans les parties 4.7 et 5.5. Cependant la structure même de notre architecture nous permet de prendre en compte ce niveau de détail d'une manière originale : puisque les agents sont constitués d'un ensemble de modules comportementaux, il est possible de réduire le nombre et/ou la complexité de ces modules afin d'adapter la puissance de calcul nécessaire aux besoins de la simulation.

3.6 Architecture réactive de base

3.6.1 Introduction

Afin de tester et valider le fonctionnement de la méta-architecture générique présentée précédemment, nous avons proposé un certain nombre de modules de comportements afin d'obtenir une architecture d'agents utilisable. Afin de limiter la complexité de cette architecture de base, nous avons choisi uniquement des modules réactifs, dont le but est avant tout d'assurer l'autonomie des agents, ainsi qu'un minimum de cohérence et de crédibilité.

Le premier module sélectionné est un module de gestion des motivations, inspiré par les différentes architectures motivationnelles abordées dans l'état de l'art ([de Sevin 2006, Dujardin 2010a]). Ce choix a deux raisons principales. Tout d'abord les travaux de [Andriamasinoro 2003] montrent que les sources des comportements humains peuvent être modélisées sous forme de motivations à partir de la pyramide des besoins de Maslow [Maslow 1943]. D'autre part, les architectures motivationnelles permettent de donner une grande autonomie aux agents.

Le deuxième module est un module d'instinct, basé sur des règles de perceptions-actions du type de celles couramment retrouvées dans les architectures de l'état de l'art, notamment les modèles basés sur de la planification [Fikes 1972] ou des systèmes de classeurs [Donnart 1996].

Ces deux modules suffisent, a priori, a donner suffisamment d'autonomie aux agents. Mais afin de garder le contrôle de leurs emplois du temps, sans pour autant tomber dans des comportements scriptés, nous rajoutons un module gérant l'emploi du temps des agents.

3.6.2 Module motivationnel

Le premier module utilisé dans l'architecture est un module motivationnel, basé sur l'approche éthologique [McFarland 1993, Blumberg 1996]. Le choix de ce premier module vient du fort intérêt des systèmes motivationnels, à tel point que de nombreuses architectures d'agents virtuels n'utilisent que ce paradigme afin de faire agir leurs agents (voir 2.2.2.5). Cette approche possède de nombreux avantages, notamment l'autonomie des agents (grâce à l'utilisation de variables homéostatiques), une grande diversité de comportement, ainsi qu'une importante

modularité (en raison de la capacité à ajouter ou enlever simplement des motivations). Les éthologistes défendent également que l'approche motivationnelle est capable d'offrir une grande crédibilité de comportements, grâce à l'utilisation de variables internes, pouvant représenter des quantités réelles (par exemple la quantité de sang dans le corps, la quantité d'eau, etc.). Évidemment ces modèles ont été élaborés pour simuler le comportements d'animaux, mais ils offrent une excellente base pour la simulation de comportements humains.

Ce modèle est inspiré des travaux de [de Sevin 2006]. Il possède 2 types de variables liées entre elles : les variables internes et les motivations.

3.6.2.1 Variables internes

Les variables internes représentent l'état interne homéostatique de l'agent virtuel et évoluent en fonction de son comportement. Par souci de simplicité, ces variables ont une valeur comprise entre 0 et 1. Une variable interne égale à 0 indique une absence totale de désir, alors qu'une variable interne égale à 1 indique un désir capital. Elles sont modifiées par les effets des actions (par exemple l'action "manger" entraîne une diminution de la variable interne liée à la "faim"), et évoluent naturellement au cours du temps (avec le temps, la variable interne de faim d'un agent aura tendance à augmenter). Ces variables évoluent dans 3 zones distinctes : une zone de confort, une zone de tolérance, et une zone de danger. Lorsqu'une variable interne est dans la **zone de confort**, le besoin associé à cette variable est considéré comme satisfait. Si elle augmente, elle passera dans la **zone de tolérance**. A ce stade le besoin peut être toléré, mais il est présent. Si la variable continue à augmenter, elle passera dans la **zone de danger**. A ce stade le besoin devient non seulement primordial, mais dangereux s'il n'est pas satisfait dans les plus brefs délais. L'objectif principal du module motivationnel est donc de maintenir aussi souvent que possible l'ensemble de ses variables internes dans leurs zones de confort. Ces variables internes sont liées à des motivations.

3.6.2.2 Motivations

Les motivations sont des variables abstraites représentant l'intérêt prêté par le module motivationnel à ce que l'agent adopte un certain type de comportement (par exemple se nourrir). Les motivations dépendent des variables internes qui lui sont associées (la motivation de "faim" dépend de la variable interne de "faim"), et des stimuli internes (par exemple la diminution de la variable interne liée à la "faim" peut entraîner une augmentation de la motivation liée à la "soif"), ou externes (par exemple la perception d'une odeur de croissant peut augmenter la motivation de "faim"). Alors que les variables internes se veulent objectives, les motivations sont subjectives. A une même valeur de variable interne, deux agents peuvent accorder une importance différente au comportement lié, en fonction de leur individualité (gourmand, stressé, etc.), et du contexte. Les motivations sont liées à des buts (par

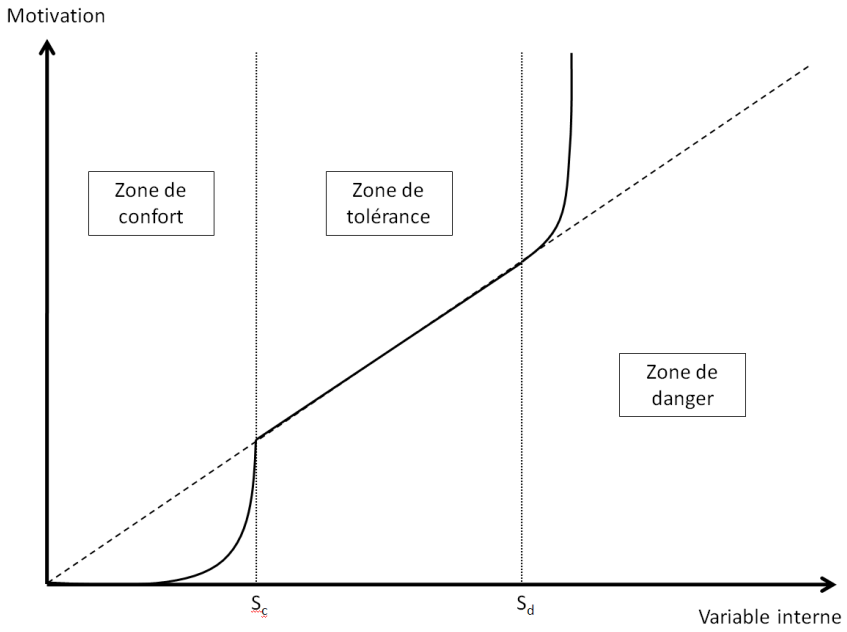

FIGURE 3.3 – Évolution d'une motivation en fonction de la variable interne

exemple "se nourrir" pour la motivation de "faim"), que le module cherchera à faire réaliser par l'agent.

3.6.2.3 Zones des confort, tolérance, et danger

A chaque variable sont associés deux seuils : un seuil de confort S_c en-dessous duquel une variable interne est dans sa zone de confort, et un seuil de danger S_d au-dessus duquel une variable interne passe dans sa zone de danger (voir 3.6.2.3). Dans la zone de confort, la courbe motivationnelle associée a une forme exponentielle. En effet, lorsque la variable interne est très faible, la motivation associée est quasi nulle. Ce n'est que lorsque la variable interne se rapproche de son seuil de confort que la motivation prend de l'importance. Dans la zone de tolérance, la valeur de base de la motivation est égale à celle de la variable interne. Dans la zone de danger, la courbe de la motivation reprend à nouveau une très forte croissance, en raison du danger lié.

Notation : Soit i la valeur de la variable interne, on définit M la valeur de sortie de la variable interne par

$$\begin{cases} M = S_c.e^{i-S_c} & \text{, si } i < S_c \\ M = i & \text{, si } S_c <= i <= S_d \\ M = i/(1-i)^2 & \text{, si } S_d < i \end{cases}$$

Le choix des seuils de confort et de danger doit être réalisé en fonction de la motivation associée, ainsi que de la simulation.

3.6.3 Module d'instinct

Afin de complémenter le module motivationnel, l'architecture utilise également un module d'instinct, très simple, basé sur des règles dites de **perception-action**. Ces règles permettent de coupler directement une perception venant de l'environnement avec une action ou un comportement. Les comportements modélisés rappellent des comportements instinctifs, puisque l'agent n'opère aucun raisonnement avant de proposer ces "réactions", aucun modèle interne n'est nécessaire non plus.

Exemple : Un exemple de réaction instinctive pourrait être :

si perception d'une explosion et **si** distance < 20m	**alors** proposer action "se baisser" avec une priorité de 0.8

Il est également possible d'ajouter des règles couplant une perception interne et un comportement. On pourrait par exemple imaginer une jauge de peur, interne à chaque agent, et lorsque cette jauge de peur atteint un certain niveau, l'agent déclenche un comportement de fuite.

Notons que ces règles de perception-action sont dotées de priorités, qui seront directement utilisées comme priorités de la proposition de comportement associée. Les réactions ne seront donc pas forcément activées lorsque la perception correspondante sera reçue par un agent. Cela dépendra des autres propositions de comportement, du plan en cours, et de l'individualité de l'agent. Notons également que plusieurs actions peuvent être associées à une même perception.

Ce module a plusieurs intérêts immédiats. Premièrement il est extrêmement simple, et permet d'ajouter aux agents des comportements réactifs augmentant considérablement leurs crédibilité. En effet, si un module de motivation permet de rendre les agents fortement autonomes, il ne permet pas de prendre en compte l'ensemble des comportements non-motivationnels, et en particulier les comportements instinctifs, non planifiés, en réponse à des stimuli venant de l'environnement. L'ajout de ces comportements augmente de manière importante l'impression de "vie" de la simulation. Par ailleurs, ce module peut permettre de créer de petits scripts, dictant le comportement de certains agents dans certaines situations précises.

3.6.4 Emploi du temps

Les deux modules précédents sont très souvent employés pour simuler le comportement d'animaux. En effet, les motivations et les comportements instinctifs sont considérés comme étant les principales sources du comportement animal. Cependant, dans le cas d'humains virtuels évoluant dans un environnement urbain, ces modules souffrent d'un problème majeur. L'activité humaine, en particulier celle ciblée par notre projet "la vie dans la ville", est souvent dictée par des contraintes

liées au travail. Ne pas prendre en compte l'activité professionnelle dans les sources de comportement humain limite fortement la crédibilité des simulations. La reproduction des pics d'activités d'une ville (foules de piétons, flux de véhicules, encombrement des voies de circulation, engorgement des transports en commun, etc.) est un élément clef de tout simulateur de ville virtuelle, notamment lorsque l'urbanisme et les transports sont des domaines d'application cible. Or la gestion des activités professionnelles via les deux modules précédents est problématique. En effet, il est difficile de considérer le travail comme une "motivation", dont l'envie ou le besoin évolue au cours du temps, et qu'il faudrait gérer de manière homéostatique. Le travail est plus facile à modéliser lorsqu'il est assimilé à un comportement obligatoire (pas dans le sens "comportement scripté", mais dans le sens "obligation morale"), à horaire fixe.

Chaque agent, en fonction de son type, reçoit un travail, et un emploi du temps correspondant. Cet emploi du temps comporte des horaires de travail, un lieu de travail, et une priorité. Pendant ses horaires de travail, le module d'emploi du temps envoie une proposition de comportement liée à son travail (se rendre à son travail si l'agent ne s'y trouve pas, et travailler s'il s'y trouve) dotée de la priorité correspondante (tout le monde n'apporte pas la même importance à son travail). Encore une fois, ces comportements restent des propositions de comportement, si un comportement plus prioritaire est proposé pendant les heures de travail, il y a de fortes chances pour que l'agent décide d'arrêter de travailler (par exemple si un incendie se déclenche).

Exemple : Soit t l'heure actuelle de la simulation.

$$\left.\begin{array}{l}\textbf{si } 9\text{h} < \text{t} < 12\text{h} \\ \textbf{ou } 13\text{h}30 < \text{t} < 18\text{h}\end{array}\right\} \quad \begin{array}{c}\textbf{alors} \text{ proposer action "travailler"} \\ \text{avec une priorité de } 0.7\end{array}$$

Notons que ce module n'est pas limité aux activités professionnelles. Toute activité dépendant d'un horaire peut être prise en compte. Ainsi, il devient facile d'orienter le comportement des agents, de manière à ce qu'ils respectent (ou reproduisent) les comportements humains réels observés. Par exemple, le module d'emploi du temps peut augmenter la priorité des comportements liés à la restauration (faim), aux horaires classiquement associés aux repas en proposant ces comportements avec une priorité faible. Cette priorité viendra s'ajouter (voir partie 4) aux éventuelles propositions de comportements liées à la faim, venant du module de motivations. En effet, le fait que les humains mangent généralement à des horaires fixes provient de conventions sociales culturelles (les horaires classique des repas peuvent varier selon les cultures, par exemple entre l'Espagne et la Suède). Ce sont des connaissances auxquelles un agent virtuel n'a pas accès. Il est donc nécessaire de leur inculquer des "connaissances", en quelque sorte ad-hoc puisque dépendantes du contexte de la simulation désirée.

3.6.5 Conclusion

Nous venons de présenter les différents modules réactifs incorporés à l'architecture, ce qui donne l'architecture réactive de base. La figure 3.4 montre le schéma de cette architecture.

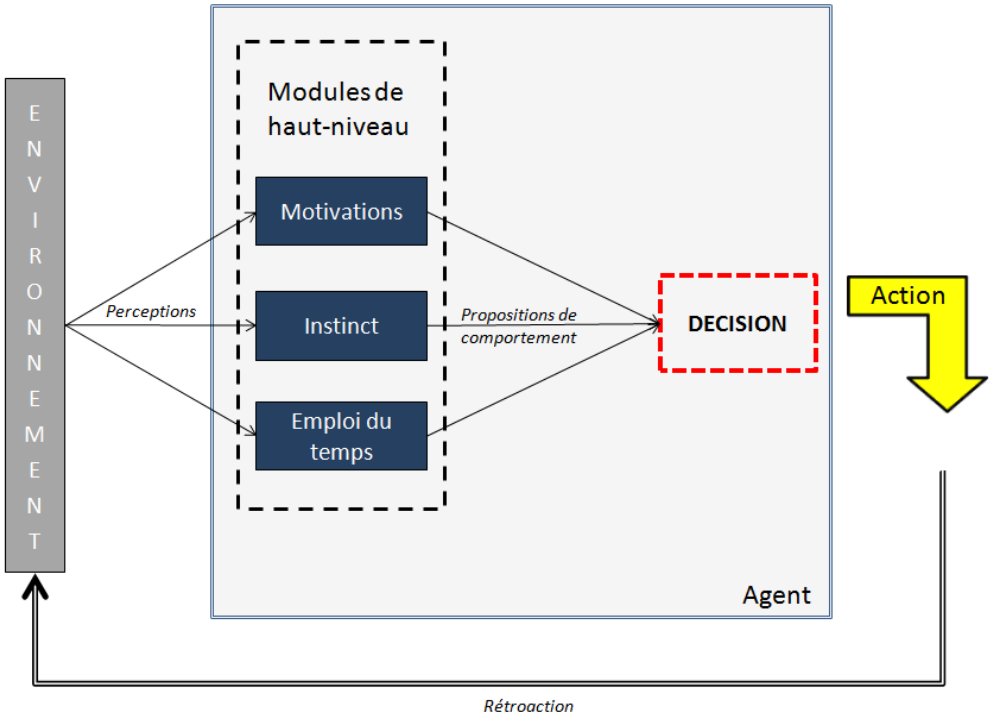

FIGURE 3.4 – Schéma de l'architecture réactive de base

3.7 Autres modèles présents dans l'architecture

Les trois modules présentés précédemment permettent, à eux seuls, d'obtenir des simulations d'une certaine qualité (voir le chapitre sur les évaluations : 6). Ils ont été développés et incorporés à l'architecture par le rédacteur de ce document, c'est-à-dire l'un des concepteurs de l'architecture. Or les mécanismes d'ajout de modules de haut-niveau se veulent génériques. Il est donc nécessaire de vérifier que l'intégration de modules de haut-niveau quelconques, développés de manière externe à l'architecture (sans connaître son fonctionnement), et par un autre concepteur, se déroule correctement. Ainsi, deux autres modules de haut-niveau ont été incorporés à l'architecture. Ces modules ont été développés au cours des thèses réalisées au Laboratoire d'Informatique de Paris 6, dans le cadre du projet Terra Dynamica, par Sabrina Campano et Cyril Poulet, encadrés respectivement par Vincent Corruble et Nicolas Sabouret, et par Vincent Corruble et Amal El Fallah Seghrouchni.

3.7.1 Comportements affectifs

En temps qu'humains, il nous est beaucoup plus facile de prêter des émotions à un humain qu'à un animal, à partir de l'observation de leurs comportements. Afin de gagner en crédibilité, un humain virtuel doit donc être doté d'émotions [Ochs 2009]. C'est pourquoi un module simulant des comportements affectifs [Campano 2012, Campano 2013a] a été ajouté. Ce module a pour objectif de prendre en compte les facteurs humains (typiquement des émotions comme la peur, ou des sentiments comme l'estime de soi) lors de la simulation d'un agent virtuel. La majorité des modèles traitant ces facteurs intègrent des catégories d'émotions qui impactent la prise de décision, c'est-à-dire influencent l'orientation du comportement standard de l'agent. Cependant la détermination de ces catégories et leur impact sur le comportement est une tâche complexe. C'est pourquoi ce modèle fonctionne sans aucune catégorie. La prise en compte des facteurs humains permet d'ajouter des comportements considérés comme affectifs dans le comportement global des agents. Le module est basé sur le modèle de conservation des ressources (les ressources pouvant être psychologiques ou matérielles), considérant les émotions comme des propriétés émergentes.

Exemple : Soit A un agent. Considérons les ressources r_1 = "*réputation*", r_2 = "*ponctualité*" et r_3 = "*santé*". Considérons les actions a_1 = "doubler une personne dans une file d'attente" qui provoque une baisse de la ressource de réputation, et a_2 = "menacer quelqu'un", qui menace la ressource de santé d'une personne. Supposons que A attend dans une file d'attente pour retirer de l'argent. La file est longue et il est pressé. Cet agent pourrait essayer de doubler la personne devant lui (exécuter a_1), s'il considère sa ressource de ponctualité comme plus importante que sa ressource de réputation. Cependant si l'agent devant lui, répond à sa tentative en le menaçant (action a_2), A devra comparer l'importance de sa ressource de santé par rapport à sa ressource de ponctualité.

Un simple classement de l'importance de toutes les ressources existantes dans la simulation suffit à ce module pour proposer à l'agent des comportements. L'objectif de ces comportements est qu'ils soient jugés comme résultant d'une émotion par un observateur externe (si l'agent double c'est qu'il est "excité", "en colère", "sans gêne", mais s'il se rétracte quand il est menacé, il a "peur"), alors que le modèle ne contient aucune variable d'émotion.

Le fonctionnement précis de ce module ne sera pas détaillé ici, plus d'informations peuvent être trouvées dans [Campano 2013b].

3.7.2 Coordination

Si la prise en compte des facteurs humains est un élément important en vue de la simulation d'humains virtuels crédibles, la coopération entre agents en est un autre. En effet, tous les modules présentés précédemment sont centrés sur un seul

agent. Les autres personnages virtuels étant considérés comme des obstacles (pour le déplacement par exemple), ou des gênes (dans des files d'attente), voire n'étant pas du tout pris en considération. De nombreux comportements nécessitent une coordination, ou une coopération, entre agents. Le module de coordination ajouté dans l'architecture [Poulet 2012b] traite de cet aspect, et plus particulièrement de la coordination d'agents réalisant une patrouille [Machado 2003, Almeida 2004]. La patrouille est un cas d'application très intéressant de la simulation urbaine, par exemple dans le cas de catastrophes naturelles (recherche de survivants ou de blessés, déblaiement de zones effondrées etc.), comme pour la *Robocup Rescue*[3]. Ce module nécessite une communication entre agents, un partage d'informations, et une coordination afin d'optimiser les temps de trajet, l'efficacité des participants, ainsi que les fréquences de visite des lieux importants.

Le nombre d'actions pouvant être proposé par ce module est assez faible, puisque seules trois actions sont prises en compte : "entrer dans la patrouille", "sortir de la patrouille", et "se déplacer" (c'est-à-dire patrouiller vers un endroit précis). Toute la complexité de l'approche repose sur le libre-arbitre des agents et la gestion d'autres comportements en parallèle, en dehors de la patrouille. Typiquement afin de se coordonner, il est important d'accéder à une bonne estimation des temps de trajet d'un agent d'un point A vers un point B. Or un agent devant également satisfaire d'autres motivations (faim, soif...) en plus de ses devoirs liés à la patrouille, peut avoir des temps de déplacement variables, sans pour autant que ce dernier ne sorte complètement de son rôle de patrouilleur. L'originalité principale de la thèse repose sur l'étude de ce problème en système ouvert, c'est-à-dire que les agents peuvent entrer et sortir de la patrouille à tout moment.

Aucun détail de fonctionnement ne sera donné ici, vous pouvez vous référer à [Poulet 2012a, Poulet 2013] pour plus d'explications.

3.8 Conclusion

La première contribution de cette thèse est la participation à la conception d'une architecture d'agents générique, autorisant l'intégration de n'importe quel module proposant des comportements, quels que soient son architecture interne ou les modèles utilisés [de Sevin 2012a]. Il en résulte une architecture extrêmement flexible, permettant la modélisation d'un large spectre d'agents, exhibant des comportements variés, pertinents dans un grand nombre de situations, et donc de types de simulations. Cette architecture a un objectif d'intégration et de composition de théories et de domaines divers. Le principe défendu est que les différents domaines s'intéressant à la modélisation d'agents virtuels ont développé des systèmes permettant la reproduction d'aspects différents des comportements (humains ou non). En les incorporant au sein d'une même architecture, il devient possible de modéliser chaque agent en choisissant quelles théories utiliser, en fonction du type d'agent modélisé et de ses objectifs.

3. http ://www.robocuprescue.org/

Dans ce chapitre nous avons expliqué le vocabulaire, les concepts et le formalisme que nous allons manipuler tout au long de notre exposé. Nous avons également présenté l'architecture générique, et son fonctionnement de haut-niveau, ainsi que l'instanciation que nous avons réalisé de la méta-architecture, à savoir une architecture réactive permettant de simuler des agents autonomes.

3.9 Limitations et perspectives

Le modèle présenté précédemment peut être amélioré de multiples manières. La caractérisation des actions reste ainsi assez basique, et pourrait être considérée comme une limite pour des modélisateurs cherchant à simuler des comportements d'une grande complexité. Il est en effet toujours possible de complexifier les actions d'une simulation, de rajouter des caractéristiques, des particularités à prendre en compte. On pourrait par exemple imaginer des actions dont le bénéfice serait non pas personnel, mais global, ou partagé entre un groupe d'agents. On pourrait aussi gérer des actions dont la réalisation devrait passer par une coopération entre agents, et dont la durée dépendrait des capacités de chacun, etc.

Les propositions de comportement sont elles aussi réduites à une expression très simple. Pour l'intégration de certains modèles psychologiques complexes, un formalisme de proposition de comportements plus élaboré serait peut-être nécessaire, par exemple en ajoutant des conditions aux propositions, ou des caractéristiques. On peut par exemple, penser à des propositions de comportements portant sur des actions futures, ou dépendant des résultats d'un autre comportement.

De plus, le nombre de modules de haut-niveau développés reste à ce jour assez faible. Dans un but de constitution d'une réelle bibliothèque de modules comportementaux, il serait intéressant de développer et d'incorporer d'autres modules, et de vérifier la cohésion de l'architecture.

Chapitre 4

La composition de comportements

Sommaire

4.1 Introduction

L'architecture présentée dans la partie précédente permet l'intégration de modules de haut-niveau hétérogènes prenant en compte un large spectre de comportements. Cette caractéristique très intéressante du point de vue architectural introduit une complexité supplémentaire dans le mécanisme de sélection de comportements par rapport à une architecture décisionnelle plus classique. En effet, la prise en compte dans un même processus décisionnel de comportements fortement hétérogènes de complexités diverses oblige à gérer une **composition de comportements**.

Ainsi, le module décisionnel, ne doit pas uniquement sélectionner les actions exécutées par l'agent. Il va au préalable devoir intégrer des propositions de comportement hétérogènes, afin de les comparer au sein d'un seul et même processus

décisionnel, décomposer ces comportements en suite d'actions élémentaires, puis éventuellement les coupler afin de sélectionner un comportement de compromis prenant en compte tout ou partie des propositions de comportements (et donc de l'ensemble des sources comportementales de l'agent). Le module décisionnel va également planifier le comportement de l'agent en choisissant dans l'environnement les lieux (ou services) lui permettant d'accomplir ses buts de la manière la plus pertinente possible (voir schéma 4.1).

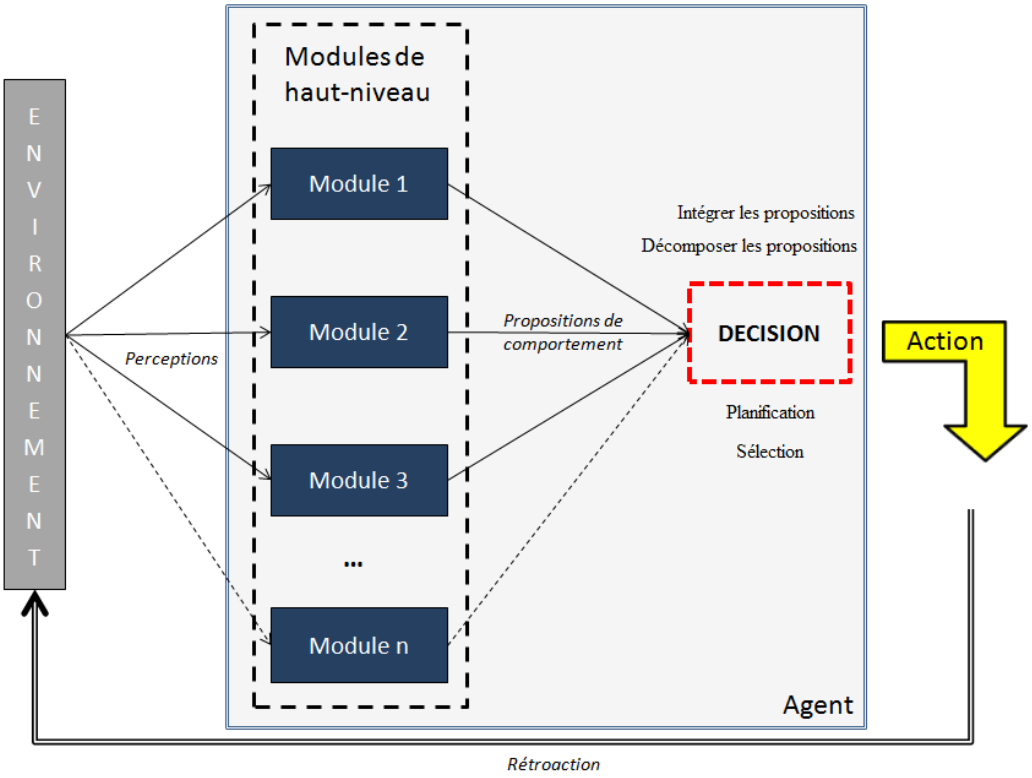

FIGURE 4.1 – Schéma de l'architecture, centré sur le module décisionnel

Le module décisionnel développé dans le cadre de cette thèse, permet de gérer le problème de la composition de comportements hétérogènes (de part leurs origines, leurs complexités, leurs buts, leurs motivations, etc.). C'est une des contributions propres de ce travail.

4.2 La composition de comportement

4.2.1 Hiérarchie à libre-flux, intérêts et limites

La **Hiérarchie à libre-flux** [Tyrrell 1993a] offre un certain nombre de caractéristiques intéressantes par rapport à ces problématiques :

- Elle facilite l'intégration d'un ensemble de stimuli à divers niveaux de décomposition des comportements.
- Elle permet la décomposition de tous les comportements en actions élémentaires, et la propagation des niveaux d'activations.
- Elle autorise la sélection de l'action au dernier niveau de décomposition, ce qui permet de sélectionner une action de compromis pouvant être utile à plusieurs comportements simultanément, y compris ceux peu prioritaires (pas d'élagage).

Cependant ces caractéristiques ne sont pas suffisantes à répondre à nos besoins, nous allons donc ajouter les caractéristiques suivantes :

- Au lieu d'intégrer uniquement des stimuli aux différents niveaux de la décomposition, nous allons intégrer des comportements. Ils pourront être hétérogènes et de complexités diverses.
- Plutôt que de nous restreindre à un mécanisme de sélection d'actions, nous allons ajouter des capacités de planification au système.
- Nous allons également enrichir le graphe comportemental en ajoutant des relations entre comportements (et/ou, antécédence, etc.).

Ainsi, en partant d'un modèle de décision classique, nous obtenons une architecture décisionnelle entièrement nouvelle.

4.2.2 Modification de la hiérarchie à libre flux

Les concepts développés par la hiérarchie à libre-flux vont donc servir de base à un nouveau mécanisme de sélection du comportement. Dans une hiérarchie à libre-flux, les différents réseaux d'activation sont générés lors du lancement de la simulation. Dans notre architecture, les sorties des modules de haut-niveau ne peuvent pas être directement intégrées dans ces réseaux puisqu'elles ne sont pas connues a priori (du point de vue du module décisionnel, les modules de haut-niveau sont des boîtes noires). Il est donc nécessaire de trouver un moyen de les connecter dans les réseaux. Cette intégration se fera via les propositions de haut-niveau, qui seront considérées comme des entrées des réseaux d'activation, au même titre que les stimuli.

Par ailleurs, alors qu'une hiérarchie à libre-flux n'est capable que de sélectionner le comportement le plus pertinent en fonction du contexte, notre processus décisionnel devra être capable de comparer des plans entre eux. Pour cela, nous allons ajouter une étape supplémentaire au processus décisionnel d'une hiérarchie à libre-flux. Après la phase de décomposition des comportements en actions élémentaires et celle de propagation des utilités, nous allons rajouter une phase de construction de plans (voir 4.5), et c'est les différentes instanciations de ces plans qui seront comparées les unes avec les autres.

En ce qui concerne la complexité du graphe comportemental, ce n'est qu'une modification mineure par rapport à une hiérarchie à libre-flux. Les liens entre comportements seront pris en compte lors de la phase de construction de plan.

4.3 Processus décisionnel

Le processus décisionnel est complexe en raison de l'hétérogénéité des comportements proposés en entrée, mais en quoi consiste exactement leur diversité ? Ils sont hétérogènes principalement en raison de la variété de leurs sources. Comme différents modules de haut-niveau peuvent être intégrés dans l'architecture, et que ces modules fonctionnent sur des modèles différents les uns des autres, les comportements en sortie peuvent varier fortement. Ces différences s'expriment sur trois points clefs :

- Leur **importance**. Les modules de haut-niveau n'ont pas connaissance des autres modules actifs, ni de leurs propositions de comportements. Ils n'ont par ailleurs accès qu'à une petite partie du contexte global de l'agent (celle pertinente par rapport à leurs compétences). Ils ne peuvent donc pas moduler les priorités des comportements qu'ils envoient par rapport au contexte. Un module n'ayant pas accès à la dangerosité d'une situation ne baissera pas la priorité de ses propositions si l'agent est en danger de mort par exemple. Comment le module décisionnel doit interpréter les priorités reçues ?
- Leur **complexité**. La complexité d'un comportement est sa hauteur dans le graphe comportemental, c'est-à-dire le nombre maximal de décompositions qui existent entre ce comportement et les actions élémentaires qui le composent. Plus un comportement possède une décomposition importante, plus il est considéré comme complexe. Comment le module de décision doit s'y prendre pour comparer un comportement extrêmement complexe, englobant de nombreux sous-comportements eux-même dépendants d'un certain nombre de sous-objectifs, et une action élémentaire ?
- Leur **objectifs**. Les modules à la source des comportements s'intéressent à des aspects très différents du comportement, et ne prennent en compte que les informations pertinentes par rapport à leur expertise et leur domaine de compétence. Deux comportements proposés par deux modules différents peuvent donc n'avoir strictement aucun rapport l'un avec l'autre. Comment la décision peut coupler des comportements que tout oppose ?

Ces trois problèmes posés par la composition de comportement, seront détaillés dans les trois sections suivantes. L'intégration des comportements traite des différences d'importance des comportements, la décomposition s'occupe des différences de complexité, et la planification se préoccupe des objectifs des comportements.

4.3.1 Intégration des comportements

4.3.1.1 Échelle de valeur

Le premier problème auquel est confronté le module décisionnel est la nécessité d'intégrer dans un même processus décisionnel des comportements hétérogènes. Avant même de détailler plus en avant les comportements proposés par les modules de haut-niveau, les différentes manières de les décomposer, etc. le module décisionnel doit interpréter les priorités reçues.

En effet, en plus de ne pas être attentifs à l'ensemble des éléments constituant le contexte global de l'agent, les modules de haut-niveau ne partagent même pas une même échelle de valeur. Les priorités doivent être comprises entre -1 et 1, avec -1, et 1 respectivement un refus et une obligation d'adopter le comportement, et 0 une indifférence totale. Mais à quoi correspond 0,5 ? Aucune échelle n'indique à quoi correspondent les valeurs intermédiaires, ce qui aurait permis d'avoir un minimum de cohérence au niveau des priorités.

Exemple : Soit deux modules de haut-niveau M_1 et M_2. M_1 étant un module de gestion des motivations, et M_2 un module de gestion des émotions. Si, quelque soit la situation, M_1 envoie des propositions de comportements avec des priorités supérieures à 0.7, et que M_2 ne propose que des comportements dotés de priorités inférieures à 0.3, le comportement de l'agent ne sera pratiquement jamais influencé par ses émotions. A l'inverse, si M_1 envoie toujours des propositions de comportements avec des priorités inférieures à 0.3, et que M_2 ne propose que des comportements dotés de priorités supérieures à 0.7, le comportement de l'agent sera quasiment toujours provoqué par des émotions, et non des motivations.

Pourquoi ne pas instaurer une telle échelle ? Premièrement, rien ne l'interdit. L'architecture étant générique, son premier objectif est de pouvoir s'adapter à toute volonté du modélisateur, en vue de réaliser une simulation bien précise. Il est donc tout-à-fait possible de créer une échelle de valeur à laquelle les modules de haut-niveau peuvent faire référence afin de garder une cohérence globale.

Exemple :

De 0 à 0.2 → priorité négligeable
De 0.2 à 0.4 → priorité faible
De 0.4 à 0.6 → priorité moyenne
De 0.6 à 0.8 → priorité forte
De 0.8 à 1 → priorité absolue

Deuxièmement, cette échelle suppose un minimum de compréhension inter-modules. En effet, cela suppose que tous les modules sont capables de reconnaître une situation ou une action mettant la vie de l'agent en jeu, ce qui modifie fortement

notre objectif initial, qui était de donner une liberté totale de modélisation pour ces modules. Pour contrer ce problème, il est possible d'interdire certaines plages de valeurs à un module. Par exemple un module ne traitant pas de problèmes vitaux ne pourrait pas utiliser les priorités supérieures à 0,9, ou inférieures à -0,9.

Mais il est important que ces mesures restent des *options* de développement des modules de haut-niveau. Car il est impossible de reporter cette arbitrage dans le cœur décisionnel des agents. En effet, le module décisionnel est partagé par tous les agents, il doit être objectif, et surtout ne contenir aucun élément ad-hoc. Or la cohésion des priorités est un problème typiquement subjectif, fortement dépendant des objectifs de la simulation. Par exemple, une simulation peut porter sur le repérage de comportements anormaux ou incohérents (entraînements de type sécurité, ou jeux vidéos). Dans ce cas, la modélisation et la simulation de comportements aberrants sera au cœur même des besoins du modélisateur. Toute régulation des priorités doit donc se faire au niveau des modules proposant les comportements, ou au niveau de l'individualité des agents.

En effet, une partie de l'arbitrage peut également se réaliser à travers les importances relatives de chaque module. Dans la partie 3.5 nous avons évoqué des poids fixes, mais rien n'empêche d'introduire des conditions.

Exemple : Soient un agent A, et deux modules M_1 et M_2. Notons p_{max} la priorité la plus importante envoyée par M_2. Les poids Pds se calculent selon la formule (il n'y a aucun lien entre les poids des deux modules) :

$$Pds(A,t,M_1) = \begin{cases} 0.8 \text{ si } p_{max} < 0.8 \\ 0.3 \text{ sinon} \end{cases} \quad \text{et} \quad Pds(A,t,M_2) = \begin{cases} 0.5 \text{ si } t > 2h \\ 0.7 \text{ sinon} \end{cases}$$

De part la recherche de généricité de l'architecture, il ne faut pas poser d'hypothèses trop contraignantes qui interdiraient certains types de simulations. Cependant le problème reste posé : si les modèles utilisés ne sont pas corrects, et que les agents n'agissent pas de manière cohérente en raison de l'équilibrage des modules de haut-niveau, que faire ? Il faut que le modélisateur soit informé de ce déséquilibre (un système qui corrigerait automatiquement ces erreurs pourrait être contre productif, le modélisateur doit rester maître du fonctionnement des agents. Le modélisateur peut ainsi corriger lui-même les erreurs. Mais il serait cependant intéressant de lui proposer un système automatisé d'équilibrage, à condition que cela reste une option.

4.3.1.2 Une recherche de crédibilité

En prenant le contexte et l'individualité de l'agent en considération, le module décisionnel a la charge de moduler les priorités brutes reçues. Ainsi, la situation courante (distance des lieux intéressants, temps nécessaire pour l'accomplissement d'un comportement, plans et actions en cours, etc.) et les préférences de l'agent seront intégrés dans le processus décisionnel. Les priorités initialement envoyées n'ont

qu'un impact limité sur les priorités finales des différentes options de comportements (les différents plans).

Pour finir, il ne faut pas oublier que le comportement humain n'est pas prévisible. Dans une même situation, deux personnes peuvent prendre deux décisions différentes. De plus, imaginons une situation purement théorique dans laquelle deux personnes rigoureusement identiques seraient appelées à prendre une décision dans l'exacte même situation. Même dans ce cas, leurs choix pourraient différer, en raison par exemple de leur humeur par exemple, ou des derniers événements vécus. En réalité, l'ensemble des paramètres entrant en compte dans un processus décisionnel humain est tellement important que nous n'avons nous-même pas conscience de leur nombre. Un observateur se retrouvant face à un comportement qu'il ne comprend pas, ne le considérera pas forcément comme moins crédible qu'un comportement plus simple à expliquer (voir partie 6.4.3). L'objectif premier de l'architecture est la simulation de comportements crédibles, et non une recherche d'optimalité comportementale.

4.3.2 Décomposition des comportements

4.3.2.1 Propagation des niveaux d'activation

Le deuxième aspect problématique que rencontre le module décisionnel est la différence de complexité des comportements en entrée. Afin de résoudre ces conflits nous réutilisons le principe de décomposition totale, proposé dans la hiérarchie à libre-flux, en les appliquant à ce nouveau problème. Tous les comportements proposés sont entièrement décomposés via le graphe comportemental, jusqu'au niveau des actions élémentaires.

A partir du graphe comportemental complet, les comportement proposés par les modules de haut-niveau sont activés, avec un niveau d'activation proportionnel à la priorité de la proposition. Ensuite, les différents sous-comportements de ces comportements seront activés, puis leurs sous-comportements, jusqu'à ce que l'ensemble des actions élémentaires pouvant intervenir dans un des comportements proposés en entrée soit activé (voir figure 4.2).

Exemple : Dans le cas de la figure 4.2, on peut prendre comme exemple que la proposition P_i propose le comportement C_1 "se nourrir". Ce comportement se décompose en 2 sous-comportements distincts : C_2 "se nourrir chez soi", et C_3 "se nourrir au restaurant". La priorité de C_2 est calculée à partir de celle de C_1, uniquement modulée par les préférences de l'agent (préfère-t-il rester chez lui ou aller au restaurant pour manger ?). Par contre, la priorité de C_3 est également modifiée par la proposition de comportement P_j. La priorité de C_3 prend donc en compte les deux propositions de comportement (la manière dont le système agrège les priorités est expliqué dans la section suivante). Au niveau des actions élémentaires, les priorités des actions A_1 et A_2 sont directement calculées à partir de celle de C_2. Dans notre exemple on peut imaginer que A_1 correspond à "préparer

le repas", et A_2 "manger chez soi". Les priorités de A_1 et A_2 ne sont modifiées que par la proposition P_i. Par contre, l'action A_3 "manger au restaurant", est impactée par les propositions P_i et P_j.

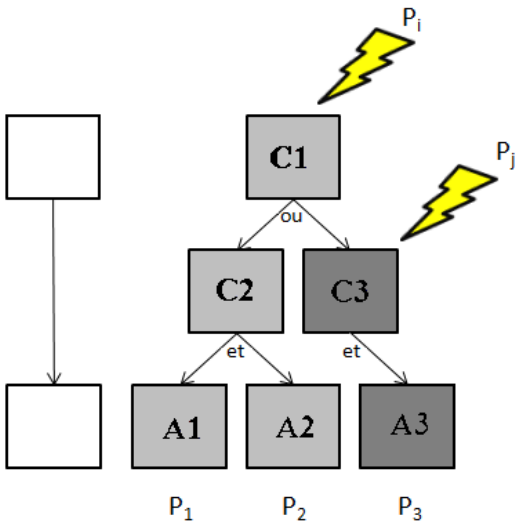

FIGURE 4.2 – Schéma de la propagation des niveaux d'activation

Cette décomposition prend en compte l'individualité de l'agent. En effet (voir partie 3.5), les agents ont des préférences sur leurs modules actifs. Ainsi lorsqu'une proposition de comportement est reçue, sa priorité est immédiatement modifiée par l'importance que l'agent accorde à ce module. De plus, les agents peuvent avoir des préférences pour certains des comportements qu'ils peuvent effectuer. Donc la priorité initiale du comportement proposé sera également impactée par l'intérêt, l'envie, que l'agent a d'accomplir ce comportement. Qui plus est, si ce comportement est décomposable, l'agent a également des préférences sur l'ensemble des décompositions possibles de ce comportement. La priorité de chaque sous-comportement sera déterminée en fonction de la priorité du comportement parent, et par les préférences de l'agent concernant ce sous-comportement. Et ainsi de suite jusqu'au niveau des actions élémentaires.

Ainsi chaque action potentiellement utile pour un ou plusieurs des comportements proposés, possède une priorité prenant en compte l'ensemble des paramètres de la situation courante et de l'agent. Les différences de complexité des comportements en entrée sont donc invisibles, puisque, quels que soient leurs complexités, tous les comportements sont décomposés en actions élémentaires.

4.3.2.2 Agrégation des priorités et des préférences

Lors de la phase de décomposition des propositions de comportement, les niveaux de priorité sont propagés des comportements vers les actions. Lors de chaque

décomposition la priorité du sous-comportement prend en compte plusieurs priorités ou préférences : les priorités des comportements "parents", les préférences de l'agent concernant le nouveau niveau de décomposition, la prise en compte des différents critères, etc.

Il est donc nécessaire de se poser la question de la fonction d'agrégation de ces priorités et préférences.

4.3.2.3 Choix de la fonction d'agrégation

De nombreuses options sont possibles. Dans sa thèse [Dujardin 2010a] propose plusieurs fonctions d'agrégation des évaluations des différentes motivations (dans son modèle, toutes les sources de comportement sont motivationnelles). Les évaluations étant dans $\mathbb{R}^+$, c'est dans cet espace que la fonction sera utilisée. La fonction $\Phi(x)$ retenue est la fonction produit, en raison de la présence d'un élément neutre (x=1), d'un élément absorbant (x=0), de la capacité de la fonction à exprimer l'attraction (x>1) et la répulsion (x<1), et de la simplicité de la fonction. Si cette fonction est effectivement intéressante, il nous semble qu'elle ne respecte pas le principe de "concurrence équilibrée entre comportements" énoncé par Tyrrell (voir 2.3.1), en accordant une trop grande importance à la revalorisation multi-objectifs (un comportement utile à plusieurs modules obtient une priorité beaucoup trop importante). De plus il nous semble intéressant de rester dans un même intervalle borné afin de garder une cohérence dans les priorités, c'est-à-dire garder l'intervalle $]-1;1[$ avec -1 et 1 des priorités absolues (et donc impossible à atteindre). Passer dans $\mathbb{R}^+$ revient à ne plus avoir de priorité absolue, quelle que soit une priorité p_1 il existe une priorité p_2 plus importante, il est donc impossible de s'assurer qu'un comportement sera immédiatement exécuté par un agent, ce qui limite l'intérêt du modèle (puisque pour certaines simulations il peut être utile d'avoir un module de haut-niveau pouvant envoyer des comportement avec une priorité absolue, par exemple des modules de scripts, ou de scénarisation).

Nos critères de sélection de la fonction d'agrégation sont :

- Existence d'un élément neutre.
- Existence d'un élément absorbant.
- Capacité à exprimer l'attraction et la répulsion.
- Simplicité de la fonction.
- Définition dans l'intervalle [-1,1].
- Conservation de l'intervalle de départ.

A partir de ces critères, nous choisissons la fonction **sigmoïde**. C'est une fonction très classique, notamment couramment utilisée dans les réseaux de neurones en temps que fonction d'agrégation.

Définition : Une fonction sigmoïde $S(x)$ est définie par la formule :

$$S(t) = \frac{1}{1 + e^{-\lambda x}}$$

Par rapport à la fonction produit, la fonction sigmoïde possède deux avantages. Premièrement, elle possède des asymptotes dont les équations sont $y = 0$ et $y = 1$. Il est simple de garder la préférence finale dans l'intervalle [0,1], et donc maintenir une cohésion dans l'échelle des préférences. Deuxièmement, la tangente au point d'inflexion ($x = 0$) a une pente de $\lambda/4$. Grâce au paramètre λ il devient donc facile de garder le contrôle du poids de la revalorisation multi objectifs. C'est-à-dire qu'il devient possible de rendre les compromis entre comportements plus ou moins intéressant, en modifiant la valeur de λ.

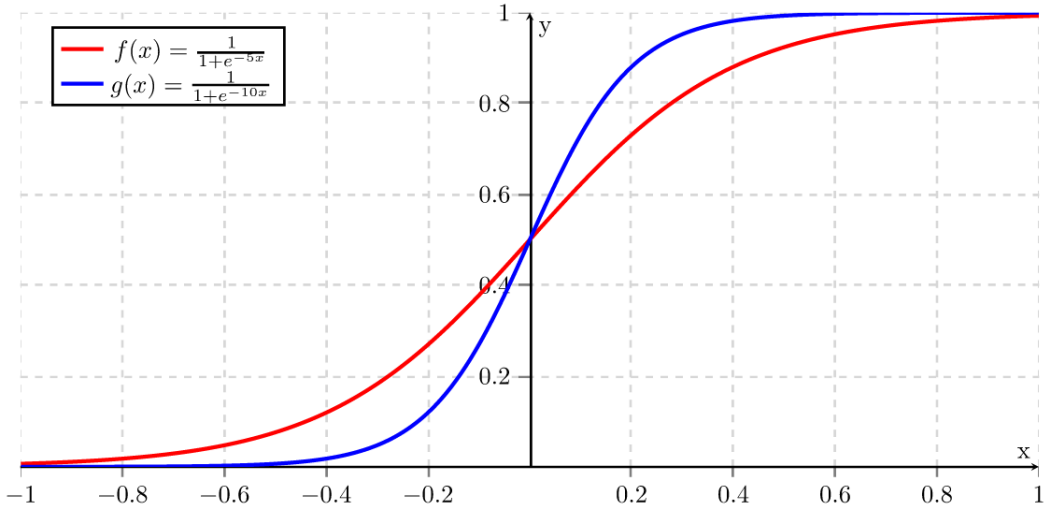

FIGURE 4.3 – Exemples de fonctions sigmoïdes

4.3.2.4 Agrégation

Le mécanisme d'agrégation de préférence est simple. Les différents composants à agréger peuvent être de deux types : soit une priorité d'un des comportements de plus haut niveau (un même comportement peut appartenir à plusieurs comportements de plus haut niveau), soit la préférence du comportement. Voici le calcul permettant de réaliser l'agrégation des préférences :

Soient $p_1, ..., p_n$ les n priorités des comportements de plus haut niveau dont le comportement fait partie, avec $p_1 \leq p_i$ pour tout i, et P la préférence du comportement.

On calcule $x_{initial}$, tel que $S(x_{initial}) = p_1$

$x_{initial}$ correspond donc à l'abscisse de la plus forte priorité reçue.

Ensuite on calcule x_{final} tel que $x_final = x_{initial} + \sum_{i=2}^{n} p_i + P$

On modifie cette abscisse en lui ajoutant toutes les priorités. Ainsi, chaque priorité positive augmentera la priorité du comportement, et chaque priorité négative la diminuera.

Une fois que l'abscisse finale a été calculée, on obtient la priorité finale grâce au calcul :

La priorité du comportement agrégeant l'ensemble des préférences et priorités est : $p_{final} = S(x_{final})$

De cette manière toutes les priorités positives viendront s'ajouter à la priorité initiale (p_1) et indiquer une attraction, alors que les priorités négatives seront retranchées, et indiqueront une répulsion. Quoi qu'il en soit, la priorité finale restera comprise entre 0 et 1.

4.4 Caractéristiques du processus décisionnel

4.4.1 Fréquence de replanification du processus décisionnel

La fréquence de replanification est une caractéristique importante du processus décisionnel. En effet, une fréquence trop importante est coûteuse en temps de calcul, et peut être contre-productive, par exemple si l'agent change d'avis trop souvent. A l'inverse, une fréquence trop faible limite la réactivité de l'agent et le rend trop rigide dans ses choix. Dans notre modèle, cette fréquence est paramétrable et dépend du type de l'agent (voir partie 3.5). Elle peut ainsi être adaptée en fonction des besoins de l'application, ou des ressources disponibles. Cependant elle est positionnée par défaut à une valeur de 1 seconde, car c'est l'ordre de grandeur du temps de réaction humain. Ainsi, lorsqu'un événement se produit, les agents y réagissent (ou choisissent de ne pas y réagir) avec un temps de réaction proche de celui d'un humain réel.

4.4.2 Utilisation de la sémantique de l'environnement

Une des particularités du projet, est l'utilisation d'une sémantique de l'environnement [Harkouken-Saiah 2012]. Cette couche d'informations modifie forcément la manière dont les agents font leur choix. La sémantique de l'environnement est utilisée de trois manières différentes par la décision.

- **Interrogative**. C'est l'utilisation la plus courante. Le module décisionnel demande à la sémantique quels lieux peuvent être utiles à la réalisation d'un certain comportement. Ces lieux sont triés par ordre de pertinence, en fonction de l'individualité de l'agent. Ainsi, la décision peut ne traiter que les options les plus intéressantes, sans perdre de temps à envisager des possibilités peu intéressantes.

- **Réactive**. Lorsqu'un agent ne sait pas comment utiliser un service de l'environnement, il peut faire appel à la sémantique de l'environnement de manière réactive.

- **Pro active**. La sémantique de l'environnement peut également proposer d'elle-même des informations aux agents. Pour ce faire, il faut que ces agents s'inscrivent comme intéressés par un certain type de service. Quand un service correspondant sera détecté dans son environnement proche, une notification lui sera envoyée.

4.4.3 Critères

Les critères présentés dans la partie 3.3.3 prennent toute leur importance au niveau décisionnel. Les critères qui auront été jugés pertinents pour caractériser les actions du modèle sont pris en compte par le module décisionnel afin de préciser les choix de comportement.

Par défaut seuls deux critères objectifs sont utilisés : le coût monétaire et le coût temporel. Mais dans le processus décisionnel intervient également l'aversion que l'agent a envers les critères utilisés (voir 3.5), de même que ses préférences (qualité subjective des actions). Le contexte est également pris en compte.

Exemple : Soit A un agent cherchant à réaliser le comportement "se nourrir". Ses préférences peuvent pencher plutôt pour l'action "manger au restaurant" ou plutôt vers l'action "manger chez soi", mais les critères rentrent également en compte. Ainsi le coût respectif de chaque plan, ainsi que leur durée viennent modifier les priorités (a-t-il suffisamment d'argent pour aller au restaurant ? A-t-il le temps de préparer le repas ? A-t-il le temps de manger dans un bon restaurant ou doit-il se contenter d'une restauration rapide ?). De plus l'aversion de l'agent par rapport à ces critères joue un rôle important (est-ce que l'agent a besoin de faire des économies ? Est-il dépensier ? Aime-t-il prendre son temps ?).

4.5 Algorithme décisionnel

Le processus décisionnel est abordé dans les parties précédentes, nous allons présenter ici l'algorithme de principe de ce fonctionnement.

```
Données :
Soit A un agent;
Soit Pr l'ensemble des propositions de comportements reçue;
tant que simulation en cours faire
    si tempsDeReflexion() alors
        propagationPriorités(A, Pr);
        construirePlan(A, Pr);
        pour chaque Plan Pl faire
            calculPreferencePlan(A, Pl);
            pour chaque Action Act dans Pl faire
                interrogerEnvironnement(A, act);
            fin
            calculerInstanciationsPlan(Pl);
            pour chaque Instanciation I dans Pl faire
                pour chaque Critere C faire
                    NoterInstanciation(I,C);
                fin
            fin
            choisirMeilleureInstanciation(A, Pl);
        fin
        ChoisirNouveauPlan(A);
    fin
fin
```

Algorithme 2 : Algorithme de sélection de plan

avec

- **tempsDeReflexion** : une méthode qui retourne *vrai* si l'agent doit lancer un processus décisionnel lors du pas de temps courant (voir 4.4.1).
- **propagationPriorités** : une méthode qui propage l'ensemble des priorités de toutes les propositions de comportement reçues, en agrégeant à chaque étape de la décomposition les différentes sources tout en prenant en compte l'individualité de l'agent. Voir partie 4.3.2.4.
- **construirePlan** : une méthode qui construit l'ensemble des plans Pl permettant d'accomplir l'ensemble des propositions de comportement Pr. Pour cela il est nécessaire de décomposer les comportements proposés grâce au graphe comportemental de toutes les manières possibles, tout en prenant en compte les éventuels liens entre actions.
- **calculPreferencePlan** : une méthode qui calcule une préférence moyenne du plan à partir de l'ensemble des préférences des actions qui le composent.
- **interrogerEnvironnement** : une méthode qui utilise la sémantique de l'environnement (mode interrogatif) afin de récupérer les objets de l'environnement les plus pertinents pour réaliser le comportement désiré. Cette méthode

retourne les X objets les plus pertinents (X étant un paramètre modulable, en fonction du type d'agent), classés par pertinence, en fonction de critères tels que le coût (prix de l'action), la distance, la difficulté, etc. (plus de détails dans [Harkouken-Saiah 2012]). Lors de la demande à l'environnement, la décision communique des informations telles que les préférences de l'agent, afin que celui-ci puisse les prendre en compte dans ses calculs de pertinence. Ce pré-traitement est très intéressant car il permet à la décision de n'envisager que les possibilités les plus pertinentes, au lieu de devoir les envisager toutes.

- **calculerInstanciationsPlan** : cette méthode construit la totalité des instanciations de plan possibles à partir des objets retournés par la méthode précédente. Comme les instanciations de plans sont constituées d'un ensemble d'actions concrètes, il est important de limiter le nombre de possibilités envisagées (X) afin de maîtriser l'explosion combinatoire (d'où l'importance d'étudier des objets/actions concrètes pertinentes).
- **NoterInstanciation** : cette méthode attribue une note à chaque instanciation de plan. Cette note prend en compte divers paramètres, notamment les critères (voir partie 3.3.3) modulés par l'importance accordée à ces critères par l'agent.
- **choisirMeilleureInstanciation** : cette méthode sélectionne l'instanciation de plan possédant la meilleure note. Toutes les autres instanciations ne sont plus considérées.
- **ChoisirNouveauPlan** : le nouveau plan de l'agent est celui ayant la meilleure instanciation. Un bonus d'inertie est accordé au plan en cours, ce qui permet de prendre en compte les coûts de changement de comportement, et de limiter les problèmes d'oscillation comportementale.

4.6 Exemple

Afin de clarifier les choses, nous allons donner un exemple simplifié d'un déroulement du processus décisionnel.

Exemple : Soit A un agent, au temps t d'une simulation. La figure 4.4 montre le graphe décisionnel simplifié de l'agent, et les deux seules propositions de comportement reçues au temps t : "se nourrir" et "faire du shopping". La première proposition est un comportement pouvant se réaliser de deux manières différentes, soit l'agent peut manger dans un restaurant, soit il peut manger dans un fast-food. La seconde proposition est directement une action élémentaire : "faire du shopping".

La première phase du processus décisionnel consiste à décomposer les propositions reçues et à propager les priorités des comportements jusqu'aux actions élémentaires. Ici, la priorité P_1 de la première proposition est décomposée vers les actions "manger au restaurant", et "manger dans un "fast-food" en prenant en compte les préférences de l'agent pour ces deux actions, ce qui aboutit respectivement aux priorité P_R et P_F. De la même manière, la préférence de l'agent

pour l'action "faire du shopping" vient transformer P_2, la priorité de la seconde proposition, en P_C.

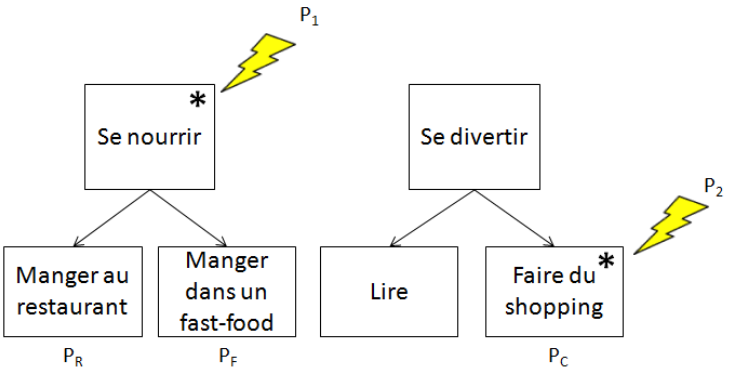

FIGURE 4.4 – Représentation d'un graphe décisionnel simplifié, et des deux propositions de comportement entrantes

Afin de réaliser ces deux propositions, l'agent a 4 plans différents. Il peut

- Soit "manger au restaurant", puis "faire du shopping" (plan 1).
- Soit "manger dans un fast-food", puis "faire du shopping" (plan 2).
- Soit "faire du shopping", puis "manger au restaurant" (plan 3).
- Soit "faire du shopping", puis "manger dans un fast-food" (plan 4).

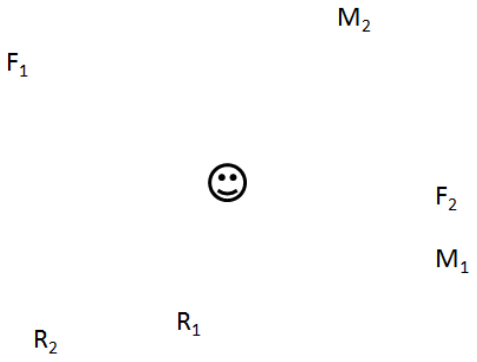

FIGURE 4.5 – Représentation schématique de l'agent et disposition des différents points d'intérêt

Afin d'instancier ces 4 plans, supposons que l'agent connaisse 2 lieux lui permettant de réaliser chacune de ces 3 actions (2 restaurants R_1 et R_2 ; 2 fast-food F_1 et F_2 ; et 2 magasins M_1 et M_2), ce qui nous donne 4 instanciations pour chacun

des plans, et donc 16 possibilités à comparer les unes aux autres. La disposition des lieux et de l'agent est indiquée dans la figure 4.5.

Regardons ce qu'il se passe pour le premier plan. Les 4 instanciations du premier plan sont :

- L'agent peut décider d'aller d'abord à R_1 puis à M_1 (instance 1).
- L'agent peut décider d'aller d'abord à R_1 puis à M_2 (instance 2).
- L'agent peut décider d'aller d'abord à R_2 puis à M_1 (instance 3).
- L'agent peut décider d'aller d'abord à R_2 puis à M_2 (instance 4).

Pour chacune de ces instances, le module décisionnel va calculer une note, par rapport aux différents critères des actions (uniquement temps et coût monétaire dans notre exemple). Lors de la comparaison des instances d'un même plan, seul le critère temporel sera utilisé, puisque toutes les instances correspondent aux mêmes actions, et donc au même prix. Le choix se fera en fonction des distances parcourues par l'agent pour aller de sa position actuelle jusqu'au restaurant, puis jusqu'au magasin.

Seule l'instanciation ayant reçu la meilleure note sera conservée, et sera comparée aux 3 autres meilleures instanciations des 3 autres plans. Lors de cette comparaison, seront prises en compte les priorités correspondantes aux 3 actions (P_R, P_F, P_C) et les notes des instanciations en fonction des deux critères (ici les coûts peuvent changer, puisque l'agent peut réaliser le comportement "se nourrir" de deux manières différentes).

La meilleure instanciation sera conservée, et deviendra le plan courant.

En reprenant la disposition proposée dans la figure 4.5, a priori les 3 meilleures instanciations sont indiquées sur la figure 4.6.

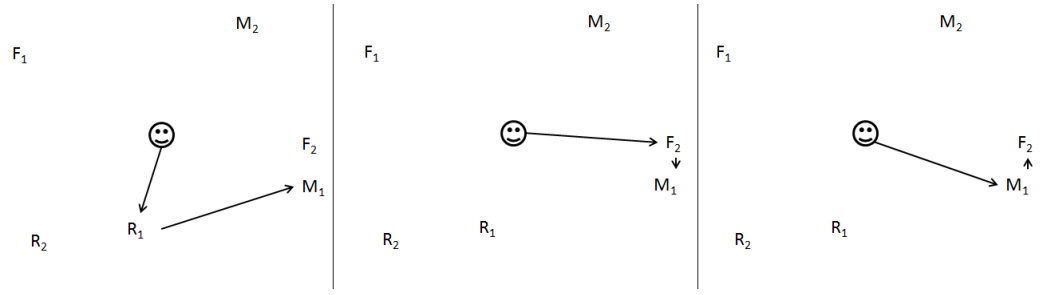

FIGURE 4.6 – Trois instanciations de plans a priori pertinentes

Le magasin M_1 étant situé à côté du fast-food F_2, il semble intéressant de les choisir toutes deux pour réaliser les actions. D'un autre côté, le restaurant R_1 étant proche de l'agent, il est également intéressant de s'y rendre immédiatement. Le choix entre ces trois plans peut se faire en fonction de la préférence de l'agent entre un restaurant et un fast-food, ou en fonction de son envie la plus forte, se restaurer ou faire du shopping. Mais il se peut tout-à-fait que dans cette situation, l'agent sélectionne une autre instanciation de plan que les trois proposées (par exemple

s'il adore le restaurant R_2 et qu'il est prêt à sacrifier beaucoup de temps pour s'y rendre).

4.7 Passage à l'échelle

Le processus décisionnel peut être complexe, et gourmand en ressources de calcul. Dans certains cas, par exemple si la simulation nécessite la simulation d'un grand nombre d'agents et que les ressources disponibles sont rares, il peut être nécessaire de simplifier ce processus afin d'assurer le passage à l'échelle. Nous avons abordé cette problématique dans la partie 3.5.8 en définissant la notion d'importance des agents. En fonction de l'importance d'un agent, il est possible de simplifier la complexité du processus décisionnel. Les points sur lesquels il est possible de jouer sont les suivants :

- **La fréquence de replanification**. Rien n'oblige tous les agents à replanifier au même rythme. La fréquence de base est définie par le type de l'agent, mais il tout à fait possible de la diminuer pour les agents les moins importants. En effet, une fréquence élevée permet aux agents d'être fortement réactifs et de prendre immédiatement en compte leurs perceptions. Mais si un agent n'est pas visible par l'observateur, rallonger son temps de réaction n'est pas gênant.
- **La profondeur de la recherche d'instanciation de plan**. Il s'agit du paramètre X de la méthode "interrogerEnvironnement" de l'algorithme décisionnel, c'est-à-dire le nombre maximal d'objets différents de l'environnement que l'agent prendra en compte afin de chercher la meilleure instanciation d'une action. En d'autres termes, il s'agit du nombre maximal d'actions concrètes que l'agent envisagera lorsqu'il cherchera à instancier un plan.

  Exemple : Soit A un agent, qui cherche à se restaurer et à regarder un film. Supposons que seules les actions "manger au restaurant" et "aller au cinéma" lui permettent d'accomplir ces objectifs. Avec X=5, l'agent prendra en compte au maximum 5 restaurants et 5 cinémas de l'environnement, soit $5^2 * 2 = 50$ instanciations de plan différentes à comparer. Avec X=3, le nombre de ces instanciations est de $3^2 * 2 = 18$.

  Diminuer ainsi le nombre des possibilités étudiées diminue l'efficacité des comportements, puisque l'agent peut ne pas voir une combinaison, ou passer à côté d'un lieu permettant un compromis intéressant. Mais, étant donné que la sémantique de l'environnement réalise un pré-traitement des informations avant d'indiquer à l'agent les lieux les plus pertinents, les comportements des agents ne deviennent pas absurdes pour autant, et gardent une cohérence par rapport à leurs critères et préférences.
- **Instaurer un nombre maximal de propositions de comportements par module de haut-niveau**. Un autre moyen simple de réduire la complexité du processus décisionnel est d'instaurer une limite de nombre de propositions de comportement, que les modules de haut-niveau ont le droit d'en-

voyer. A priori, les modules n'enverraient plus que les propositions de comportements dotées des plus fortes priorités, les comportements des agents ne seraient donc pas trop fortement touchés. Toutefois, les candidats de compromis deviendraient plus rares.

- **Désactiver certains modules de haut-niveau**. Une autre possibilité, un peu plus agressive, est de désactiver certains modules de haut-niveau présents. Dans notre cas, le module d'anticipation présenté dans le prochain chapitre serait un candidat parfait.

En plus de ces alternatives, il est possible de mettre en place un autre type de niveau de détails. En effet, si les options que nous venons de présenter permettent de simplifier le "cerveau" d'*un* agent, il est également réalisable d'agréger les cerveaux de plusieurs agents, afin qu'un seul et unique processus décisionnel dicte la conduite d'un groupe d'agents. Cette piste de recherche a été récemment explorée par [Navarro 2011, Navarro 2013b], nous ne la développerons donc pas.

4.8 Conclusion

Nous venons de présenter le mécanisme au cœur du modèle décisionnel. C'est un mécanisme qui permet de produire le comportement des agents virtuels, à partir d'un ensemble quelconque de modules de haut-niveau, pouvant être développés par des experts, sans connaissance du modèle interne des agents. Le processus décisionnel intègre des comportements hétérogènes, les décomposent en actions élémentaires, compare les différents plans possibles menant à la réalisation des intentions de l'agent, puis sélectionne celui qui répond le mieux, à la fois aux besoins de l'agent, mais aussi à ses préférences, son individualité, ainsi qu'au contexte. Ce module permet de répondre au problème de la composition de comportements.

Nous avons également présenté les différents paramètres du mécanisme, tels que la fréquence de replanification, le choix de la fonction d'agrégation des préférences, et la manière dont est utilisée la sémantique de l'environnement. Nous avons enfin expliqué l'algorithme décisionnel et donné un exemple de son fonctionnement.

Ce sujet a fait l'objet de l'article [Reynaud 2013].

4.9 Limitations et perspectives

La manière dont sont gérées les connaissances n'est pas entièrement satisfaisante. En effet, dans la version actuelle de l'architecture, c'est le module d'environnement sémantique qui prend en charge les connaissances des agents. Lors de la phase d'instanciation de plan, c'est l'environnement qui indique à l'agent quels sont les lieux dans lesquels il pourrait réaliser les actions qui l'intéressent. C'est donc l'environnement qui, en fonction du type de l'agent, décide quels sont les lieux que l'agent pourrait connaître. Cette gestion des connaissances pourrait passer du côté agent. Une première manière de procéder est de faire retenir à chaque agent les lieux qu'il a déjà visités, afin de créer une collection de lieux

connus, qui, à terme, devrait lui permettre de se passer de l'appel à la sémantique de l'environnement. La liste des lieux connus initialement pourrait, elle, être dépendante du type de l'agent (un touriste connaît moins de lieux qu'un habitant du quartier), et de son inventaire (plans, guides, etc.).

Une autre perspective intéressante serait d'intégrer un mécanisme d'apprentissage "par renforcement" [Sutton 1998, Kaelbling 1996] au module décisionnel. Cet apprentissage se baserait sur le niveau d'insatisfaction de l'agent (voir définition en 3.5.7). C'est en effet lui qui détermine la *qualité* intrinsèque des comportements de l'agent : meilleurs sont les comportements, plus le niveau d'insatisfaction a tendance à diminuer, et inversement. Un agent agissant de manière aléatoire sans prendre en compte les propositions de comportement de ses modules de haut-niveau, devrait avoir tendance à voir son niveau d'insatisfaction augmenter considérablement. Cet indice est donc parfait pour réaliser un apprentissage, centré sur les poids des modules. Ainsi, lorsqu'un comportement est adopté, le ou les modules ayant proposé ce comportement seraient récompensés (augmentation du poids associé au module) ou sanctionnés (baisse du poids) en fonction de l'impact de ce comportement sur le niveau d'insatisfaction global de l'agent.

Chapitre 5

Le processus d'anticipation

Sommaire

5.1 Introduction

5.1.1 Contexte

Le processus de planification utilisé dans le mécanisme décisionnel est proche d'une planification *dynamique*. L'idée étant que, comme l'environnement est dynamique, l'agent doit replanifier son comportement très régulièrement afin d'être réactif aux modifications de son contexte. Planifier avec une grande précision sur le long terme n'est donc pas forcément utile. Les actions à court terme sont les plus importantes puisque, une fois que ces actions auront été réalisées, l'état interne de l'agent et l'état du monde seront modifiés, et les plans devront être corrigés en fonction de ces modifications.

Cependant ce fonctionnement plonge l'agent dans l'instant présent, et ne lui permet pas de prévoir et de planifier sur le long terme. Or ces caractéristiques sont

majeures dans le comportement humain ([Pezzulo 2008]). Afin de répondre à la contrainte de crédibilité des comportements des agents, il est nécessaire d'inclure dans le processus décisionnel des capacités d'anticipation, qui permettent à l'agent de prendre en compte des prédictions lors de la phase de planification. Il ne s'agit pas forcément de planifier sur le très long terme, mais de prendre en compte de potentiels futurs événements dans une décision immédiate.

Une définition de l'anticipation qui correspond bien à nos objectifs est : "Un processus, ou comportement, qui ne dépend pas uniquement du passé et du présent, mais aussi de prédictions, attentes, ou croyances concernant le futur" [Butz 2003b]. Nous complétons cette définition avec celle-ci : "une prédiction est une représentation d'un événement futur particulier" [Pezzulo 2009b].

L'anticipation met en jeu des connaissances de la psychologie cognitive et des neurosciences. C'est un processus reconnu comme cognitif, et c'est peut-être l'un des plus importants. Par ailleurs, il est reconnu que l'anticipation intervient dans de nombreux processus cognitifs différents [Butz 2007b]. L'intégration dans l'architecture de ce type de processus a pour objectif d'améliorer la qualité des comportements des agents, c'est-à-dire améliorer la crédibilité, notre objectif principal, via une amélioration de l'efficacité des comportements.

Nous intégrons donc dans notre architecture un nouveau module de haut-niveau, appelé module d'anticipation, qui permet de contrebalancer le côté réactif des agents et de les rendre plus cognitifs. Ce module cognitif est la seconde contribution propre de cette thèse.

5.1.2 Intérêt

Après étude des travaux connexes, dans le domaine de l'anticipation pour des agents autonomes, il nous semble que la classification la plus pertinente est celle de [Butz 2007a]. Celle-ci indique qu'il en existe trois grands types : l'anticipation de récompense, l'anticipation sensorielle, et l'anticipation d'états. L'anticipation de récompense étant la plus basique, puisqu'elle n'envisage que les effets des actions. L'anticipation sensorielle est plus intéressante, mais elle implique d'appliquer des filtres sur les perceptions des agents, ce qui est peu intéressant dans notre cas. En effet, une grande partie des perceptions des agents est centralisée et mise en commun via l'environnement (appliquer des filtres impliquerait de personnaliser les perceptions des agents, et donc de perdre en généricité). La dernière est la plus ambitieuse, puisque c'est la seule qui envisage et anticipe les états futurs du monde. C'est donc sur l'anticipation d'états que nous nous concentrerons.

Les deux objectifs visés par l'ajout de capacités d'anticipation sont la crédibilité des comportements et la généricité du modèle. En effet, si rajouter de l'anticipation vise à augmenter la *qualité* des comportements, ce gain ne doit pas se faire au détriment du caractère générique de l'architecture. Il est donc impératif que le

module chargé de l'anticipation soit lui-aussi le plus générique et modulaire possible. Pour cette raison, des solutions permettant à l'agent d'anticiper afin d'éviter des situations indésirables ([Davidsson 2003]) ne sont pas souhaitables, et ceci pour plusieurs raisons.

La première raison est liée au domaine d'application principal de ce genre de travaux : la robotique. En robotique les environnements manipulés restent la plupart du temps assez simples, ce qui n'est pas toujours le cas dans des simulations multi-agents (par exemple pour les simulations en environnement urbain de grande taille). Or, caractériser une situation dans un environnement complexe peut vite devenir une tâche délicate.

Par ailleurs, en raison de l'aspect générique de l'architecture et de l'intégration de nombreuses théories différentes, il n'est pas toujours possible de caractériser simplement un ensemble de situations "non désirées" par la totalité des modules de haut-niveau (ni même pour une majorité d'entre eux). Les modules peuvent avoir des avis fortement divergents sur la désirabilité d'une situation, voire ne pas avoir d'avis du tout. Chercher à éviter des situations indésirables reviendrait donc à obliger l'intégration dans chaque module de haut-niveau d'un mécanisme de notation des situations, un processus forcément lourd et compliqué à mettre en place.

Dernièrement, cet objectif ne nous semble pas assez ambitieux. Pourquoi se restreindre à éviter certaines situations, alors que l'on pourrait chercher tout simplement à améliorer le comportement des agents, sans aucune autre restriction ? Notre modèle a une particularité très intéressante pour cela : le niveau d'insatisfaction, directement calculé à partir des propositions de comportement. Ce niveau est parfait comme valeur à minimiser, car il permet de manière très simple de noter la qualité d'un comportement. Plus un comportement contribue à diminuer la degré d'insatisfaction d'un agent, meilleur il est. Plutôt que de nous intéresser aux situations dans lesquelles l'agent peut se retrouver, on se concentre sur ses comportements.

Ce nouveau module trouvera sa place parmi les autres modules de haut-niveau, c'est-à-dire qu'il sera un module comportemental comme n'importe quel autre (voir figure 5.1).

5.2 Un besoin de modèles

5.2.1 Construire les prédictions

Afin d'anticiper, un agent a besoin de réaliser des prédictions. C'est grâce à ces prédictions que l'agent peut s'attendre à la réalisation d'événements futurs, et agir au préalable en ayant conscience de cela. Ces prédictions peuvent être de deux types : internes ou externes. Une prédiction interne correspond à une prévision de l'évolution d'un état interne à l'agent. Cet état interne peut concerner son inventaire, c'est-à-dire une caractéristique de l'agent lui-même, ou son insatisfaction, c'est-à-dire une caractéristique interne à ses modules de haut-niveau. Une prédiction

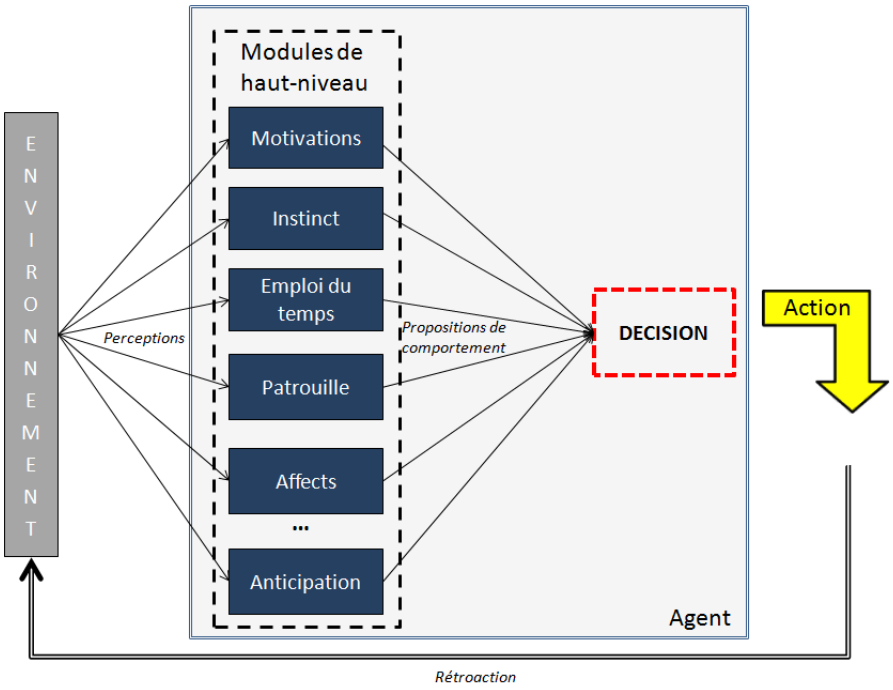

FIGURE 5.1 – Schéma de l'intégration du module d'anticipation dans l'architecture

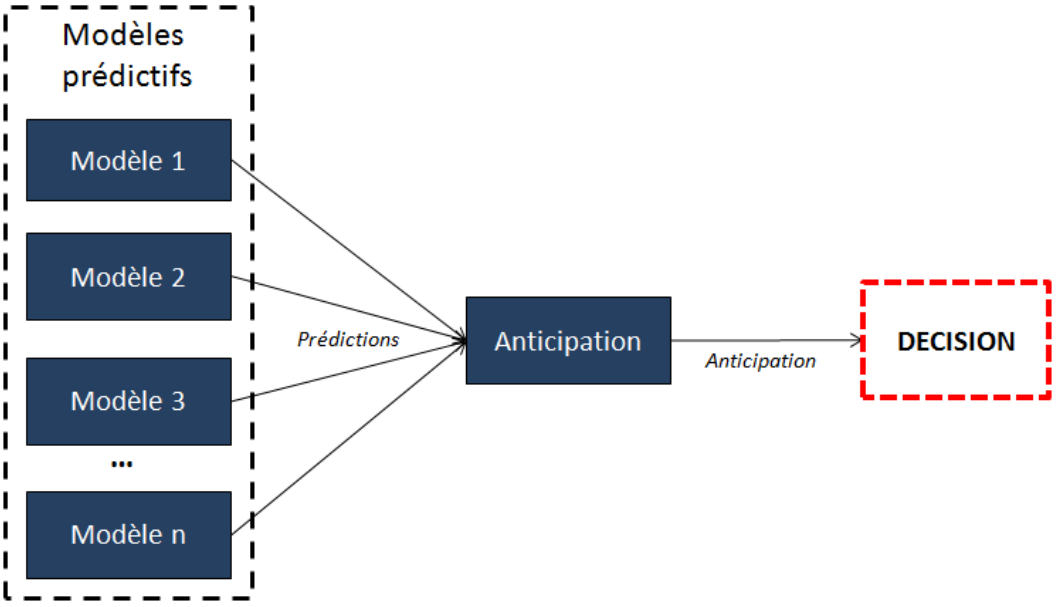

FIGURE 5.2 – Schéma haut-niveau du module d'anticipation

externe concerne un futur état du monde. La figure 5.2 donne un aperçu de haut-niveau du fonctionnement du module d'anticipation.

5.2.2 Quels modèles ?

Une prédiction ne peut se construire qu'à partir de modèles. Différents modèles entrent en considération ici :

- Un modèle du monde, pour les prédictions externes.
- Un modèle des actions, nécessaire pour que l'agent sache comment ses actions vont modifier l'environnement.
- Un modèle de chaque module de haut-niveau, pour réaliser des prédictions internes sur ses futures insatisfactions.

Les deux premiers modèles (modèle du monde et des actions) sont très classiques [Davidsson 1994], et sont souvent considérés comme un seul modèle (*World Model*). Par contre les autres modèles, ceux correspondants aux différents modules de haut-niveau, sont une nouveauté. En effet, rares sont les architectures prévoyant l'intégration de modules comportementaux divers, et ces architectures n'intègrent pas de capacités d'anticipation. L'idée d'utiliser à la fois des modules de haut-niveau quelconque et un modèle de ces modules servant à réaliser des prédictions est donc une innovation importante.

De plus, dans le chapitre précédent nous avons vu que notre méthode de planification est dynamique, et qu'elle n'est pas faite pour prévoir un comportement sur le long terme. C'est pourquoi nous rajoutons un dernier modèle : un modèle de la décision. Il permettra à l'agent de faire des prédictions sur ses futurs choix de comportements. Ainsi il sera capable, sans planification sur le long terme, de faire des prédictions sur ses futures actions.

Ces modèles font partie des connaissances de l'agent. Nous ne détaillerons pas ici les langages de représentation utilisés, puisqu'ils sont relatifs aux modèles utilisés. Tous ces modèles ne sont accessibles que par le module d'anticipation.

5.2.2.1 Modèle du monde

Le modèle le plus couramment utilisé pour réaliser des prédictions est le modèle du monde [Davidsson 1994]. En effet, grâce à une représentation modélisée de son environnement, un agent est capable de prévoir à l'avance des événements, des changements d'états, etc.

Ces prédictions sont primordiales car elles permettent d'éviter des comportements aberrants, comme un agent planifiant de partir pour un rendez-vous au moment même où il devrait déjà y être (ne pas prendre en compte une estimation du temps de trajet nécessaire), ou un automobiliste refusant de faire le plein d'essence tant qu'il en reste dans le réservoir (ne pas prendre en compte la consommation d'essence nécessaire pour aller faire le plein). Elles permettent aux agents de prévoir le temps de trajet d'un point A à un point B, ce qui leur permet d'espérer arriver à l'heure au travail, ou à un rendez-vous.

Ces prédictions couramment effectuées par des humains se font naturellement, sans que nous ayons réellement conscience de réaliser des prédictions. Cependant, pour qu'un agent puisse anticiper, même des événements paraissant évidents, il est indispensable que ces événements soient mentionnés dans le modèle de l'environnement.

Ce modèle peut être capable d'exprimer, par exemple :

- Des probabilités d'apparition d'événements conditionnées par l'heure de la simulation ou par le déclenchement d'autres événements.
- Des faits généraux sur le monde (lieux, horaires d'ouverture, conditions météo, etc.).
- Des approximations de temps de trajet.

Exemple : Probabilité de pluie un jour donné, en fonction des conditions climatiques : 20%.

Les horaires d'ouverture typiques d'un magasin quelconque sont de 9h à 19h.

Dans nos simulations, nous nous concentrons sur les estimations de temps de trajet, car ce sont les prédictions les plus souvent utilisées : à chaque fois qu'un agent veut se rendre quelque part, il est pertinent qu'il ait une idée, même approximative, du temps que cela va lui demander. Comme notre contexte applicatif est la simulation urbaine, les actions des agents ont tendance à être d'assez haut niveau (manger, travailler, faire du shopping). Contrairement à d'autres types de simulation plus centrées sur le comportement détaillé des agents (qui auraient des actions élémentaires plus bas niveau), nous nous intéressons plus à la cohésion globale des comportements. Ceci explique que nos actions sont plutôt longues, et peu d'actions peuvent se réaliser au même endroit. Dès qu'un agent se projette dans le futur, même à court terme (2 à 3 actions dans le futur), cette projection l'amène quasiment toujours à se déplacer mentalement dans la ville, et donc à être capable de prévoir un temps de trajet. Pour ce faire, nous avons pris le parti de la simplicité, et nous calculons une estimation de temps de trajet à pied grâce à la formule suivante :

$EstimationTempsDeTrajet = VitesseAgent * d_M$

Avec $VitesseAgent$ la vitesse de référence utilisée par le module de navigation (c'est un paramètre interne à l'agent), et d_M la distance de Manhattan entre le point de départ et le point d'arrivée.

Rappel : La distance de Manhattan d_M entre deux points A et B de coordonnées respectives (X_A, Y_A) et (X_B, Y_B) est la distance associée à la norme A, c'est-à-dire :

$d_M = |X_B - X_A| + |Y_B - Y_A|$

Nous avons choisi cette distance plutôt qu'une autre car, bien que le trajet d'un agent soit, à Paris, quasiment toujours plus court que cette distance, elle permet de prendre en compte le temps perdu lors des diverses traversées de chaussées, et

semble donc un compromis intéressant.

Les autres connaissances liées au modèle du monde sont moins critiques par rapport aux prédictions des agents, et une grande partie de ces connaissances peut être apprise (voir 5.2.4).

5.2.2.2 Modèles des actions

Le modèle des actions indique les connaissances, que l'agent possède sur les effets des actions qui peuvent avoir lieu dans la simulation. Ces connaissances peuvent porter sur n'importe quelles caractéristiques de l'action : durée, effets, etc. Il est possible de considérer le modèle des actions comme faisant partie du modèle du monde, mais comme cette partie est bien distincte, et correspond à des notations et connaissances très spécifiques, nous l'avons séparée du modèle du monde.

Exemple : L'action "retirer de l'argent" permet d'augmenter la variable d'inventaire "argent liquide".
Acheter une boisson dans un distributeur coûte 1 euro.

Ce modèle est absolument crucial dans le but de réaliser des prédictions, en effet les actions sont le principal vecteur des modifications tant internes à l'agent, qu'externes. Sans un modèle des actions, un minimum précis et complet, l'agent ne pourra aucunement prévoir l'évolution de son état interne ni de l'état de monde. Or il se trouve que dans le cadre de nos simulations, le modèle d'action utilisé reste assez incertain (voir 3.3.2) : les durées et les coûts monétaires sont tirées aléatoirement dans une fourchette assez grande. Nous avons donc décidé de donner directement aux agents le modèle d'actions véritablement utilisé dans la simulation, plutôt que d'en faire un modèle. Nous garantissons donc une bonne précision, tout en gardant une forte composante aléatoire. Ce fonctionnement est, a priori, réaliste, puisqu'il imite la manière de fonctionner des humains. Nous sommes en effet capables, quelle que soit l'action, de donner une fourchette de temps et de prix pour la réalisation de cette action.

Ces deux premiers modèles sont des modèles externes, qui concernent l'environnement. Les deux suivants sont internes, et concernent directement l'agent.

5.2.2.3 Modèles des modules de haut-niveau

Chaque module de haut-niveau peut être modélisé de manière à ce que l'agent puisse formuler des prédictions concernant les futures propositions de comportements que ces modules enverront, ainsi que sur leurs niveaux d'insatisfaction respectifs.

Les modules de haut-niveau sont des boîtes noires du point de vue du module décisionnel de l'agent. Aucun détail de leur activité interne n'est connu. Afin que la décision soit capable de prédire leur fonctionnement, il est nécessaire de lui fournir des modèles ressemblants. Notons que l'objectif de ces modèles est de décrire l'évolution des propositions de comportement, et pas de modéliser le véritable fonctionnement interne du modèle. C'est le même genre d'objectif que celui de nos agents, qui est d'exhiber des comportements crédibles ou "d'avoir l'air intelligent", et pas d'imiter le fonctionnement interne du cerveau humain.

Exemple d'informations données afin de modéliser un module motivationnel : Le module motivationnel propose toujours le comportement "se nourrir".
La priorité de cette proposition augmente de 0.2 toutes les heures.

Mais cette modélisation ne serait pas complète si elle n'était pas capable de prendre en compte correctement les événements de la simulation. En effet, le modèle des actions présenté précédemment est loin d'être complet, puisque chaque action peut avoir des effets extrêmement différents sur chacun des modules actifs d'un agent (voir 3.4.2.2). Il en va de même pour les différents événements pouvant intervenir.

Exemple : Une explosion peut augmenter la peur pour un module d'émotions, mais peut provoquer un comportement très précis pour un module prenant en charge le comportement des policiers.

Le rôle de toute une partie des modèles des modules de haut-niveau sera donc de modéliser les impacts des divers événements.

Exemple : L'action "manger au restaurant" réinitialise à 0 la priorité de la proposition de comportement "se nourrir" du module motivationnel.

Dans le cas de nos simulations, 3 modules de haut-niveau doivent être modélisés. Les modules d'emploi du temps et d'instinct sont tellement simples, que nous avons choisi de prendre le modèle véritablement utilisé dans la simulation comme référence. Par contre nous avons jugé que le module de motivation pouvait être simplifié, en supprimant les variables internes du modèle. Ainsi seules les variables de motivations sont prises en compte. Le modèle est imparfait, mais sa précision suffisante pour l'utilisation qui en est faite.

5.2.2.4 Modèle de la décision

En plus de ces modèles, nous considérons également un modèle du module de décision, qui est capable, à partir d'un état futur (à la fois interne et externe) anticipé, de prédire le plan préféré, et donc la prochaine action de l'agent. Ce modèle peut ainsi être utilisé plusieurs fois de suite afin de prédire un enchaînement d'actions.

Ce modèle est le plus sensible. En effet, c'est celui devant modéliser le module le plus complexe, et celui prenant en compte le plus de paramètres. C'est donc le modèle le plus vulnérable face aux imprécisions de modélisation.Le module décisionnel étant au cœur de l'architecture, sa modélisation est donc particulièrement délicate : le modéliser grossièrement interdirait toute projection viable dans le futur (en tout cas plus loin que la fin de l'action en cours). De plus le module décisionnel est suffisamment complexe pour que sa modélisation soit un travail long et complexe.

C'est pourquoi, dans le cadre de nos simulations, nous avons utilisé le module décisionnel comme simulateur de lui-même. C'est-à-dire que lorsque l'agent réalise des prédictions sur ses futurs choix de comportement, il se sert du modèle décisionnel véritablement utilisé, et l'applique à une situation virtuelle anticipée. La qualité des prédictions est donc excellente, au prix d'un coût en ressources de calcul élevé, puisque les calculs ne sont pas simplifiés. La question de l'efficacité de ce modèle par rapport à sa complexité sera posée dans le chapitre des évaluations et des expérimentations (section 6.4.1.1).

5.2.3 Comment obtenir ces modèles ?

Dans une première approche, le moyen le plus simple de fournir ces modèles à l'agent est de les faire développer par le modélisateur. Un des principaux avantages de fonctionner de la sorte est que le modélisateur est maître de la qualité de ses modèles, et peut donc maîtriser leur complexité. Il peut également choisir le type de réalisme recherché. Par exemple, il est possible de fournir aux agents un modèle réaliste du monde. Mais il est souvent plus intéressant de leur fournir un modèle réaliste des connaissances des habitants de la ville.

Exemple : Imaginons une simulation cherchant à montrer l'impact de travaux sur une voie de circulation entraînant le blocage d'une rue. Le modèle du monde, par rapport à la situation "normale" est modifié de telle sorte que la rue en question ne soit plus accessible par les véhicules. Cependant, si les agents ont une connaissance du monde parfaite, ils vont tous planifier des chemins n'empruntant pas cette rue, et l'impact simulé des travaux sur la circulation ne sera pas du tout correct.

Par contre, si le modélisateur leur fournit un modèle de connaissance réaliste pour des habitants de la ville, seuls les habitants du quartier auront connaissance de ces travaux. Une partie des agents va donc potentiellement planifier un chemin passant par cette rue pour aller travailler, et devront replanifier une fois leurs connaissances mises à jour. Le trafic simulé, notamment aux alentours des travaux, sera plus proche de la réalité.

Notons que si les modèles d'actions, d'environnement et de décision sont toujours modélisables, ce n'est pas forcément le cas de tous les modules de haut-niveau. En effet, rappelons que ces modules peuvent représenter n'importe quel mécanisme de proposition de comportement. Il est possible par exemple de créer un module de haut-niveau entièrement aléatoire. Sans aller jusque là, de nombreux processus cog-

nitifs sont extrêmement complexes, voire impossibles, à anticiper (l'état émotionnel ou les envies subites en sont des exemples). Dans ce cas, il est envisageable de ne pas essayer d'anticiper ce genre de comportement et de ne pas fournir à l'agent le modèle correspondant. L'agent sera donc incapable d'anticiper certains types de comportement, tout comme nous le sommes.

5.2.4 Apprentissage des modèles

Une extension particulièrement intéressante, mais non développée dans le cadre de cette thèse, est de réaliser un apprentissage sur ces modèles. L'apprentissage pouvant se réaliser à partir d'un modèle fourni en entrée, ou à partir de rien. Grâce à des processus d'apprentissages avec boucle de rétroaction, il est possible d'affiner les modèles existants, de réduire les marges d'erreur et de calculer des degrés de confiance. En effet, dans tous ces apprentissage, la gestion d'un degré de confiance est très importante. Ce degré augmenterait avec le nombre de prédictions concordantes, et diminuerait avec celles ne concordant pas (pour finir par éliminer la prédiction si la confiance est trop faible).

5.2.4.1 Modèle du monde

Il est possible de suivre la piste de recherche proposée par [Capdepuy 2007], qui permet, à partir de l'étude d'une liste d'événements, d'inférer des relations entre ces événements. L'idée serait ici plutôt un apprentissage hors ligne, avec étude du déroulement de plusieurs journées. Dans un premier temps il serait possible de se concentrer sur la recherche de lien temporels entre événements, que ce soit un lien entre l'apparition d'un événement et l'heure de la journée, ou des liens entre événements.

Exemple : Le magasin M ferme tous les soirs à 20h.
L'événement A se produit toujours entre 20 et 30 minutes après l'événement B.

En ce qui concerne les temps approximatifs de trajet entre 2 points, plusieurs possibilités sont envisageables. Une option intéressante serait d'enregistrer tous les trajets effectués par l'agent, ainsi que le temps ayant été nécessaire. A partir de cette base de connaissances, il faudrait ensuite essayer d'extraire une relation entre distance et temps. Plusieurs fonctions de calcul de la distance pouvant être testées (distance euclidienne, distance de Manhattan, etc.). Si aucune relation n'est trouvée, cela signifie que l'apprentissage des temps de trajet nécessitera des capacités de recherche de chemin, ce qui est une option envisageable, mais coûteuse.

5.2.4.2 Modèle des actions

En ce qui concerne le modèle des actions, l'apprentissage est plus simple à réaliser, même en partant d'un modèle vierge. En effet, dans une même simulation, les actions ont a priori toujours les mêmes effets, à quelques variations près. Il suffit

donc d'enregistrer l'ensemble des effets d'une action, et de considérer que la répartition des observations est celle des effets de l'action. Le modèle appris s'affinera donc de lui-même, jusqu'à devenir extrêmement proche du modèle véritablement utilisé dans la simulation au bout d'un nombre suffisant d'observations.

Cependant, si le modèle appris ne converge pas, c'est que le modèle des actions véritablement utilisé dans la simulation, prend en compte des probabilités d'échec, ou des variations d'effets en fonction de facteurs internes à l'agent ou de facteurs relatif à l'environnement. Dans ce cas, l'apprentissage devient beaucoup plus complexe, voire impossible, puisque l'agent doit prendre en compte de nombreux paramètres, dont certains peuvent lui être impossibles à connaître (les agents n'ont pas une connaissance parfaite du monde).

5.2.4.3 Modèle des modules de haut-niveau

Le problème de l'apprentissage de la modélisation des modules de haut-niveau est la méconnaissance totale des théories utilisées. Sans apport initial de connaissances, il est très difficile d'apprendre ces modèles, puisque l'état "complet" du monde doit être pris en compte et que la processus d'anticipation n'y a pas accès. Des processus d'apprentissages non-supervisés pourraient donner des résultats exploitables.

5.3 La planification par anticipation

Grâce aux modèles présentés précédemment, un agent doté du module d'anticipation est capable de prédire l'état du monde, ainsi que son propre état interne à un moment donné du futur. A partir de ces prédictions, il est capable d'utiliser son modèle de la décision pour prédire le comportement qu'il sélectionnerait dans ces conditions. Et ce processus est répétable, avec une perte de précision, et une augmentation de l'incertitude (liée à l'imperfection des modèles utilisés) à chaque pas dans le futur anticipé.

Nous appelons ce processus **planification par anticipation**, puisqu'il permet à l'agent d'anticiper le déroulement de son plan courant, et d'élaborer une meilleure stratégie. C'est-à-dire un plan intéressant à appliquer immédiatement, pour de futures raisons. La qualité d'un plan dépend directement du niveau d'insatisfaction de l'agent à son issue, si un plan permet d'atteindre un meilleur niveau de satisfaction, alors il est préférable.

5.3.1 Algorithme

Le processus de planification par anticipation que nous avons développé est inspiré de travaux classiques [Davidsson 1994, Doniec 2008], mais utilise les particularités de notre système décisionnel, en particulier le système de propositions de comportement. Par ailleurs, l'utilisation de modèles des modules de haut-niveau

donne à l'algorithme de l'anticipation une forme entièrement nouvelle. Cet algorithme est décrit ci-dessous :

Données : Soit A un agent;
Soit t_c l'heure exacte de la simulation;
Soit t = t_c;
tant que *ConditionArrêt est fausse* **faire**
  PrédictionActionSuivante(A,t);
  PrédictionFinAction(A,t);
  PrédictionEtatsFutur(A,t);
  PrédictionFuturesPropositionsComportements(A,t);
  RechercheMeilleurPlan() ;
fin

Algorithme 3 : Algorithme de planification par anticipation

avec :

- **PrédictionActionSuivante(A,t)** : si t est égal à t_c (c'est-à-dire si c'est la première itération de l'algorithme), la méthode renvoie l'action courante de l'agent A, au temps t. Sinon, si t > t_c, la méthode utilise le modèle de la décision pour déterminer quelle action devrait être sélectionnée par le module décisionnel véritablement utilisé dans la simulation, pour un état global du monde tel que déterminé par la méthode PrédictionEtatsFutur, et des propositions de comportements calculées par la méthode PrédictionFutures-PropositionsComportements. L'action retournée est notée a.
- **PrédictionFinAction(A,t)** : cette méthode utilise le modèle des actions pour calculer une estimation du temps restant t_r avant que l'action a se termine. La méthode met à jour t : t = t+t_r.
- **PrédictionEtatsFutur(A,t)** : cette méthode utilise le modèle de l'environnement pour calculer l'état futur de l'environnement (probabilité d'occurrence des événements, etc.). Elle utilise également le modèle des actions afin de prendre en compte les effets de l'action a. Les différents modèles internes sont également utilisés pour mettre à jour les états internes des modules de haut-niveau de l'agent. Cette méthode réalise donc une prédiction sur l'état global du monde à l'instant t.
- **PrédictionFuturesPropositionsComportements(A,t)** : cette méthode va prendre en entrée l'état global du monde prédit par la méthode précédente, et va utiliser les différents modèles des modules de haut-niveau pour prédire les propositions de comportements que ces modules feraient dans ces conditions. De plus, cette méthode permet d'estimer I_t, l'insatisfaction de l'agent A au temps t (voir 3.5.7).
- **RechercheMeilleurPlan()** : l'agent a été capable d'effectuer une prédiction de sa future insatisfaction I_t, au temps t dans le futur. La méthode Recherche-MeilleurPlan, va essayer de chercher un plan (i.e. une combinaison d'action) qui permettrait d'obtenir une meilleure future satisfaction. Si un tel plan est trouvé, il est envoyé au module décisionnel sous forme d'une proposition de

comportement, dont la priorité dépend à la fois du **degré de confiance**, ainsi que de **l'importance** de ce gain (voir partie 5.3.4.

- **La condition d'arrêt** sera discutée dans la partie 5.3.2.

La figure 5.3 montre le fonctionnement du module d'anticipation.

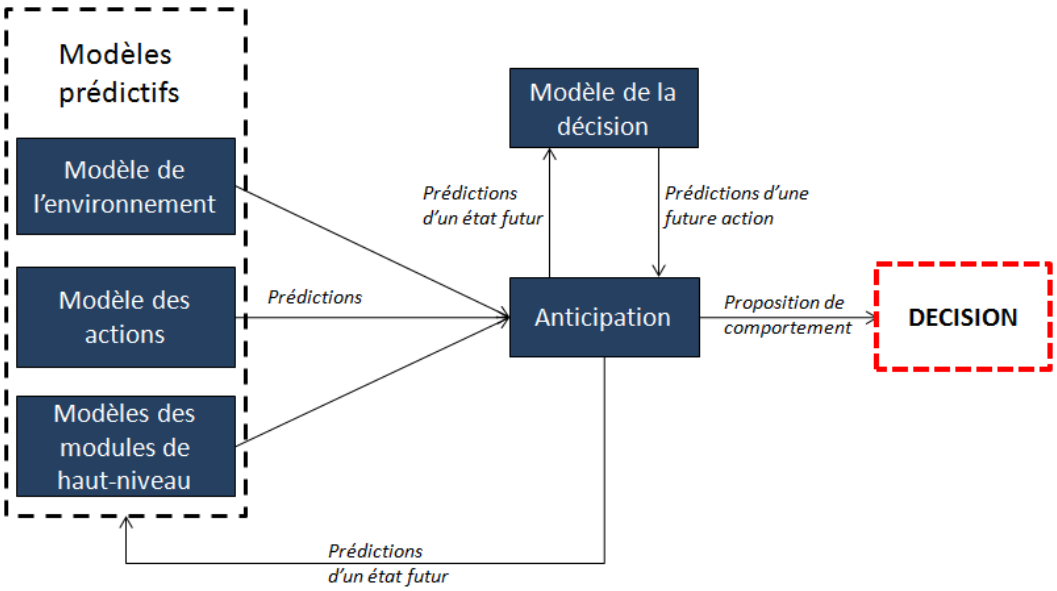

FIGURE 5.3 – Schéma du fonctionnement du module d'anticipation

5.3.2 Condition d'arrêt : l'horizon temporel

Le processus de la planification par anticipation est coûteux. Il fait intervenir un certain nombre de modèles, et notamment le modèle de la décision (qui dans notre cas est le module de décision lui-même). Chaque itération de cet algorithme est donc coûteux, à la fois en ressource et en temps. Par ailleurs, plus l'agent va essayer d'anticiper sur le long terme, c'est-à-dire plus il va itérer le processus, plus la précision des prédictions sera faible. Quelle que soit la qualité intrinsèque des différents modèles, plus l'horizon temporel anticipé est grand, plus les erreurs de prédictions seront importantes. La question de la condition d'arrêt du processus est donc primordiale. S'arrêter trop tôt revient à ne rien anticiper, et donc ne rien pouvoir prévoir sur le long terme, et s'arrêter trop tard revient à consommer énormément de ressources pour un résultat potentiellement inutile, voire contre-productif. En effet, si les erreurs de prédictions sont suffisamment importantes, le module d'anticipation pourrait détecter un plan qu'il jugerait, à tort, comme meilleur que le plan initial.

Différentes possibilités sont envisageables pour gérer cette condition d'arrêt. La plus naturelle est peut-être une simple durée : les agents essayent d'anticiper sur

une durée déterminée à l'avance, au delà de laquelle on considère que les résultats ne sont plus assez valables pour être pris en compte. Cette durée peut-être déterminée de plusieurs manières : en fonction des ressources disponibles, du nombre d'agents simulés, ou après avoir réalisé des tests de la qualité des prédictions en fonction de la durée anticipée.

Une autre condition d'arrêt naturelle serait un nombre d'itérations maximum, c'est-à-dire un nombre maximum d'actions anticipées dans le futur (chaque itération de l'algorithme permet à l'agent de se projeter "une action plus loin" dans le futur). En effet, les erreurs de prédictions les plus dommageables pour le processus de planification par anticipation sont celles concernant la prédiction de la future action. En effet, une légère erreur dans une variable d'inventaire, ou dans la date de déclenchement d'un événement ne provoque pas une grande divergence de prédiction. Par contre la moindre erreur dans la prédiction de la prochaine action peut causer une divergence très importance entre les prédictions et "réalité" de la simulation. Limiter le nombre maximal de prédictions de ce type permet d'éviter des erreurs trop lourdes.

D'autres types de critères d'arrêt sont possibles, en fonction des besoins et caractéristiques de la simulation. La durée moyenne des actions est par exemple déterminante quant à cette condition d'arrêt, tout comme la précision des différents modèles utilisés (inutile de chercher à anticiper loin avec des modèles très simplifiés).

Dans le cas de nos simulations, nous préfèrerons une condition d'arrêt basée sur un nombre maximal d'itération, afin de maîtriser le nombre d'appels au modèle de la décision. La majorité de nos actions ont une durée assez courte. Dans d'autres scénarios avec des actions plus longues, ou dont la durée est beaucoup plus variable, il pourrait être intéressant de choisir un autre critère, voire une combinaison de plusieurs critères.

5.3.3 Fréquence de la planification par anticipation

Si la question de l'horizon est intéressante, celle de la fréquence de lancement du processus doit également être posée. Ici encore, une fréquence trop faible risquerait de ne pas détecter un certain nombre de comportements intéressants, alors qu'une fréquence trop importante serait contre-performante (détection plusieurs fois du même comportement).

Pour les mêmes raisons que précédemment, cette fréquence ne peut être donnée, ni calculée à l'avance. Elle est subjective, et dépend de la simulation, du nombre d'agents simulés, etc. Cela signifie que le module d'anticipation ne va pas forcément relancer un processus d'anticipation lors de chaque nouveau processus décisionnel. Les processus d'anticipation ont lieu de manière asynchrone par rapport aux processus décisionnels, et les propositions de comportements détectées sont reproposées tant qu'un nouveau processus d'anticipation n'a pas eu lieu (ou que le comportement en question a été terminé).

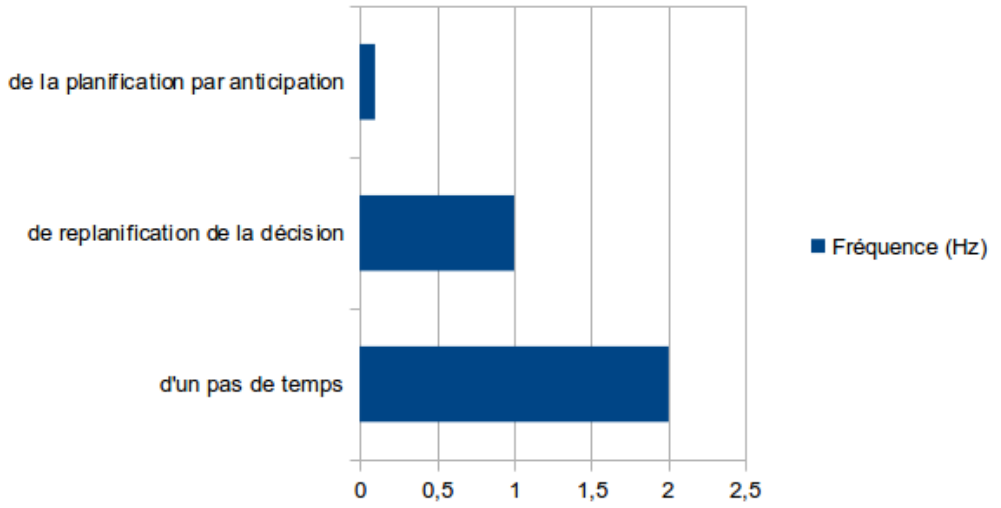

FIGURE 5.4 – Graphique présentant la fréquence de la planification par anticipation par rapport à celle de replanification du module décisionnel et celle d'un pas de temps.

La figure 5.4 présente la fréquence de la planification par anticipation que nous avons choisie par rapport à celle de replanification du module décisionnel (voir section 4.4.1), et celle d'un pas de temps.

Dans l'ensemble des simulations que nous avons réalisées, la durée d'un pas de temps a été fixée à 50 ms, celle de replanification du processus décisionnel a été fixée à 1 seconde (car la vitesse de réaction d'un humain est de l'ordre de la seconde), et la vitesse de planification du processus d'anticipation a été fixée à 10 secondes (voir la section 6.4.1.2 pour une description des évaluations ayant mené à ce choix de fréquence).

5.3.4 Les propositions de comportement anticipées

5.3.4.1 Envoi de propositions

Le processus décrit dans la partie 5.3.1 permet de détecter des comportements utiles immédiatement, pour des raisons détectables uniquement dans le futur. Le critère permettant la comparaison entre l'intérêt de plusieurs comportements est le niveau d'insatisfaction. Si l'adoption d'un autre comportement permet d'atteindre un meilleur niveau de satisfaction, alors cet autre comportement est préférable.

Lorsqu'un tel comportement est détecté par le processus de planification par anticipation, le module d'anticipation doit le communiquer au module décisionnel, afin qu'il soit pris en compte dans le processus décisionnel immédiat.

5.3.4.2 Poids de ces propositions

Les propositions envoyées par le module d'anticipation sont des propositions de comportement au même titre que toutes les autres : il n'y a pas de raison de leur attribuer un poids plus important, à cause de leur origine plus "cognitive". Comme tous les autres modules de haut-niveau, le module d'anticipation est doté d'un poids qui modifie toutes les propositions de comportements sortantes, et qui représente l'intérêt que l'agent porte à ce module. Ainsi il est possible de modéliser des agents très prévoyants, prêtant une grande attention aux propositions de leur module cognitif, et d'autres agents beaucoup plus insouciants, qui ont tendance à ne pas vraiment les prendre en considération.

Par ailleurs, la priorité de ces propositions de comportements dépend du gain espéré de satisfaction, c'est-à-dire de la différence entre la satisfaction estimée si l'agent exécute les comportements "normaux" prédits, et celle estimée si l'agent choisit de suivre le comportement proposé par le module d'anticipation. Plus le gain anticipé est important, plus adopter cet autre comportement devient intéressant. Cette priorité doit également dépendre du degré de confiance du module d'anticipation concernant ce gain. Plus les prédictions sont certaines, et plus l'adoption du comportement est intéressante.

La formule suivante indique la manière dont est calculé la priorité d'une proposition de comportement venant du module d'anticipation :

$$Priorité = Gain * Confiance$$

$$\text{avec } Gain = \frac{I_E - I_P}{I_E} \text{ et } Confiance = \frac{1}{N_I}$$

I_E étant le niveau d'insatisfaction estimé si le plan "normal" est suivi, I_P étant le niveau d'insatisfaction prévu si le plan proposé par le module d'anticipation est suivi, et N_I étant le nombre d'itérations du processus de planification par anticipation ayant été nécessaire avant de trouver le "meilleur plan".

Remarque : On note que si un plan permet de passer à une insatisfaction nulle en une seule itération, la priorité est de 1. Seuls les plans permettant d'obtenir une meilleure satisfaction que le plan normal étant étudiés, la priorité minimale est de 0. Les priorités sont donc dans l'intervalle $]0; 1]$. On note qu'il est possible d'agrandir cet intervalle à $[-1; 1]$ si l'on veut pouvoir empêcher certain comportements, en plus d'en proposer d'autres.

Remarque : Il est tout à fait possible de choisir d'autres fonctions de calcul du gain ou de la confiance. Par exemple en faisant entrer dans le calcul de la confiance les différents degrés de confiance des modèles utilisés.

5.4 Exemple

Afin d'illustrer ce chapitre, nous donnons un exemple simplifié du déroulement d'un processus de planification par anticipation. Cet exemple n'est pas imaginaire, il est tiré d'une simulation réellement effectuée lors des évaluations du module d'anticipation (voir section 6.4).

Soient A et A' deux agents, tous deux dotés d'un module de motivations et d'un module d'emploi du temps. A la différence de A, A' est doté du module d'anticipation. A part cette différence, les deux agents sont rigoureusement identiques : même individualité, mêmes préférences, même emploi du temps, mêmes motivations, même lieu d'habitation, etc. Étudions le comportement de A, et comparons le à celui de A'.

Selon son emploi du temps, A doit être à son travail à 8h. Seulement le module d'emploi du temps ne déclenche le comportement "travailler" qu'à l'heure pile à laquelle il doit travailler, c'est-à-dire 8h. Ce qui fait que jusqu'à 8h, A reste chez lui à satisfaire sa motivation la plus élevée (l'envie de divertissement) en regardant la TV. A 8h, le module d'emploi du temps envoie sa proposition de comportement (dotée d'une priorité élevée), donc A sort de chez lui et se rend à son travail sans avoir mangé. Il arrive en retard puisqu'il est parti à l'heure à laquelle il aurait dû arriver. A midi, heure à laquelle il a une pause, il est affamé, puisqu'il n'a pas mangé de la journée.

Comparons maintenant ce qu'il se passe lorsque le module d'anticipation est activé :

A 7h45, A' est en train de regarder la TV chez lui, son module d'anticipation lance un processus de planification par anticipation. Il estime que l'agent devrait regarder la TV encore 1h (estimation tirée du modèle des actions). Il se projette donc à 8h45, prend en compte les effets de l'action "regarder la TV", met à jour l'état interne de l'agent, et essaye de prédire les propositions de comportement qui seraient envoyées à ce moment là. La plus prioritaire de ces propositions serait celle envoyée par le module d'emploi du temps (à 8h45, il devrait être au travail). La satisfaction de l'agent est donc assez faible, puisque le module d'emploi du temps est largement insatisfait (l'agent n'est pas au travail alors qu'il le devrait). Le module d'anticipation recherche donc un meilleur plan, lui permettant d'obtenir une meilleure satisfaction à 8h45. Or, il se trouve que le meilleur plan qu'il trouve est de rester devant la TV jusqu'à 7h50, puis de partir travailler (son modèle du monde lui indique qu'il lui faut environ 10 minutes pour aller au travail). A 7h50, le module d'anticipation envoie donc une proposition de comportement "travailler", avec une priorité assez forte puisque le gain espéré est important. A' part donc au travail à 7h50. Sur le chemin, un nouveau processus de planification par anticipation est lancé. Il anticipe que l'agent arrivera pile à l'heure à son travail, aucun meilleur plan n'est trouvé. Par contre, à l'itération suivante, il se projette jusqu'à la fin de

l'action suivante, à savoir "travailler", ce qui l'amène à midi (grâce au modèle des actions). A cet instant, il anticipe que son niveau d'insatisfaction sera élevé en raison d'une faim très importante (grâce au modèle du module motivationnel). Il cherche donc un plan alternatif, et propose à l'agent d'acheter quelque chose à manger sur le chemin (son modèle du monde lui indique qu'il existe une boulangerie sur le trajet).

A' arrive donc à l'heure à son travail (ou quelques minutes en retard si son modèle du monde est incorrect, ou qu'il a mis trop de temps à la boulangerie), et ne passe pas toute la matinée à avoir faim.

Remarque : Dans cet exemple, l'horizon temporel du processus de planification par anticipation est fixé à 2 : seules 2 actions dans le futur sont anticipées. Si on avait fixé à 3 cet horizon, l'agent aurait pu à 7h45 prévoir qu'il allait (1) regarder la TV, (2) se rendre à son travail, et (3) travailler jusqu'à midi. Il aurait donc potentiellement pu directement proposer comme plan, de partir travailler et de manger sur la route (en prenant en compte le temps nécessaire à la boulangerie afin qu'il soit sûr d'arriver à l'heure).

5.5 Passage à l'échelle

L'ajout de ce module d'anticipation permet d'augmenter la qualité des comportements des agents (voir expérimentations en 6.4), ce qui répond à la problématique liée à la crédibilité des comportements. Cependant ce module a un coût computationnel important, il est donc nécessaire de s'assurer que la contrainte liée au passage à l'échelle est toujours satisfaite.

Plusieurs possibilités sont envisageables afin de réduire la consommation de ce module. La première et la plus évidente est de jouer sur les divers paramètres : horizon et fréquence. S'il est nécessaire de simuler un grand nombre d'agents dotés du module d'anticipation, il est possible de réduire l'horizon temporel envisagé ainsi que la fréquence à laquelle les agents tentent d'anticiper leurs comportements. Aucune solution générique n'est envisageable, toutes les simulations sont uniques. Des tests seront donc nécessaires afin de déterminer les valeurs "optimales" de ces paramètres, ainsi que la marge qu'il est possible de prendre.

Par ailleurs, un autre moyen de réduire les temps de calculs nécessaires pour le fonctionnement de ce module est de simplifier les modèles utilisés, voire d'en supprimer certains. Plus les modèles utilisés seront complexes, plus ils ont de chance d'être précis, mais plus ils seront coûteux à utiliser. De plus, si certains modules sont difficiles à modéliser, ou si certains modèles sont peu fiables, il peut être aussi pertinent de ne pas les prendre en compte, et de les supprimer de la base de connaissances de l'agent.

Enfin, dernier point, le module d'anticipation est un modèle très facile à désactiver en cours de simulation. En effet, contrairement à un module de motivation qui, s'il est désactivé pendant longtemps, peut conduire à des valeurs de motiva-

tions peu crédibles lors de sa réactivation, le module d'anticipation ne propose pas de comportements propres, ni ne manipule de variables caractéristiques. Il ne fait qu'*améliorer* les comportements des agents. Il est donc à la fois simple, mais également très profitable (le module d'anticipation coûte cher !) de désactiver le module d'anticipation pour les agents jugés peu importants. Un fonctionnement adaptatif, en fonction de l'importance relative de l'agent est donc fortement recommandé.

Pour finir, il est tout à fait envisageable de coupler ces pistes, en ajustant les paramètres, la qualité des modèles utilisés, et l'activation du module en fonction du degré d'importance de l'agent.

5.6 Conclusion

Nous venons de présenter la troisième contribution de cette thèse, à savoir un mécanisme d'anticipation, capable de s'intégrer au processus décisionnel. Grâce à son fonctionnement modulaire, il est capable de fonctionner de manière générique, et de réaliser des prédictions sur le fonctionnement des différentes théories intégrées dans l'architecture. Le processus, appelé "planification par anticipation", permet à l'agent de se projeter dans le futur, avec l'ensemble des informations à sa disposition, et de chercher un moyen de se retrouver avec une meilleure satisfaction dans le futur. Les situations non désirées sont ainsi évitées, sans qu'aucune notation des situations n'existe, uniquement grâce au niveau d'insatisfaction. Le processus en lui-même est largement paramétrable, d'une part via les différents modèles utilisés, et d'autre part via des paramètres tels que la condition d'arrêt, et la fréquence d'anticipation.

Ce module est également générique et peut s'intégrer à n'importe quel type d'architecture, tant que les modèles prédictifs nécessaires à son fonctionnement lui sont fournis, et qu'il a accès à une méthode permettant de calculer un niveau de satisfaction à partir de l'état de l'environnement et des états internes de l'agent. Un article a d'ailleurs été publié sur la question spécifique du module d'anticipation [Reynaud 2012].

5.7 Limitations et perspectives

Ce processus souffre de certaines limites. La principale est son coût élevé en temps de calcul. Chaque processus d'anticipation nécessite l'intervention de nombreux modèles, devant se projeter plusieurs "pas de temps" dans le futur. L'activation pour un agent, du module d'anticipation, n'est pas gratuite. Ce qui touche directement une autre contrainte, celle liée au passage à l'échelle. En effet, si les agents sont plus gourmands en ressources, il devient plus difficile d'en simuler un très grand nombre en parallèle.

Par ailleurs, tout est centré sur le niveau d'insatisfaction. Or ce critère de notation de la qualité des comportements est discutable. D'autres critères pourraient être envisagés, et testés.

De nombreuses autres perspectives sont intéressantes, comme l'intégration d'un

mécanisme permettant de focaliser les processus d'anticipation sur certains types de comportements. Certains comportements sont en effet plus faciles, et plus intéressants à anticiper (typiquement les comportements ayant une date ou heure limite après laquelle ils ne sont plus réalisables). D'autres comportements sont plus difficiles à anticiper, et d'un intérêt limité. Il est donc intéressant d'ajouter dans le processus d'anticipation, un mécanisme permettant de reconnaître et de s'adapter, en fonction des comportements à anticiper. De plus, ce mécanisme permettrait de restreindre l'utilisation de l'anticipation, et donc d'économiser du temps de calcul en vue de passer à l'échelle.

Soulignons également une autre piste de recherche qui s'intéresserait à la détection automatique du meilleur horizon pour le processus de planification par anticipation. Ceci grâce, par exemple, à un enregistrement des prédictions, et leur comparaison avec les faits réels se produisant un peu plus tard, afin de déterminer la perte de précision liée au critère temporel. Une autre solution consisterait à calculer automatiquement une valeur de confiance de chaque prédiction, et de fixer l'horizon par rapport à cette valeur : dès que la valeur de confiance passe en-dessous d'un certain niveau, l'agent arrête le processus de planification par anticipation.

Chapitre 6

Évaluations et expérimentations

Sommaire

6.1 Introduction

6.1.1 Contexte

Dans les parties précédentes nous avons présenté les différentes pistes de recherche de cette thèse. Dans cette partie nous allons exposer les expérimentations

que nous avons réalisées et discuter des résultats obtenus afin de tester la validité de nos affirmations.

Deux types bien distincts d'expérimentations ont été réalisés : des expérimentations objectives, et des expérimentations subjectives. En effet, parmi les thèses que nous défendons, certaines peuvent être vérifiées de manière objective, grâce à des mesures et des comparaisons. Il s'agit par exemple d'un gain d'efficacité d'un comportement lorsqu'un certain module est activé, ou du nombre d'agents pouvant être simulés en parallèle sans ralentissement de la simulation étant donné le contexte interne (l'environnement utilisé, les modules de haut-niveau actifs, etc.) et externe (principalement la machine utilisée). D'autres thèses par contre ne peuvent recevoir de validation que subjective, étant donné la nature de ce qui doit être obtenu. L'exemple le plus important étant la volonté d'obtention de comportements *crédibles*, la crédibilité étant considérée comme totalement dépendante de l'observateur. Pour valider la crédibilité des comportements simulés, il sera donc nécessaire de recourir à des expérimentations utilisateurs, dans lesquelles les participants indiqueront leur observations et appréciations.

6.1.2 Objectifs et organisation

Dans ce chapitre nous allons vérifier que les 4 objectifs que nous nous étions fixés sont respectés, à savoir :

- que l'architecture proposée est générique,
- qu'elle possède un processus décisionnel flexible,
- que les comportements exhibés par les agents sont crédibles,
- que le système dans son ensemble est capable de passer à l'échelle.

6.1.2.1 Généricité de l'architecture

L'architecture d'agents proposée est-elle suffisamment générique pour que des modélisateurs ne la connaissant pas, puissent y intégrer leurs propres modules de haut-niveau, et les tester dans leurs propres simulations ?

La seule manière de valider ce point, est de faire en sorte que des modélisateurs externes au projet Terra Dynamica, et ne connaissant pas l'architecture, développent des modules de haut-niveau et les intègrent dans des simulations.

Dans le même temps, nous allons nous intéresser à la validation des différents modules de haut-niveau que nous avons développés. En effet, avant de nous intéresser à la suite du processus décisionnel, c'est-à-dire les processus internes au module décisionnel, il est important de valider les entrées de la décision, à savoir les propositions de comportements. Ces propositions étant envoyées par les différents modules de haut-niveau que nous avons développés, nous allons commencer par vérifier qu'ils répondent bien à leurs objectifs.

Cependant, ces modules ne sont pas au cœur du travail, et ne font pas partie des contributions de la thèse. De simples validations de principe seront donc

apportées ici, sans entrer dans le détail de leur fonctionnement.

6.1.2.2 Flexibilité du processus décisionnel

Maintenant que les entrées du module décisionnel ont été vérifiées, il est possible d'étudier son fonctionnement interne. La caractéristique principale que nous cherchons à vérifier est la capacité de ce module à réaliser une composition de comportement. Il faut en plus que les comportements produits en sortie soient cohérents, et qu'ils soient issus d'une collaboration entre l'ensemble des modules de haut-niveau.

Pour cela nous regarderons en détail comment notre système se situe par rapport aux critères de Tyrrell [Tyrrell 1993a], en nous intéressant tout particulièrement à la capacité de faire des compromis entre comportements, et à sélectionner des comportements de manière opportuniste afin de gagner en efficacité.

Par ailleurs, nous vérifierons également que le module décisionnel est correctement calibré, et que les poids des modules de haut-niveau modifient convenablement la génération comportementale.

Toutes ces validations seront réalisées grâce à des expérimentations objectives, et des mesures quantitatives.

6.1.2.3 Crédibilité des comportements

Dans cette thèse nous défendons l'idée que l'ajout de capacités d'anticipation dans le processus décisionnel, augmente la crédibilité des comportements des agents. Afin de vérifier la validité de cette assertion, nous allons mener des expérimentations subjectives, visant à comparer la crédibilité des comportements entre des agents dotés de capacités d'anticipation, et d'autres qui en sont dépourvus. Nous allons également vérifier que ces capacités d'anticipation augmente l'efficacité des comportements, et essayer de trouver des liens entre efficacité et crédibilité.

Dans le même temps, nous allons calibrer le module d'anticipation au niveau de ses deux paramètres principaux : l'horizon temporel, et la fréquence du processus d'anticipation.

L'augmentation de la crédibilité des comportements des agents ne doit pas avoir un coût en temps de calcul trop important. Dans la partie suivante, nous allons vérifier que l'ajout de ces capacités d'anticipation ne nuit pas à la capacité du système à passer à l'échelle.

6.1.2.4 Passage à l'échelle

Cette question fera l'objet d'une double validation. Tout d'abord une validation objective se basant sur la mesure des temps de calcul des différents modules mis en jeu dans l'ensemble de la thèse. Ensuite, une validation subjective, portant sur

la capacité d'un observateur externe à se rendre compte de l'intelligence des agents d'une foule.

6.1.2.5 Exemples applicatifs

En parallèle de ces validations théoriques spécifiques, nous présenterons des exemples applicatifs concrets ayant été réalisés au cours du projet Terra Dynamica par différents partenaires. Ces applications, en se servant des différentes caractéristiques précédentes, permettent de réaliser une validation pratique supplémentaire.

6.1.3 Environnement utilisé

Toutes les expérimentations ont été réalisées avec la même machine tournant sur Windows XP, dotée d'un processeur Intel Xeon X5482 à 3.2 GHz et de 8 GB de RAM. Toutes les simulations ont été réalisées au sein du même environnement, une reconstitution en 3 dimensions du quartier autour de la place de la République, à Paris. Le quartier modélisé a une surface de 1.6 km^2. Environ 5000 bâtiments sont représentés, dont une centaine sont accessibles (les autres sont principalement des immeubles d'habitation, or la modélisation de leur intérieur n'est pas utile dans le cadre de simulations urbaines). Tous les commerces (magasins, restaurants, cafés, banques, etc.) du quartier sont représentés, et les agents peuvent se rendre à l'intérieur pour y effectuer des actions. Le plan de l'environnement est donné figure 6.1.

Le logiciel de visualisation utilisé est un logiciel appelé ***Thales View***, qui est une propriété de THALES. Il permet de visualiser l'environnement en 3 dimensions, et de s'y déplacer (voir figure 6.2).

En ce qui concerne les agents, plusieurs types ont été créés (des touristes, des employés de bureaux, des habitants du quartiers, des policiers, etc.) de manière à peupler de manière cohérente l'environnement. Les différents modules de haut-niveau présentés dans le chapitre 3 ont été utilisés afin de générer leurs comportements : le module de motivations (avec quelques motivations classiques, faim, soif, fatigue, ennui, hygiène), le module d'emploi du temps (indiquant par exemple le lieu et les horaires de travail de l'agent) et le module d'instinct (gérant principalement des réactions de peur face à des situations considérées comme dangereuses). Le module d'anticipation ne sera utilisé que dans des simulations particulières, et son emploi sera explicitement indiqué.

Les agents sont créés dans des **sources**, placées à des endroits stratégiques de l'environnement (bouches de métro, extrémités de rues passantes, etc.). La fréquence de leur création a été calculée au préalable, en fonction du moment de la journée, suite à des comptages effectués place de la République, à Paris. Les flux de piétons sont donc vraisemblables et similaires aux flux réels (il en va de même pour les flux de véhicules).

Nous ne donnerons pas ici le détail des différents scénarios utilisés ni les graphes

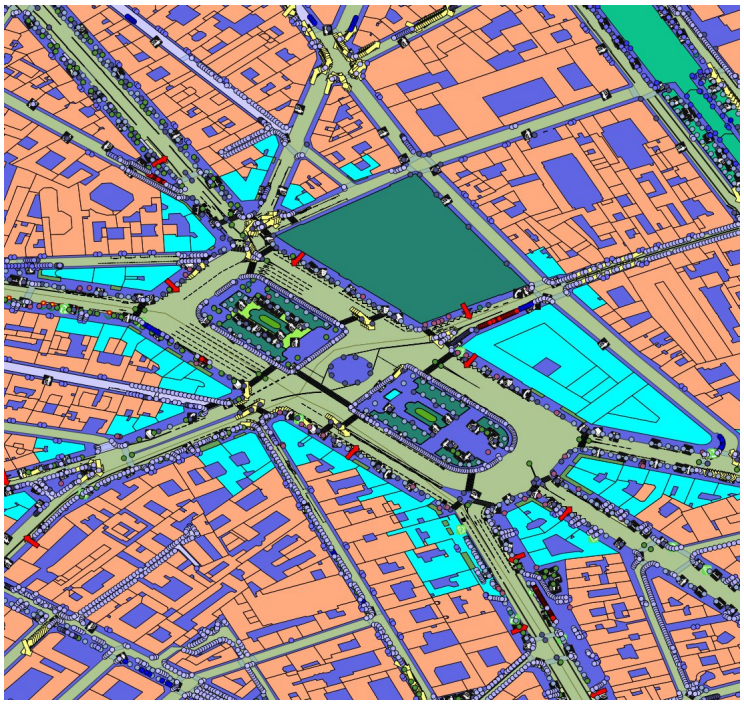

FIGURE 6.1 – Plan de l'environnement utilisé

FIGURE 6.2 – Aperçu de l'environnement via le logiciel de visualisation Thales view

comportementaux complets des différents simulations (voir annexe B pour un graphe décisionnel complet), mais voici l'ensemble des actions possibles dans l'ensemble des simulations utilisées pour les expérimentations : "*Manger dans un fast-food*", "*Manger dans un restaurant*", "*Manger un sandwich*", "*Manger chez soi*", "*Boire un café*", "*Boire un soda*", "*S'asseoir sur un banc*", "*Se reposer à la terrasse d'un café*", "*Lire*", "*Regarder la TV*", "*Acheter de la nourriture*", "*Acheter des vêtements*", "*Acheter des cigarettes*", "*Acheter des revues*", "*Acheter des médicaments*", "*Aller au wc*", "*Travailler*", "*Fuir*", "*Se protéger*", "*Éteindre feu*", "*Protéger ressource*", "*Patrouiller*", "*Jeter un objet à la poubelle*", "*Retirer de l'argent*".

6.2 Généricité de l'architecture

6.2.1 Architecture d'agents

Un de nos premiers objectifs a été de proposer une architecture décisionnelle suffisamment générique et simple à prendre en main, pour que des théories hétérogènes puissent être intégrées et implémentées en son sein par des intervenants non spécialistes.

Cette capacité a été démontrée deux fois durant le projet Terra Dynamica et le développement du modèle, sans qu'il y ait eu besoin de réaliser des expérimentations spécifiques, puisque des modules de haut-niveau ont été développés et intégrés par des développeurs externes. Il s'agit d'une part d'un modèle de comportements affectifs fondé sur le principe de conservation des ressources ([Campano 2013b]), et d'autre part d'un modèle de coordination multi-agents appliqué au problème de la patrouille ([Poulet 2012a]). Ces deux modèles ont été intégrés à l'architecture sous forme de modules de haut-niveau, développés au Laboratoire d'Informatique de Paris 6.

Les deux modèles sont fortement hétérogènes, puisque l'un gère des piles de ressources (pas obligatoirement physiques, mais pouvant également être psychologiques ou morales), et provoque des comportements de protection ou d'acquisition de ces ressources, alors que l'autre gère des déplacements et la coordination entre un nombre variable d'agents dans un groupe de patrouilleurs.

Nous avons donc bien vérifié que les interfaces mises en place étaient suffisamment explicites, bien documentées et ouvertes, pour que des développeurs externes puissent ajouter des modules à l'architecture. Nous avons vérifié également que l'architecture est suffisamment générique pour intégrer des modules fortement hétérogènes. Mais il est également nécessaire de vérifier que les comportements proposés par ces modules s'intègrent de manière cohérente dans le processus décisionnel, et que le comportement de sortie bénéficie bien de la coopération de l'ensemble des modules actifs.

6.2.2 Évaluations de principe sur les modules de haut-niveau réactifs

Avant de se lancer dans une validation du module décisionnel, il est important de vérifier que les différents modules réactifs de haut-niveaux implémentés répondent bien de manière correcte par rapport à nos attentes et qu'ils sont correctement équilibrés les uns par rapport aux autres.

Pour cela, nous lançons une simulation se déroulant sur une journée "complète" (de 8h à 18h). Nous y étudions le comportement de 100 agents, dotés des 3 modules réactifs de base : le module motivationnel, le module d'emploi du temps et le module d'instinct. Pour tous les agents, les 3 modules ont des poids identiques les uns aux autres.

Pour l'ensemble de ces modules, nous nous contentons de validations de principes, sans lancer d'expérimentations très complexes.

6.2.2.1 Module motivationnel

Pour valider ce module, nous vérifions que les différentes motivations prises en compte évoluent de manière cohérente au cours d'une journée, et que les propositions de comportement sont bien proportionnelles aux valeurs internes.

Pour cela, nous comptons le nombre de fois qu'un agent exécute une action liée à l'une des 5 motivations utilisées dans les simulations. Les résultats apparaissent dans la figure 6.3.

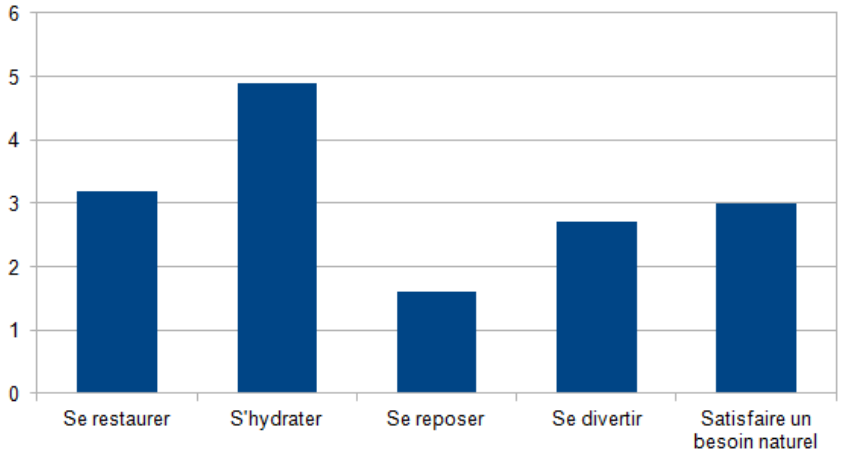

FIGURE 6.3 – Graphique présentant le nombre moyen de comportements liés à chacune des motivations dans la journée

Nous ne rentrerons pas plus dans le détail de ces comportements, nous nous contentons de trouver *crédible* que les agents mangent ou grignotent en moyenne 3 fois entre 8h et 18h, s'hydratent (pauses cafés incluses) environ 5 fois et aillent

aux toilettes 3 fois. Cette vérification n'a rien de formel, mais suffira dans le cadre de ces expérimentations.

Par ailleurs, nous regardons combien de temps les agents ont passé dans les différentes zones motivationnelles (confort, tolérance, et danger).

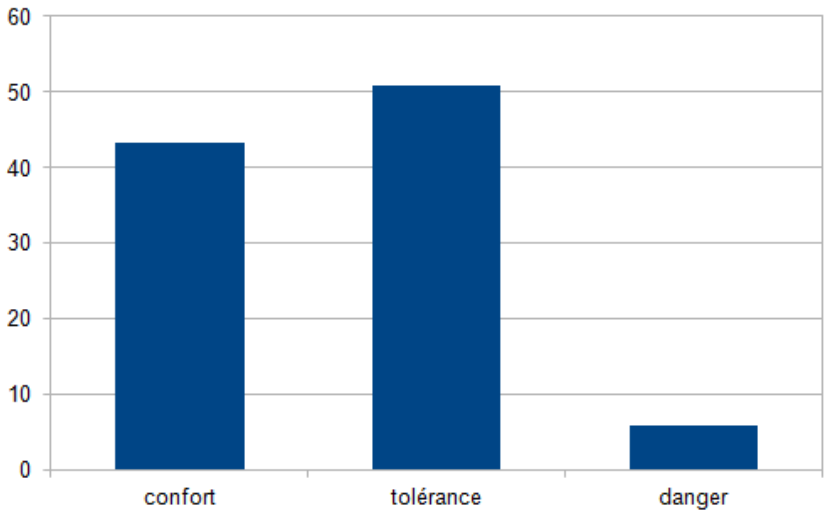

FIGURE 6.4 – Graphique présentant les pourcentages de temps passé dans les zones motivationnelles

Ces moyennes, présentées dans la figure 6.4, nous semblent également correspondre aux objectifs fixés : les agents passent très peu de temps dans la zone de danger (ce qui semble normal, sauf cas particuliers, les agents sont toujours capables de rester dans des zones de bien-être relatif) et se répartissent de manière quasi équivalente entre les zones de confort et de tolérance.

6.2.2.2 Module d'emploi du temps

En ce qui concerne l'emploi du temps, nous vérifions uniquement que les horaires de travail sont globalement respectés.

En moyenne, les agents ont travaillé 6 heures et 22 minutes sur les 7 heures prévues dans leur emploi du temps. Les 38 minutes proviennent principalement de deux causes. Premièrement, sans module d'anticipation les agents sont incapables de prévoir le temps de trajet nécessaire pour se rendre sur leur lieu de travail, ils arrivent donc systématiquement avec un léger retard. Deuxièmement, les agents interrompent de temps en temps leur travail pour réaliser de petites actions motivationnelles, comme aller au toilettes, ou boire un café (voire lire une revue, s'ils s'ennuient trop !).

Compte tenu des circonstances, un tel respect des horaires nous semble convenable, tout en sachant qu'avec l'intégration d'un module d'anticipation, les résultats devraient s'améliorer.

6.2.2.3 Module d'instinct

Le module d'instinct est tellement simple qu'une validation semble superflue. En effet, ce module ne fait que proposer un comportement doté d'une priorité fixe, à chaque fois qu'un certain événement est détecté. La seule validation que nous avons effectuée a été de vérifier que les priorités de ces propositions de comportement étaient calibrées de manière cohérente, c'est-à-dire qu'elles étaient suffisamment importantes pour prendre le pas, dans quasiment tous les cas, sur les autres propositions venant des autres modules (une réaction instinctive n'est pas sensée pouvoir être évitée).

Or il se trouve que le taux de sélection des comportements instinctifs dépasse 95%. Nous considérons donc ce module comme validé.

6.3 Flexibilité du processus décisionnel

6.3.1 Introduction

Maintenant que nous avons validé les différents modules de haut-niveau développés et intégrés à l'architecture, il devient possible de valider le module décisionnel. Plusieurs composantes peuvent être validées, et à ce titre les caractéristiques proposées par Tyrrell (voir 2.3.1) sont intéressantes à étudier :

1. Gérer tous les types de comportements.
2. Persistance des actions.
3. Priorité des comportements proportionnelle aux états (internes et externes) courants.
4. Préférence des actions consommatoires par rapport aux actions appétitives.
5. Concurrence équilibrée entre comportements.
6. Persistance du comportement courant.
7. Opportunisme.
8. Pas de système en *Winner-take-all*.
9. Combinaison des préférences.
10. Compromis.
11. Combinaison flexible des stimuli.

6.3.2 Expérimentation générale

Nous lançons une expérimentation générale, avec un scénario complet et 100 agents dotés de l'ensemble des modules de haut-niveau existants (motivations,

emploi du temps, instinct, coordination et comportements affectifs), et nous allons étudier les différents comportements enregistrés afin de vérifier l'ensemble des points exposés précédemment.

Au bout de 10 heures dans la simulation, nous comptons le nombre de fois où chaque action a été terminée au cours de la simulation. Cela nous permet de tracer le diagramme 6.5 qui présente la répartition des activités des agents dans la simulation. Les barres représentent le temps moyen qu'un agent a passé à réaliser l'action correspondante pendant la journée (l'échelle est en minutes). Le chiffre indiqué au sommet de la barre indique le nombre moyen de fois que cette action a été réalisée par un agent durant la simulation.

Notons que sur ce graphique, l'action "travailler" n'a pas été rapportée pour des raisons de lisibilité. En effet, les agents travaillent environ 276 minutes (4 heures et 36 minutes). Rajouter une barre de cette taille dans le graphique aurait pour effet d'écraser les actions d'une durée très faible, nous avons donc préféré ne pas l'afficher.

Ce graphique permet de reconstituer une journée standard d'un agent moyen, il mange environ 3 fois par jour, travaille 4h36, boit un peu plus d'un café par jour, regarde la télé un peu plus d'une demi-heure, etc.

Avant toute autre chose, cela indique que notre module décisionnel est capable de traiter le **premier point** évoqué par Tyrrell : gérer tous les types de comportements (ou du moins tous ceux que nous avons essayé de prendre en compte dans nos diverses simulations).

Par ailleurs, durant cette heure de simulation, nous enregistrons également toutes les actions interrompues avant leur terme, c'est-à-dire à chaque fois qu'un agent décide de ne pas terminer l'action qu'il a en cours, quelle qu'en soit la raison. Au final, 91.5% des actions se sont déroulées jusqu'à leur terme, ce qui confirme le **deuxième point** : les agents ont tendance à persister dans leur action courante, même une fois que les objectifs associés à l'action ont été remplis (afin d'éviter de devoir sélectionner à nouveau cette action peu de temps après).

Vu la complexité du processus décisionnel et le nombre des paramètres pris en compte, il est difficile de comparer directement la priorité d'un comportement avec les états (internes et externes) courants qui lui sont liés (**troisième point**). Nous allons donc uniquement comparer le nombre de fois que le comportement de sortie de l'agent est dicté par la proposition de comportement prioritaire (la proposition de comportement dotée de la plus grande priorité, une fois le modificateur de poids du module appliqué).

Dans environ 63% des cas, le comportement d'un agent découle de la proposition de comportement prioritaire, dans 23% de la deuxième, et dans 9% des cas de la troisième. Évidemment, ces proportions ne sont pas suffisantes à valider ce point, mais indiquent toutefois une nette préférence pour les comportements les

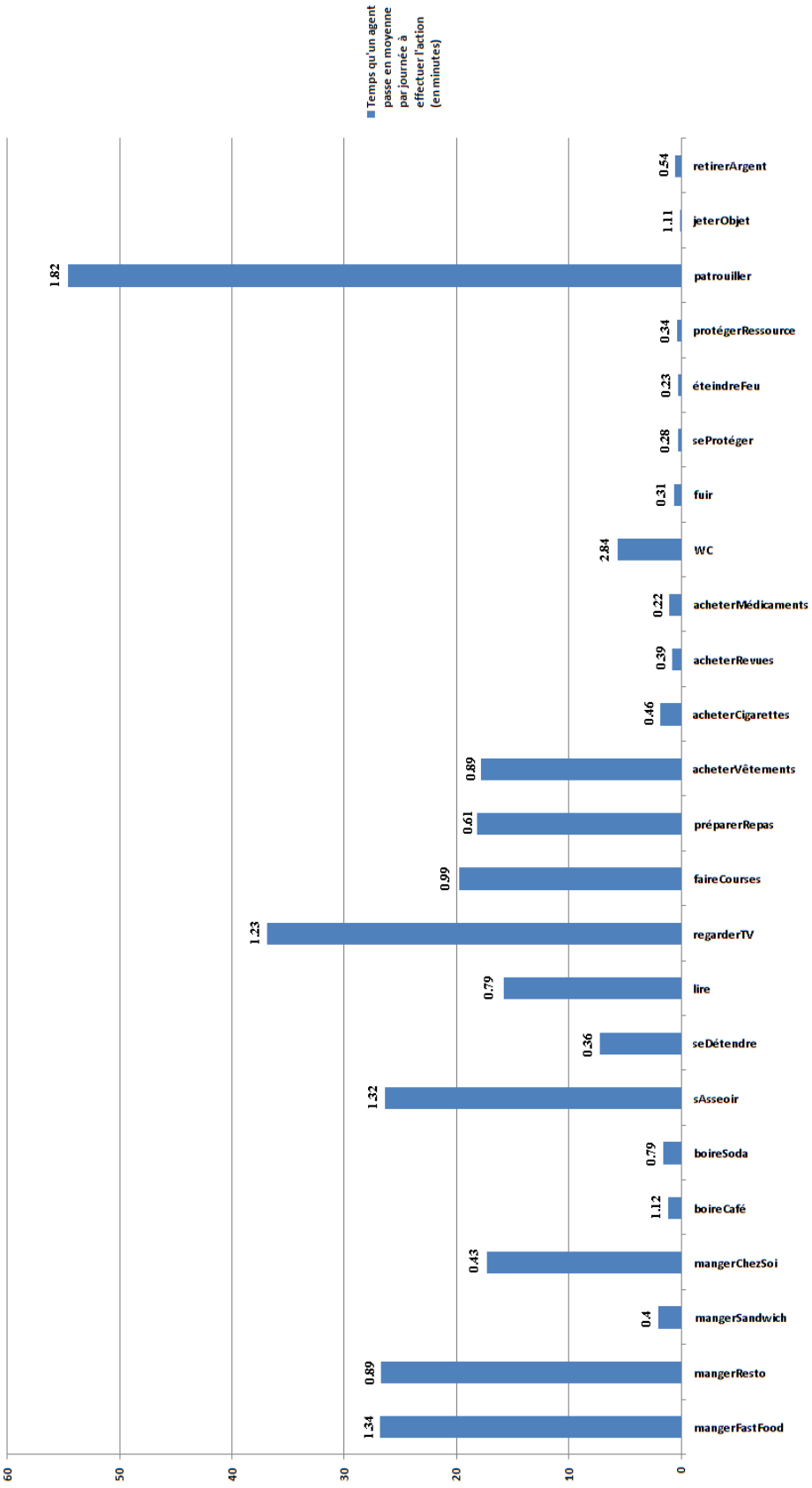

FIGURE 6.5 – Graphique présentant la répartition des activités des agents durant une journée. Les barres indiquent les temps moyens qu'un agent met à exécuter l'action correspondante. Et les nombres au-dessus des barres indiquent le nombre d'exécutions de l'action par journée.

plus prioritaires, tout en laissant une part non négligeable du choix reposer sur le contexte courant des agents. Il est vraiment difficile de calibrer le modèle sur ce genre de points autrement qu'en se fiant à nos impressions. Dans le cas présents, ces valeurs nous paraissent cohérentes.

Le **quatrième point** est un peu en marge de notre modèle. En effet, notre modèle d'action ne prend pas en compte les actions appétitives. Les actions de déplacement n'apparaissent pas directement puisqu'elles sont directement prises en charge par le module de déplacement. Le système préfère donc les actions consommatoires aux actions appétitives de manière totalement explicite : l'action "se déplacer en direction du restaurant" est masquée dans le comportement "manger au restaurant". Nous n'irons pas plus loin dans la validation de ce point.

Le **cinquième point** revient à vérifier que les comportements sont en concurrence équitable. Même si un comportement permet de répondre à plusieurs objectifs, il ne doit pas devenir trop prioritaire par rapport aux autres comportements. Ce point fait partie des raisons pour lesquelles la fonction d'agrégation des préférences du module décisionnel est une fonction sigmoïde : elle permet de prendre en compte plusieurs propositions de comportement se rejoignant, d'obtenir une priorité de sortie plus grande que n'importe quelle priorité "entrante", tout en limitant son augmentation.

Un bon exemple de l'équilibre de cette concurrence, est que les agents continuent d'aller dans des cafés pour boire, alors qu'ils auraient pu aller dans un restaurant pour boire et manger ; ou qu'ils continuent d'aller s'asseoir sur des bancs, alors qu'ils auraient pu aller dans un café pour boire et se reposer. Le contexte courant de l'agent, ses préférences, et les caractéristiques des actions faisant partie intégrante du processus décisionnel, un comportement permettant de résoudre plusieurs buts en même temps n'est aucunement assuré d'être prioritaire, même si sa priorité augmente effectivement.

Le **sixième point** se trouve pris en compte dans notre modèle par l'existence d'un bonus de persistance (voir 4.5), qui vient ajouter une priorité virtuelle au comportement courant, afin qu'il reste prioritaire par rapport à un autre comportement à peine plus prioritaire. Ce bonus permet de limiter le problème classique de l'oscillation comportementale. Mais ce n'est pas une solution parfaite, la valeur de ce bonus doit être déterminée expérimentalement, et il ne constitue qu'un frein au problème de l'oscillation sans le résoudre entièrement.

Durant les 10 heures de simulation avec 100 agents, nous enregistrons le nombre de fois que ce bonus permet le maintien du comportement courant par rapport à un autre : 103 réalisations. Au cours d'une journée, en moyenne, tous les agents continuent une fois leur comportement courant, bien qu'un autre comportement soit devenu un peu plus prioritaire. Nous vérifions également que le problème de l'oscillation comportementale n'est pas présent dans notre simulation, en comptant le nombre de fois qu'un agent oscille (c'est-à-dire qu'il interrompt son comporte-

ment courant pour un second comportement, puis à nouveau interrompt ce second pour revenir au premier). Nous détectons 55 oscillations durant la simulation. Ce qui signifie qu'au cours d'une journée, en moyenne environ 1 agent sur 2 hésite, et revient deux fois sur une décision. Ce nombre est un peu élevé, bien qu'il ne semble pas aberrant. Mais il s'avère que pour environ la moitié de ces oscillations, la deuxième interruption n'en est pas réellement une, car le second comportement est prévu pour être interrompu en cours de réalisation (il n'a pas de durée définie). C'est typiquement le cas d'une fuite lors d'un événement angoissant : l'agent prend la fuite et interrompt sa fuite dès que sa peur est retombée. La durée prévue de l'action de fuite n'est pas réellement significative.

En ce qui concerne le **septième point**, l'opportunisme, notre modèle le prend en compte en intégrant dans la note de priorité d'un comportement le temps nécessaire à son accomplissement. Ainsi un comportement moins prioritaire qu'un autre, mais beaucoup plus rapide à exécuter sera préféré : c'est un comportement opportuniste.

Le **huitième point** quant à lui signifie l'obligation pour l'agent de continuer à prêter attention aux comportements non prioritaires, alors même qu'un comportement prioritaire s'exécute. C'est-à-dire qu'un comportement prioritaire peut être interrompu, si pour des raisons environnementales, un autre comportement devient très intéressant.

Pour vérifier la présence de tels comportements (et ainsi valider ces deux points), nous nous référerons à l'expérimentation réalisée en 6.3.4.

Le **neuvième point** est trivial à vérifier, puisque dans nos simulations, chaque action possède deux composantes de coût : le coût monétaire et le coût temporel, sur lesquels chaque agent possède des préférences propres (en plus de posséder une préférence directement liée à l'action elle-même). A chaque fois qu'un agent opère un choix, il a donc été obligé de combiner plusieurs préférences.

En ce qui concerne le compromis, le **dixième point**, il était au centre de notre problématique. Les compromis sont automatiquement pris en compte lors de la décomposition des propositions de comportement en actions élémentaires, puisque les priorités sont agrégées dans le cas d'actions de compromis. Dans les simulations cela se traduit par exemple, par le fait que les agents ont tendance, au sein de l'ensemble des comportements motivationnels, à accomplir en priorité ceux liés à la faim et la soif. En effet, ils permettent très souvent d'accomplir des compromis (typiquement, manger permet de combler la faim et la soif), alors que les autres motivations sont plus difficiles à combiner (se divertir, est bien souvent une activité à part entière). Ceci explique pourquoi les actions "manger dans un restaurant", et "manger dans un fast-food" sont si nombreuses : elles sont un compromis parfait entre les motivations de faim et de soif.

Le dernier point, tout comme les points 9 et 10, est assuré par la propagation des priorités des comportements lors de la décomposition des comportements en actions élémentaires.

6.3.3 Impact du poids des modules

Tous les agents ne possèdent pas les mêmes modules actifs. Ces modules dépendent du type de l'agent. Par ailleurs, tous les agents d'un même type ne possèdent pas la même attirance ou le même intérêt pour les comportements proposés par ces différents modules. En fonction de leur individualité, les agents attribuent des poids à chacun de leurs modules actifs. Ces poids indiquent l'importance que ce module a pour l'agent. Il est nécessaire de vérifier qu'ils modifient donc correctement le processus décisionnel, et qu'un module doté d'un poids important impacte plus fortement la décision qu'un module de faible importance.

Pour vérifier cela, nous lançons une expérimentation dans laquelle nous étudions le comportement d'agents dotés seulement d'un module motivationnel (avec les motivations habituelles) et d'un module d'emploi du temps (dans le cas présent les agents ont une patrouille à effectuer), et nous faisons varier les poids de ces deux modules. Nous formons ainsi 4 groupes, G_1, G_2, G_3, et G_4 de la manière indiquée sur le tableau 6.1.

	Poids du module motivationnel	Poids du module d'emploi du temps
G_1	0.8	0.2
G_2	0.6	0.4
G_3	0.5	0.5
G_4	0.2	0.8

TABLE 6.1 – Table présentant la répartition des poids des modules entre les différents groupes d'agents

Nous avons lancé 10 simulations différentes, dans lesquelles nous avons suivi les comportements de 10 agents de chaque groupe. Pour chaque agent, nous enregistrons le but qu'il complète en premier : est-ce qu'il satisfait d'abord l'ensemble de ses motivations, ou bien termine-t-il en priorité sa patrouille ? Les 400 comportements ont été reportés dans le graphique 6.6.

Ce graphique montre deux choses. La première est que les poids des modules modifient bien la décision, et ce de manière cohérente. Lorsqu'un module a un poids de 0.8 et l'autre un poids de 0.2 (groupes G_1 et G_4), les comportements du module préféré sont toujours terminés en premier ; avec un poids de 0.6 pour le module motivationnel et un poids de 0.4 pour le module d'emploi du temps (groupe G_2), environ 1 agent sur 2 termine sa patrouille avant de satisfaire ses motivations ; alors qu'avec des poids identiques de 0.5 (groupe G_3), 80% des agents terminent leur

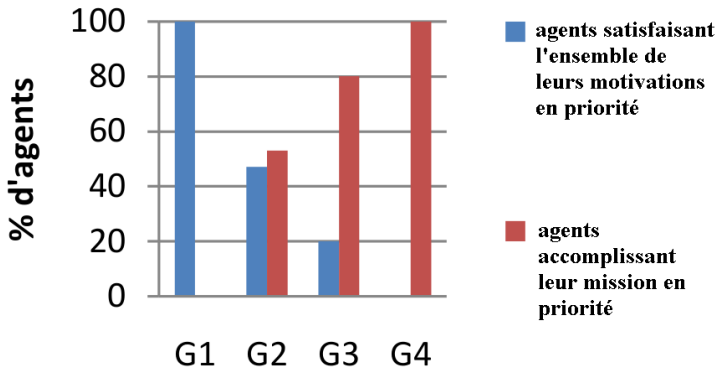

FIGURE 6.6 – Graphique présentant l'impact du poids des modules sur le comportement d'un agent

patrouille en priorité. On peut donc confirmer que plus un module a un poids élevé, plus l'agent aura tendance à suivre ses propositions de comportement en priorité.

La deuxième chose est qu'une égalité entre les poids de deux modules n'implique pas une égalité dans leurs comportements respectifs : les agents du groupe G_3, bien que possédant deux modules dotés de poids identiques, ont une très nette tendance à terminer d'abord leur patrouille (80%). A première vue, cela peut passer pour une erreur de conception : si deux modules ont la même importance pour l'agent, alors leurs comportements respectifs devraient également avoir la même importance (si des agents ont le même niveau d'intérêt pour deux choses différentes, ils devraient choisir entre elles de manière équilibrée). Mais en fait, il n'y a aucune raison d'obtenir un équilibre en sortie, et cela pour plusieurs raisons. Tout d'abord, le poids des modules n'intervient qu'au niveau des propositions de comportements. Ce sont donc les propositions des modules qui ont un poids similaire, et non les comportements de sortie, qui sont impactés par de très nombreux autres facteurs (préférences des agents, coûts, etc.). Par ailleurs, rien n'indique que les modules de haut-niveau envoient des priorités de comportements similaires. Il est tout à fait possible d'imaginer un module envoyant des propositions dotées de priorités très fortes, et un autre n'envoyant que des propositions peu prioritaires. Dans ce cas, même si le deuxième module possède un poids plus important, ses comportements resteraient peu prioritaires par rapport au premier. Nous avons pris le parti de considérer les modules de haut-niveau comme des boîtes noires afin de gagner en généricité, les poids des modules ne peuvent donc s'appliquer qu'aux sorties de ce module, afin de moduler leur importance. L'équilibrage entre modules est discuté dans la partie 4.3.1.1.

Dans le cas précis du scénario de l'expérimentation, les raisons poussant les agents à préférer les comportements liés à leur emploi du temps plutôt qu'à leurs motivations dépendent du coût des actions. Ici, les actions liées aux comportements motivationnels ont un prix, alors que les comportements liés à la patrouille sont

gratuits (il s'agit de simples déplacements).

6.3.4 Capacité de faire des compromis et de gagner en efficacité de manière opportuniste

La décision traite des comportements hétérogènes venant de modules différents. Si elle n'est pas capable de réaliser des compromis entre ces comportements, le comportement de sortie des agents ne sera ni efficace, ni crédible. Il est important que la décision soit capable de coupler les comportements.

Afin de vérifier cela, nous réalisons une expérimentation dans laquelle nous comparons les comportements de 3 groupes d'agents :

- des agents uniquement dotés de **motivations** (MO),
- des agents uniquement dotés d'un module d'emploi du temps, leur demandant d'effectuer une **mission**, c'est-à-dire de patrouiller dans la zone (groupe MI),
- des agents dotés de motivations et du module d'emploi du temps (MO + MI).

Nous réalisons 10 simulations regroupant 10 agents de chaque type.

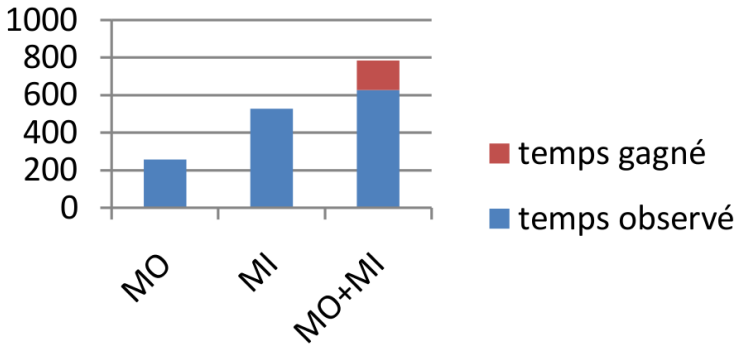

FIGURE 6.7 – Graphique montrant les temps (en secondes) mis par les agents pour remplir leurs objectifs en fonction de leurs modules actifs

La figure 6.7 montre les durées moyennes mis par les agents des 3 groupes pour satisfaire l'ensemble de leurs buts (niveau d'insatisfaction inférieur à 0.2). Les agents dotés uniquement de motivations ont mis en moyenne 250 secondes à satisfaire leurs motivations. Les agents ayant uniquement la patrouille à effectuer mettent 525 secondes en moyenne (ce qui consiste à faire le tour du quartier à pied). Les agents devant à la fois satisfaire leurs motivations et effectuer la patrouille mettent 625 secondes, ce qui est bien inférieur à la somme des deux temps précédents (250+525 = 775 secondes). Or le temps de patrouille est incompressible (quoi qu'il en soit, les agents doivent faire le tour du quartier), cela signifie donc que les agents dotés des 2 modules ont mis environ 100 secondes pour satisfaire leurs motivations, ce qui représente 40% du temps qu'il a fallu aux agents purement "motivationnels" pour le faire. Comme les agents n'ont pas pu gagner de temps pour réaliser les actions

motivationnelles (les actions ont des durées définies), cela implique qu'ils ont gagné du temps sur les trajets. Ils ont donc été capables d'interrompre des comportements de patrouille pour réaliser un comportement motivationnel "opportuniste".

Nous confirmons ce résultat en examinant les comportements de ces agents. Ils profitent effectivement du fait que leur patrouille les fait passer devant tel ou tel point d'intérêt (restaurant, café, magasin, etc.) pour interrompre la patrouille et satisfaire une motivation. Ces compromis fonctionnent mieux si les agents n'ont pas de trop grand déséquilibre dans les poids de leurs modules. En effet, si un module est trop prioritaire par rapport à un autre, les comportements de l'autre ont tendance à être passés sous silence, même si le contexte est favorable.

6.4 Crédibilité des comportements

6.4.1 Calibration des paramètres

Avant de s'intéresser au cœur du module d'anticipation, et de vérifier qu'il répond correctement aux objectifs fixés, il est nécessaire de discuter du calibrage des deux paramètres du processus d'anticipation : l'**horizon temporel** des comportements anticipés et la **fréquence** du processus de planification par anticipation.

6.4.1.1 Horizon temporel

Comme ce point a été soulevé dans le chapitre 5, l'horizon est un élément clef du processus d'anticipation. Nous rappelons que l'horizon temporel est la distance temporelle jusqu'à laquelle l'agent peut essayer de réaliser des prédictions. Plus l'agent essaye de se projeter loin dans le futur, et plus ses prédictions sont incertaines. A partir d'une certaine distance temporelle les prédictions effectuées ne sont plus assez précises pour pouvoir être exploitées (taux d'erreur trop élevé). Il est donc nécessaire de discuter de la valeur de cet horizon.

Nous avons opté pour une condition d'arrêt comptée en nombre d'actions anticipées dans le futur, c'est-à-dire un nombre maximal d'itération du processus décisionnel, pour les raisons évoquées en 5.3.2.

Afin de déterminer la ou les valeurs les plus pertinentes pour ce critère, nous avons décidé de vérifier la qualité des prédictions en fonction du nombre correspondant d'itérations du processus d'anticipation. Pour cela, nous avons lancé une simulation dans laquelle 10 agents sont dotés d'un module d'anticipation actif. Ces 10 agents ont lancé 10 planification par anticipation, à 10 instants différents de la simulation (à 5 minutes d'intervalle). Les horizons des processus d'anticipation étaient fixés à 5, c'est-à-dire que les agents itéraient le processus 5 fois. Lors de chaque itération de chaque processus pour chaque agent (c'est-à-dire $5 * 10 * 10 = 500$ itérations) le niveau d'insatisfaction prédit, la date d'occurrence dans le futur, et le numéro de l'itération en cours, ont été enregistrés. Ces 500 mesures ont été comparées aux valeurs réelles des niveaux d'insatisfaction des

agents lors de la date d'occurrence (c'est-à-dire que l'on a comparé les valeurs prédites et les valeurs réelles).

Exemple : Soit A un agent. Au temps $t = 5$ minutes, A lance un processus de planification par anticipation. Lors de la première itération du processus il anticipe qu'à $t = 6$ minutes et 37 secondes, son niveau d'insatisfaction sera de 0.4. Lors de la deuxième itération il anticipe un niveau d'insatisfaction de 0.52 à $t = 8$ minutes et 20 secondes. Et ainsi de suite.

A $t = 6$ minutes 37 secondes, le niveau réel d'insatisfaction de A est comparé avec la valeur prédite (0.4). Idem à $t = 8$ minutes 20.

L'écart entre la valeur prédite et la valeur réelle correspond à l'erreur de prédiction, plus cette erreur est faible, meilleure a été la prédiction.

La figure suivante (6.8) montre l'évolution de l'erreur moyenne en fonction du numéro de l'itération pendant laquelle la prédiction a été réalisée. Nous soulignons le fait que, dans le cadre de cette expérimentation, les agents ne faisaient que formuler des prédictions, le module d'anticipation n'envoyait aucune proposition de comportement, il n'avait donc aucune influence sur les comportements des agents.

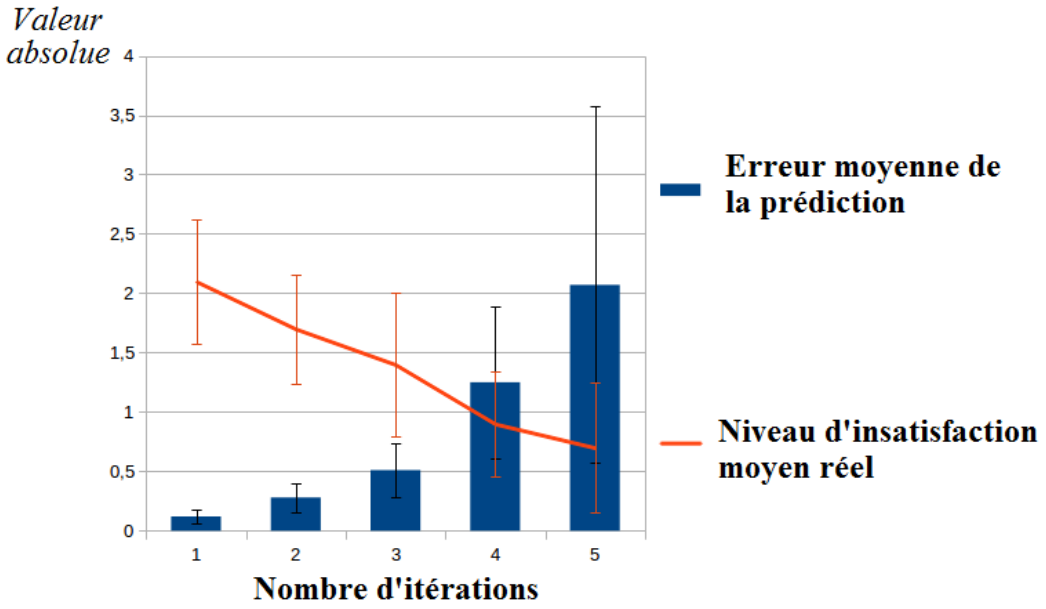

FIGURE 6.8 – Graphique présentant l'erreur moyenne de prédiction (en valeur absolue) en fonction du nombre d'itérations

Sur cette figure les barres bleues indiquent l'erreur absolue moyenne de la prédiction (avec l'écart type). L'erreur est indiquée en valeur absolue, il ne s'agit pas du pourcentage d'erreur, mais bien de l'écart entre la valeur prédite et la valeur mesurée. La courbe orange indique le niveau moyen d'insatisfaction. Ce niveau

est indiqué car il permet de mesurer l'importance de l'erreur de prédiction. Une différence de 0.5 n'a pas la même importance si le niveau d'insatisfaction est de 0.2 ou de 3.

Note : La courbe de l'insatisfaction est décroissante car les agents n'ont pas de nouveaux buts en cours de simulation. Ils se contentent d'accomplir leurs objectifs initiaux, ce qui fait décroître leur insatisfaction au fur et à mesure.

Comme attendu, l'erreur de prédiction augmente avec le nombre des itérations, avec une croissance de type exponentiel. Lors de la première itération du processus, l'erreur est négligeable, de l'ordre de 1%, mais au bout de 5 itérations, cette erreur approche les 300%. Par ailleurs, l'écart-type augmente dans les mêmes proportions.

Les raisons de telles augmentations sont multiples. Tout d'abord, lors de chaque itération le module anticipe les effets de l'action qui se termine. Or, sauf si le modèle des actions est parfait et qu'il ne possède pas de composante aléatoire, les effets ne peuvent être connus qu'approximativement. Une petite erreur liée à l'imperfection du modèle des actions est donc ajoutée lors de chaque itération. Par ailleurs, les actions de mouvement, bien que n'ayant pas d'effets, sont les plus susceptibles de rencontrer des imprévus. Les déplacements peuvent être long ; la prédiction du temps de trajet est soumise à des aléas. Mais surtout les déplacements augmentent sensiblement les probabilités d'apparition d'un événement, ou d'une perception, qui viendront modifier l'état interne de l'agent (odeur de croissants chauds qui fera augmenter la motivation de faim, vitrine d'un magasin qui donnera envie d'acheter quelque chose, etc.). De plus, les déplacements provoquent l'apparition de nouveaux comportements de compromis, qui sont difficilement détectables à l'avance (l'agent passe devant une boîte au lettres par hasard sur son chemin, et en profite pour poster une lettre qu'il avait sur lui, ou il passe devant une boulangerie et en profite pour acheter un sandwich pour plus tard, etc.).

L'ensemble des imperfections des différents modèles se cumulent lors de chaque itération, c'est pourquoi l'erreur augmente si rapidement. Mais une autre dimension encore plus grave impacte énormément le processus : c'est la *propagation* d'une erreur de décision. En effet, si le modèle de la décision se trompe dans l'anticipation de l'action suivante, tout le reste du processus devient complètement aberrant. Et la probabilité que le modèle de la décision se trompe augmente avec l'augmentation de l'erreur de prédiction, c'est un cercle vicieux puissant qui explique la forme exponentielle de la courbe d'erreur. C'est la raison pour laquelle le nombre d'itérations ne doit pas être excessif.

D'après l'étude de ce graphique, nous avons choisi de fixer l'horizon à 3 itérations. Au bout de 3 itérations, l'erreur reste acceptable, à environ 30%, alors qu'à 4 elle dépasse les 100%. 3 n'est pas un chiffre très élevé, mais il permet également de limiter fortement le coût computationnel des processus de planification par anticipation. Évidemment, un scénario différent entraîne une simulation totalement

différente. Cette expérimentation ne permet pas de déterminer le bon horizon quelle que soit la simulation, mais uniquement dans une simulation du même type. Dans le reste des expérimentations l'horizon sera fixé à 3.

6.4.1.2 Fréquence

L'autre paramètre important du processus d'anticipation est la fréquence à laquelle l'agent essaye d'anticiper. Si la fréquence est trop élevée, l'agent va perdre du temps à réaliser de nombreuses fois les mêmes prédictions, et en tirer les mêmes conclusions, alors que si la fréquence est trop faible, il pourrait passer à côté d'une anticipation intéressante. Il est donc nécessaire de se pencher sur le problème de la fréquence *idéale*.

Pour cette expérimentation nous avons pris 10 agents dotés des modules de motivations, d'emploi du temps, d'instinct, et du module d'anticipation. Nous leur avons fait effectuer un processus de planification par anticipation, chaque seconde pendant 1000 secondes. Toutes les secondes, les agents pouvaient donc formuler, ou ne pas formuler, un certain nombre de propositions de comportements. Ces propositions n'étaient pas envoyées, mais enregistrées. Pour chaque agent, nous avons enregistré le nombre de secondes entre deux variations dans les propositions de comportements.

Exemple : Soit A un agent. Au temps $t = 7$ secondes il propose le comportement C_1 avec la priorité p_1. Idem au temps $t = 8$, $t = 9$, $t = 10$. Par contre, à $t = 11$, il propose le comportement $C1$ avec une priorité p'_1. On enregistre donc qu'il lui a fallu 4 secondes avant de changer d'avis.

Ensuite pas de changement jusqu'au temps $t = 26$, date à laquelle le module propose les comportements C_1 et $C2$ avec respectivement les priorités p'_1 et $p2$. On enregistre alors qu'il a mis 15 secondes pour trouver un autre plan.

La moyenne de ces temps est de 17.55 secondes, mais avec un écart-type très important, de 13.54. Nous avons arbitrairement fixé à 10 secondes le temps entre 2 processus d'anticipation, mais ce temps est fortement dépendant du scénario, de la durée moyenne des actions, de la taille de l'environnement, etc. De plus, ce temps n'est qu'indicatif, il peut être modifié selon le type de l'agent, en fonction de la profondeur souhaitée de ses comportements, ou des ressources disponibles.

6.4.2 Gain d'efficacité des comportements

Maintenant que nous avons calibré les paramètres du processus d'anticipation, il importe de s'intéresser au différentes problématiques lui étant liées. Avant de réaliser des expérimentations sur la problématique principale (l'impact des capacités prédictives sur la crédibilité des comportements), nous allons commencer par vérifier que la présence de ces capacités permet bien d'augmenter l'efficacité des

comportements des agents.

En effet, une chose importante à vérifier concernant le module d'anticipation est qu'il permet un gain d'efficacité des comportements, c'est-à-dire une diminution du temps nécessaire à l'accomplissement des objectifs d'un agent. En effet, afin de réaliser des mesures objectives d'efficacité, il suffit de regarder la vitesse à laquelle le niveau d'insatisfaction d'un agent diminue. Plus vite il diminue, plus ses comportements sont efficaces. Pour vérifier que le module d'anticipation augmente bien l'efficacité des comportements, nous allons comparer la vitesse de diminution du niveau d'insatisfaction de deux groupes d'agents, l'un sans module d'anticipation, et l'autre avec.

Pour cette expérimentation nous allons étudier les comportements de 100 agents séparés en 2 groupes de 50. Les agents du premier groupe sont dotés d'un module de motivation, d'un module d'instinct et d'un module d'emploi du temps, et les agents du deuxième groupe sont dotés des mêmes modules, avec en plus le module d'anticipation. Les agents vont par paires. Dans chacune d'entre elles, l'un des agents appartient au premier groupe, et l'autre au second. Excepté la présence, ou l'absence, du module d'anticipation, les agents d'une même paire sont identiques : ils ont les mêmes motivations initiales, le même emploi du temps, les mêmes inventaires, la même individualité, etc. On mesure le temps qu'il faut à chaque agent pour atteindre un niveau d'insatisfaction inférieur à 0.2. Cette valeur de 0.2 n'a pas vraiment d'importance, mais nous considérons qu'en dessous de ce seuil il est possible de considérer que l'agent est pleinement satisfait.

Lorsqu'on compare les temps mis par les deux agents de chaque paire pour atteindre le palier de satisfaction, on constate que, en moyenne, les agents dotés du module d'anticipation ont mis 5.34% moins de temps que les autres (l'écart-type étant de 4.27 points. Le test de Student [Student 1908] permet de rejeter l'hypothèse nulle H_0 = "le module d'anticipation n'améliore pas l'efficacité des agents", avec une valeur critique $t = -5.01$ et un seuil de confiance de 0.95.

Ce gain de 5.34% peut sembler faible, mais il faut bien prendre en compte que les agents effectuent des actions dont la durée est incompressible. Ils ne peuvent pas gagner de temps en effectuant *mieux* ou plus rapidement leurs actions, ce n'est qu'en organisant, ou en planifiant plus efficacement l'enchaînement de leurs actions qu'ils peuvent économiser du temps. Encore une fois, ce gain est fortement dépendant du scénario, des durées des actions, des emplois du temps, etc. Par exemple un agent ayant des horaires de travail fixes à respecter, et aucun autre but, ne pourra pas gagner de temps, même en étant omniscient.

Ce gain est pourtant significatif, et prouve que le module d'anticipation augmente l'efficacité des comportements des agents. Mais est-ce que ce gain est correctement perçu par un observateur externe ? Par ailleurs, l'augmentation de l'efficacité n'est pas l'objectif ultime du module d'anticipation, qui vise surtout à augmenter la *crédibilité* des comportements.

6.4.3 Expérimentations utilisateurs : efficacité et crédibilité

6.4.3.1 Protocole des évaluations utilisateurs

Les expérimentations objectives présentées jusqu'ici ont permis de valider un certain nombre de choses, mais elles ne sont pas suffisantes pour traiter de la question de la crédibilité des comportements observés. En effet, la notion de crédibilité est entièrement subjective et dépendante de l'observateur. Afin de vérifier que la présence de capacités prédictives permet d'augmenter la crédibilité des comportements d'un agent, il est donc nécessaire d'avoir recours à des expérimentations utilisateurs.

Ces expérimentations permettront de répondre à un certain nombre de questions : est-ce que le gain d'efficacité vérifié dans la section précédente permet d'augmenter la crédibilité ? Et, pour aller encore plus loin, existe-t-il un lien entre efficacité et crédibilité d'un comportement ?

Figure 6.9 – Aperçu d'une vidéo des expérimentations utilisateurs

Pour cela, nous avons réalisé des vidéos, montrant les comportements d'un ou plusieurs agents. La figure 6.9 montre un aperçu de ces vidéos, qui indiquent également des informations neutres, aidant l'observateur à comprendre ce qui se passe, en lui donnant des informations supplémentaires utiles. Par exemple l'heure de la simulation, l'emploi du temps de l'agent, ses envies actuelles, sa personnalité, ainsi que des événements non visibles à l'écran, par exemple se déroulant à l'intérieur de bâtiments.

Après le visionnage d'une ou plusieurs vidéos, les participants devaient remplir un questionnaire en ligne. Les questions portaient essentiellement sur la notation

et la caractérisation des comportements en fonction de différents critères. L'expérimentation principale s'est déroulée en septembre 2013, avec la participation de 144 personnes. Le panel était assez diversifié, puisque 52% des participants n'avaient aucune connaissance en intelligence artificielle, 39% en avaient des notions de bases, et 9% des connaissances avancées. Lorsqu'il était demandé de donner une note à une comportement, une échelle de Likert [Allen 2007] a été utilisée. C'est une méthode de notation couramment utilisée dans les tests psychométriques, qui sert à exprimer son accord ou son désaccord par rapport à une affirmation. Nous avons choisi une échelle classique, allant de 1 à 7. Par ailleurs, les participants *pouvaient* laisser des commentaires en rapport à chaque question.

Exemple : Sur une échelle de 1 à 7, 1 représentant "pas du tout d'accord" et 7 représentant "absolument d'accord", comment noteriez-vous l'assertion suivante : "Les comportements observés me paraissent crédibles".

Dans ces expérimentations nous cherchons à vérifier les hypothèses suivantes :

- H_1 : "Les participants sont capables de détecter les comportements d'anticipation".
- H_2 : "Un observateur perçoit une augmentation de l'**efficacité** des comportements d'un agent lorsque son module d'anticipation est actif".
- H_3 : "Un observateur perçoit une augmentation de la **crédibilité** des comportements d'un agent lorsque son module d'anticipation est actif".
- H_4 : "Une augmentation de l'efficacité d'un comportement implique une augmentation de sa crédibilité, tant que le comportement n'est pas jugé *trop* efficace".

Notons que nous n'avons pas donné de définition de la crédibilité d'un comportement aux participants, c'est donc par rapport à leur propre vision de ce terme qu'ils ont répondu aux questions.

6.4.3.2 La journée de Marie

Dans cette partie, les participants n'avaient pas de vidéos à regarder, mais deux déroulements possibles d'une même journée pour un même agent (Marie) à comparer. Les participants avaient une carte de l'environnement utilisé (figure 6.10), indiquant les différents points d'intérêts (l'appartement de Marie, son lieu de travail, etc.), ainsi qu'un plan détaillé des deux journées possibles, indiquant l'ensemble des actions effectuées durant la journée, avec leurs heures de début et de fin. Les participants étaient également informés du caractère de Marie (à savoir qu'elle est consciencieuse, qu'elle adore le shopping, et qu'elle a peur de marcher seule la nuit), de son inventaire, et de son emploi du temps.

Il était demandé aux participants d'indiquer lequel des deux déroulements leur semblait le plus efficace et lequel paraissait le plus crédible (pas d'échelle de Likert

dans ces questions, uniquement un choix binaire).

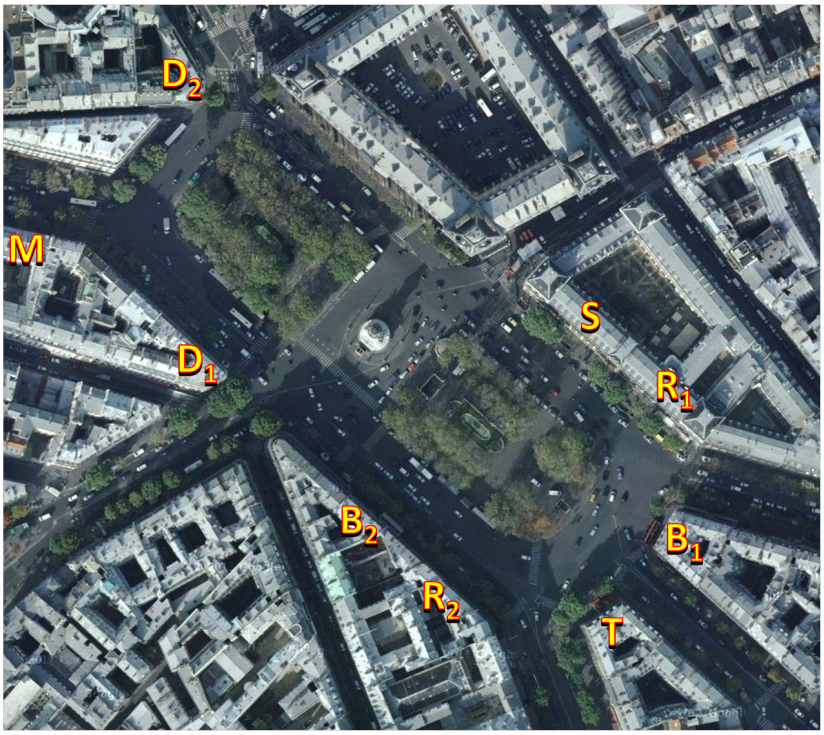

FIGURE 6.10 – Aperçu de l'environnement utilisé pour l'étude de la journée de Marie

Les deux déroulements sont issus de deux simulations identiques, à l'exception de Marie, l'agent étudié, qui, dans la deuxième simulation, possède un module d'anticipation actif (les autres modules, motivations, emploi du temps, instincts sont toujours actifs).

Les résultats sont très nets : 118 participants sur 143 indiquent que le second déroulement est plus efficace. 18 indiquent ne pas voir de différence d'efficacité, et 7 trouvent le premier plus efficace.

Les résultats sont similaires en ce qui concerne la crédibilité : 101 participants considèrent que le second déroulement est plus crédible que le premier, 26 ne voient pas de différence entre les deux, et 16 trouvent le premier plus crédible.

Ces résultats confirment H_2 et H_3. Des tests binomiaux de Student le confirment, respectivement avec $z = 8.36$ et $p < 0.001$ et $z = 9.83$ et $p < 0.001$.

Grâce aux commentaires laissés par les participants, nous sommes capables de déterminer les éléments majeurs les ayant incités à préférer le second déroulement :

- Marie quitte son appartement plus tôt, ce qui lui permet d'arriver au travail

à l'heure.

- Elle retire de l'argent pendant qu'elle se rend au travail, parce qu'elle passe devant un distributeur, bien qu'elle n'en ait pas un besoin immédiat.
- Elle choisit d'aller boire un verre avec ses amies dans un bar tout près de chez elle, bien que ce soit plus loin de son travail, parce qu'elle n'a pas envie de faire trop de marche seule en rentrant chez elle tard le soir.

Ces trois comportements particuliers sont tous issus de propositions de comportement originaires du module d'anticipation. Dans le premier cas, Marie a utilisé son modèle de l'environnement afin de déterminer une estimation de son temps de trajet, et ainsi, a pu partir en avance et arriver à l'heure. Dans le deuxième cas, le processus de planification par anticipation a détecté que dans le futur, Marie aurait besoin d'argent. Il a donc envoyé une proposition de comportement portant sur le retrait d'argent. Or comme l'agent passait juste devant un distributeur, le module de décision en a profité pour exécuter ce comportement de manière opportuniste. Dans le troisième cas, le processus de planification a permis de rajouter dans la balance décisionnelle la répulsion de Marie vis-à-vis de la marche seule la nuit, ce qui a augmenté l'intérêt du bar proche de chez elle.

En analysant les commentaires laissés par les participants, on s'aperçoit que beaucoup estiment le comportement de Marie "plus optimisé", "plus proche de la description faite de son caractère", "plus planifié", "plus logique" et qu'il "fait preuve de capacités d'anticipation" dans le second scénario. Ces commentaires vont dans le sens de H_1.

Cependant, un petit nombre de commentaires expriment un sentiment très différent. Pour certains (parmi le petit nombre de votes en faveur du premier déroulement), le comportement de Marie dans le deuxième déroulement est "trop optimisé", "nécessite de trop grandes capacités d'anticipation". Certains comparent son comportement à celui d'un "robot".

Cela nous laisse penser qu'il existe un lien entre l'efficacité perçue d'un comportement et la crédibilité qui lui est attribuée, et qu'une augmentation de l'efficacité d'un comportement contribue à augmenter sa crédibilité, jusqu'à un palier à partir duquel augmenter l'efficacité ne permet plus d'augmenter sa crédibilité. Nous allons tenter de valider cette hypothèse (H_4) dans l'expérimentation suivante.

6.4.3.3 La journée de Bob

Pour cette expérimentation, les participants visionnaient deux séquences vidéo couvrant une plage temporelle allant de 7h30 à midi, pendant laquelle les agents devaient partir de chez eux pour aller travailler.

De la même manière que dans la partie précédente, dans une des séquences, l'agent possédait un module d'anticipation actif, et pas dans l'autre (dans les deux cas, les agents doivent suivre leur emploi du temps, et prendre en compte

quelques motivations de base). Cette fois, les participants devaient donner des notes d'efficacité et de crédibilité aux comportements sur une échelle de Likert allant de 1 (comportement pas du tout efficace/crédible) à 7 (comportement parfaitement efficace/crédible).

Voici les résultats concernant l'efficacité (144 réponses) :

	Moyenne	Écart-type	Mode
Sans anticipation	**3.69**	1.60	4
Avec anticipation	**4.64**	1.74	6

TABLE 6.2 – Table présentant les scores d'efficacité des comportements donnés par les participants sur une échelle de Likert de 1 à 7

Et voici les résultats concernant la crédibilité (144 réponses) :

	Moyenne	Écart-type	Mode
Sans anticipation	**4.44**	1.53	5
Avec anticipation	**4.96**	1.65	6

TABLE 6.3 – Table présentant les scores de crédibilité des comportements donnés par les participants sur une échelle de Likert de 1 à 7

Remarque : On rappelle que le "mode" indique le choix ayant reçu le plus de réponses.

Ces deux tables montrent un net gain d'efficacité et de crédibilité lorsque le module d'anticipation est actif : la moyenne d'efficacité est augmentée de quasiment 1 point (alors que le mode augmente de 2), et la moyenne de crédibilité de 0.5 points (en parallèle d'une augmentation du mode de 1). Ces gains peuvent sembler limités, mais il faut bien se souvenir qu'ils s'appliquent sur une échelle allant de 1 à 7. Les tests de Student confirment que ces résultats sont significatifs (avec des marges d'erreurs inférieures à 0.01). Nous pouvons donc valider définitivement H_2 et H_3.

Par ailleurs ces résultats nous permettant de tracer une courbe présentant la crédibilité moyenne d'un comportement en fonction de son efficacité (figure 6.11). Pour la tracer, nous rassemblons les résultats des deux vidéos précédentes, ce qui nous donne deux fois 144 couples de scores (efficacité, crédibilité), puisque pour

chaque vidéo les participants devaient indiquer un score d'efficacité et un score de crédibilité. A partir de ces 288 couples de valeurs, nous calculons la moyenne des scores de crédibilité associés à chaque valeur d'efficacité, c'est-à-dire que nous faisons la moyenne de tous les scores de crédibilité appartenant à un couple dont le score d'efficacité est de 1, puis la moyenne de tous les scores de crédibilité appartenant à un couple dont le score d'efficacité est de 2, et ainsi de suite. La figure 6.11 présente ces résultats, les nombres au-dessus des barres indiquent le nombre de résultats pour ce score d'efficacité (le 23 de la première colonne indique qu'il y avait 23 couples de résultats dont le score d'efficacité était de 1. Juste en-dessous de ces nombres, le segment vertical indique l'écart-type.

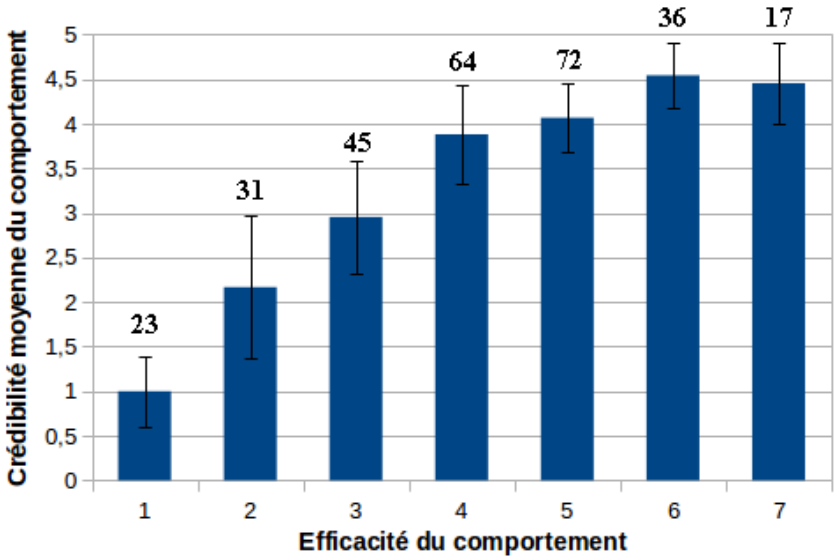

FIGURE 6.11 – Courbe de la crédibilité moyenne d'un comportement en fonction de son efficacité. Le segment indique l'écart-type, et le nombre au-dessus des barres indique le nombre de résultats.

Une telle courbe permet de montrer le lien entre l'efficacité d'un comportement et sa crédibilité en prenant en compte le caractère subjectif de ces deux notions. C'est pourquoi nous étudions les valeurs par couples, puisqu'au sein d'un couple le point de vue est le même (même observateur). Et nous pouvons mélanger les résultats obtenus dans plusieurs vidéos, puisqu'à chaque fois nous les étudions par paires.

Remarque : il aurait également été possible de tracer la courbe de l'efficacité moyenne en fonction de la crédibilité.

L'étude de ce diagramme permet d'inférer plusieurs remarques. Tout d'abord, dans le domaine de faible efficacité (colonnes entre 1 et 4) la courbe a une pente proche de 1, c'est-à-dire que les comportements très peu efficaces sont considérés en

moyenne comme très peu crédibles, et que chaque gain d'efficacité apporte un gain de crédibilité similaire. Puis, dans le domaine de forte efficacité (colonnes entre 4 et 7), la courbe semble atteindre un palier. Ceci signifierait qu'à partir d'une efficacité considérée comme moyenne, augmenter l'efficacité d'un comportement n'augmente plus sa crédibilité, ce qui confirmerait H_4.

Par ailleurs, une très légère baisse de crédibilité entre les efficacités 6 et 7 pourrait même indiquer le début d'une diminution plus importance de la crédibilité lorsque l'efficacité des comportements est perçue comme "trop importante". Cette tendance, si elle se vérifiait, pourrait être analogue au phénomène bien connu de l'*Uncanny Valley* [Mori 1970] (vallée dérangeante). Elle stipule que plus un robot androïde est similaire à un humain, plus ses imperfections sont dérangeantes. Les résultats sont présentés sous forme d'une courbe indiquant la familiarité ressentie par un humain à la vision d'un androïde, en fonction de la ressemblance de l'androïde avec l'Homme (voir figure 6.12). Cette courbe présente une *vallée étrange* montrant la gêne occasionnée par la vue d'un androïde non-humain ressemblant beaucoup à un véritable humain.

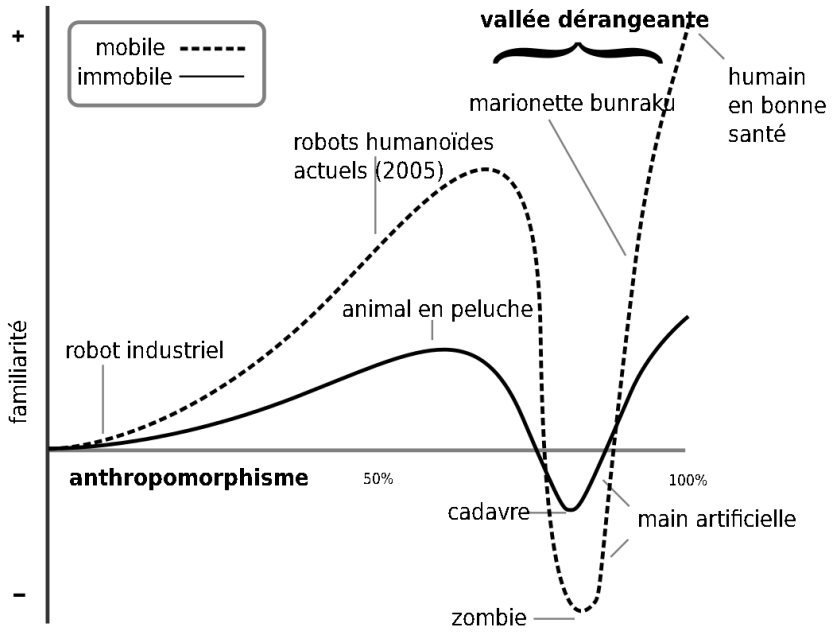

FIGURE 6.12 – Schéma de l'Uncanny Valley [Mori 1970]

Dans notre cas, il s'agirait plutôt d'un *Canny Hill*, c'est-à-dire un pic de crédibilité d'un comportement, qui devient de plus en plus crédible à mesure que son efficacité augmente, jusqu'à un palier, puis une baisse de la crédibilité en raison de son caractère *trop optimisé*, laissant de côté le côté émotionnel, imprévisible, souvent irrationnel et parfois déraisonnable des comportements humains.

Nos résultats sont insuffisants pour confirmer l'existence de ce *Canny Hill*, et la raison est simple : la séquence vidéo montrant le comportement de l'agent doté du module d'anticipation avait pour objectif d'être la plus crédible possible. Volontairement, nous n'avons donc pas poussé l'optimisation des comportements au-delà de ce qui nous semblait crédible. Or c'est justement cette zone qui manque au graphe précédent : la zone de "trop haute" efficacité". Il faudrait faire d'autres évaluations afin de clarifier l'évolution de la courbe dans cette zone.

Ces résultats sont particulièrement intéressants parce qu'ils montrent que l'optimisation des comportements n'est pas utile lorsqu'on cherche à augmenter la crédibilité. Afin de maximiser la crédibilité des comportements, il n'est pas nécessaire de maximiser la précision des modèles fournis au module d'anticipation. Au contraire, lui fournir des modèles trop précis, pourraient mener à lui rendre possible des anticipations trop précises, et donc non crédibles. Un article traitant de ce sujet a d'ailleurs été publié [Reynaud 2014].

6.5 Passage à l'échelle

6.5.1 Expérimentations objectives

Maintenant que les différentes capacités principales du modèle ont été validées, il est nécessaire de vérifier que l'architecture permet bien de passer à l'échelle, c'est-à-dire de simuler en parallèle un grand nombre d'agents dans un environnement complexe.

Pour vérifier cela, nous avons regardé le nombre d'agents simulés à partir duquel la simulation commence à ralentir. Nous sommes toujours sur la même machine, avec des cycles de 50 ms. Les résultats sont présentés dans le tableau 6.4 :

Type d'agents	Nombre maximal d'agents avant ralentissement	Temps moyen par frame (pour 500 agents) en ms
Aucun module	760	0.32
M+E+I	670	3.55
M+E+I+CA+C	630	3.76
M+E+I+A	550	4.55
M+E+I+A+CA+C	500	4.78

TABLE 6.4 – Tableau présentant les temps de calcul mis par le processus décisionnel en fonction des modules de haut-niveau actifs. M = module de motivations ; E = module d'emploi du temps ; I = module d'instinct ; A = module d'anticipation ; CA = module de comportements affectifs ; C = module de coordination

La première colonne indique quels sont les modules actifs pour l'ensemble des agents de la simulation. Dans le premier cas, les agents n'ont aucun module de

haut-niveau actifs ; basiquement ils ne font que traverser le monde sans interagir. Dans le deuxième cas, les agents sont dotés des modules de motivations, instinct, et emploi du temps. Dans le troisième cas, on ajoute les modules des comportements affectifs et de coordination. Dans le quatrième, on garde les 4 modules développés dans le cadre de la thèse : motivations, emploi du temps, instinct, et anticipation. Et dans le cinquième, l'ensemble des agents est doté de l'ensemble des modules de haut-niveau intégrés dans l'architecture.

La deuxième colonne indique combien d'agents en parallèle, il est possible de simuler, avant que la simulation ne ralentisse (c'est-à-dire que la totalité des calculs d'un pas de temps n'arrive pas à être traitée en 50 ms).

La troisième colonne indique le temps moyen mis par l'ensemble du processus décisionnel pour effectuer l'ensemble de ses traitements lors d'une frame lorsque 500 agents sont simulés en parallèle.

Les agents de la première ligne servent de "groupe témoin". Leur processus décisionnel est trivial ; ils n'ont aucun module de haut-niveau actif, donc aucune proposition de comportement, donc aucun plan à étudier. Avec ce type d'agents, il est possible de simuler (avec les conditions actuelles de simulation) 760 agents en même temps (trafic véhicule non compris). Au delà de ce nombre, la simulation commence à ralentir, en grande partie à cause des temps de traitement du module de navigation (qui dépasse les 40 ms par frame). Le processus décisionnel nécessite quand même 0.32 ms. C'est le temps qu'il lui faut pour parcourir la liste des 500 agents et vérifier qu'ils n'ont rien de mieux à faire que leur occupation actuelle (lors de la création d'un agent, un but de navigation lui est automatiquement attribué).

Les agents de la seconde ligne sont des "agents réactifs standard", ils sont dotés de l'ensemble des modules de haut-niveau réactifs développés pour l'architecture réactive de base. Avec ces agents, la simulation ralentit dès que 670 agents sont simulés. La navigation est principalement responsable de cette limitation, puisque qu'elle consomme plus de 40 ms par frame en moyenne. Les différents modules intervenant dans le processus décisionnel ont besoin de 3.55 ms en moyenne pour traiter l'ensemble des 500 agents.

Les agents de la troisième ligne sont dotés de l'ensemble des modules, excepté le module nécessitant a priori le plus de ressources, c'est-à-dire le module d'anticipation. Les moyennes de ces agents-ci se rapprochent fortement des moyennes précédentes. On constate une légère diminution du nombre d'agents simulables sans ralentissement, une quarantaine de moins, et environ 0.2 ms de plus de temps de calcul.

Les agents de la quatrième ligne sont des "agents cognitifs", puisqu'ils sont tous dotés du module d'anticipation. Notons que la fréquence du processus de planification par anticipation est réglée à 10 secondes, et l'horizon temporel à 3 itérations. Avec ce type d'agents, le simulateur n'est capable de gérer correctement que 550

agents (encore une fois, c'est la navigation qui consomme le plus de ressources, avec environ 40 ms de temps de traitement). Le processus décisionnel a besoin de 4.55 ms pour gérer les 500 agents.

Nous rappelons qu'en moyenne le module d'anticipation n'intervient que toutes les 10 secondes pour un agent, or avec des frames de 50 ms, cela correspond à 200 frames en 10 secondes. Avec 500 agents en parallèle, cela donne en moyenne 2.5 appels au module d'anticipation par frame. En considérant que l'activation du module d'anticipation n'augmente pas sensiblement le temps de traitement du module décisionnel, cela signifie que le module d'anticipation nécessite $1/2.5 = 0.4$ ms par appel.

En ce qui concerne les agents du cinquième groupe, par rapport au groupe précédent, on constate une nouvelle fois une légère diminution des performances, 50 agents de moins avant ralentissement et environ 0.3 ms de temps de calcul supplémentaire.

Dans le cadre de nos simulations, ce ne sont pas les modules intervenant dans le processus décisionnel qui sont la cause principale de la limitation du nombre maximal d'agents pouvant être simulés sans ralentissement, mais le module de navigation. Cependant, les résultats montrent clairement qu'il est facile d'économiser des ressources au niveau décisionnel, en mettant en place un niveau de détails (voir 3.5.8, 4.7, et 5.5).

Lors d'une simulation classique, c'est-à-dire avec des agents de plusieurs types, de manière à obtenir un ensemble cohérent sans surcharger le simulateur, il est possible de simuler environ 700 agents (avec un trafic routier d'environ 1300 véhicules dans la zone) avant que la simulation ne ralentisse. Nous soulignons le fait que ce nombre est atteint sans qu'aucune optimisation ne soit réalisée, c'est-à-dire que tous les calculs sont effectués dans une boucle synchrone. Avec un minimum d'optimisation, notamment une parallélisation des différentes tâches, ce nombre pourrait très facilement être dépassé. Vu la taille de l'environnement, ce nombre d'agents simulables en parallèle semble cohérent, l'objectif de passage à l'échelle est donc respecté. Par contre, nous ne sommes capables que de simuler un petit quartier de Paris, nous sommes donc encore loin de la simulation complète d'une grande ville. Certaines techniques permettraient de s'en approcher, notamment les travaux de [Navarro 2013b], couplés à notre approche de simplification de la modélisation agent en fonction de l'importance des agents.

6.5.2 Expérimentations subjectives

En parallèle de la validation relative au nombre d'agents simulables en même temps, et des temps de calculs nécessaires, nous avons également vérifié que la notion de *degré d'importance* des agents avait du sens. Nous avons aussi vérifié que la simplification de la modélisation des agents peu importants ne perturbait pas de manière significative la crédibilité de la scène étudiée par un observateur. En

effet, les expérimentations utilisateurs précédentes (voir 6.4.3) ont montré qu'un observateur est capable de distinguer les différences de comportements entre deux agents (l'un sans module d'anticipation et l'autre avec) lorsqu'il suit de près ses actes durant la journée. C'est-à-dire que si l'agent est important, la présence du module d'anticipation est importante. Mais si l'agent perd son importance, par exemple s'il est noyé au milieu d'une foule importante, est-ce qu'un observateur sera toujours capable de distinguer les différences de comportements ?

Pour répondre à cette question, nous avons proposé aux participants de notre expérimentation utilisateurs de visionner deux scènes présentant un grand nombre d'acteurs. Les deux scènes étaient "identiques", dans la mesure où elles présentaient les mêmes lieux de l'environnement, à des horaires similaires (et donc avec des foules de même type). A la suite de ces deux visionnages, il leur était demandé de dire si, dans l'une des deux scènes, les comportements des agents étaient plus crédibles que dans l'autre. Dans la première scène, l'ensemble des agents avait l'intégralité des modules de haut-niveau activés (anticipation incluse), alors que dans la deuxième, les agents n'étaient dotés que d'un module de haut-niveau entièrement aléatoire, proposant un comportement aléatoire jusqu'à son accomplissement, puis un autre, et ainsi de suite.

Les participants ont été respectivement 16.7% à répondre que la séquence 1 était plus crédible et 20.1% à indiquer la séquence 2, tandis que 63.2% d'entre eux ont répondu "aucune des deux". Ces résultats à eux seuls suffisent à montrer que la complexité décisionnelle n'intervient plus du tout de la même manière au niveau d'une foule (ou du moins, au niveau de l'observation d'un grand nombre d'agents). Ce qui va être remarqué par un observateur sera beaucoup plus le comportement macroscopique de la foule en temps qu'entité (mouvements globaux, cohésion de groupe, écoulement des flux, etc.) plutôt que les comportements microscopiques des agents qui la composent.

Par ailleurs, en étudiant les commentaires, expliquant leurs réponses, laissés par les participants ayant choisi l'une des deux séquences, on constate que les raisons pour lesquelles ils ont choisi l'une plutôt que l'autre portent toutes sur des critères non-pertinents (dans le cadre de nos travaux), c'est-à-dire soit des trajectoires incorrectes (problème de détection de chemin), soit des collisions intempestives entre agents (problème d'évitement de collision), soit des animations défectueuses (problème de rendu visuel).

Ces résultats sont très rassurants puisqu'ils indiquent que notre proposition de simplifier la modélisation des agents non importants est tout à fait valable : un observateur externe est incapable de distinguer la complexité d'un comportement s'il n'est pas capable de comprendre les raisons ayant motivé sa sélection, ou de suivre son déroulement.

6.6 Exemples applicatifs

Dans le cadre du projet Terra Dynamica, plusieurs stands ont été tenus lors du festival du numérique "Futur en Seine" en Juin 2013 à Paris. Sur ces stands ouverts au public, les différents partenaires académiques et industriels ont présenté diverses applications utilisant l'architecture d'agents proposée dans cette thèse. Toutes les applications ne se servent pas de l'ensemble des caractéristiques de l'architecture, mais elles sont toutes utilisées par plusieurs applications différentes.

Sans donner la liste exhaustive des applications développées tout au long du projet, nous allons donner plusieurs exemples représentatifs des utilisations très diversifiées qui peuvent être faites de notre architecture, dans les 4 principaux domaines d'applications visés par le projet : urbanisme, transports, jeu vidéo, et sécurité.

6.6.1 Urbanisme

Dans le domaine de l'urbanisme, une application de démonstration a été mise en œuvre en partenariat avec la Mairie de Paris. Grâce au logiciel propriétaire de visualisation de *THALES*, *Thales View*, il a été possible de faire tourner en continu sur le stand, une simulation présentant le quartier autour de la place de la République à Paris, sur un grand écran. L'ensemble des bâtiments étaient représentés, et la place était peuplée de manière réaliste (flux de piétons, de voitures, de bus, etc.). Par ailleurs, une application web, développée par *STAR-APIC*, permettait d'accéder à la simulation par internet, et de sélectionner, déplacer, créer et supprimer des agents. Dans le même temps, un agent conversationnel animé, développé par *DAVI Interactive*, était capable de répondre à des questions, telles que la position d'un agent, le nombre d'agents dans une zone donnée, etc.

FIGURE 6.13 – Capture d'écran de vues intérieures à des véhicules, avec le logiciel Thales View

La composante la plus utilisée est celle liée au passage à l'échelle, puisque de

nombreux agents étaient simulés en parallèle. Plusieurs scénarios étaient possibles, et l'un d'entre eux mettait même en scène une manifestation (avec plusieurs milliers de manifestants), mais la crédibilité des comportements était également très importante, puisqu'il était possible de suivre un agent précis de la simulation sur une longue période de temps.

FIGURE 6.14 – Capture d'écran réalisée avec le logiciel Thales View d'une manifestation simulée place de la République à Paris

6.6.2 Transports

En ce qui concerne les transports, un sas de réalité augmenté a été mis en place par le *CiTu*, qui proposait à des enfants de passer un "permis piéton", leur permettant de se former face aux dangers de la ville, circulation, véhicules prioritaires, encombrement de chaussée, etc.

Ici la généricité de l'architecture et la flexibilité du processus décisionnel étaient fortement utilisés, puisque les enfants devaient guider des passants dont les comportements étaient dangereux. De manière à ce que certains agents exhibent un tel comportement, ils étaient dotés d'un module de haut-niveau spécifique (en plus des autres modules plus "classiques" dont nous avons parlé précédemment), qui les poussaient à adopter des comportements dangereux.

6.6.3 Jeu vidéo

Deux partenaires industriels du projet étaient spécialisés dans le domaine du jeu vidéo. Ils ont tous les deux développé des jeux de démonstration mettant en valeur des aspects différents de l'architecture.

Kylotonn a proposé un jeu dans lequel le joueur doit gérer le trafic routier d'un ensemble de rues. Pour cela il doit gérer la vitesse de transition des feux

de signalisation, afin que les véhicules s'écoulent le plus vite possible à travers le quartier, sans créer d'embouteillages.

FIGURE 6.15 – Capture d'écran du jeu développé par Kylotonn

BeTomorrow, pour sa part, a développé un jeu futuriste (mais fondé sur la même base de donnée), dans lequel le joueur doit désamorcer des bombes cachées dans l'environnement avant la fin du temps imparti. Il doit également échapper aux caméras de surveillance et aux patrouilles.

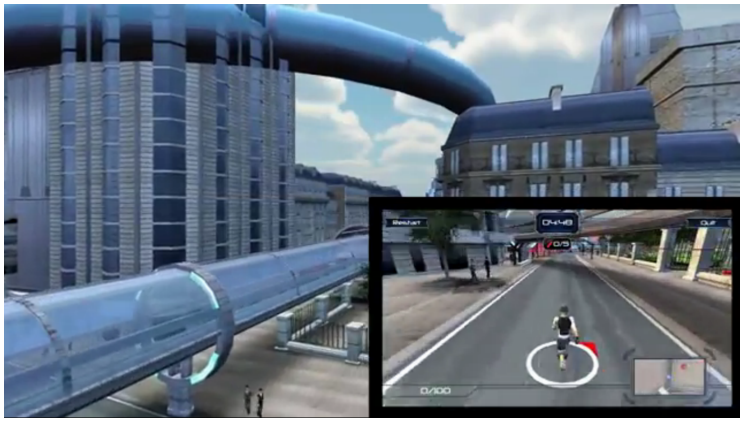

FIGURE 6.16 – Capture d'écran du jeu développé par BeTomorrow

Ces jeux font moins appel aux caractéristiques spécifiques de notre architecture, puisque les personnages non-joueurs sont peu présents, et peu importants. L'accent est donc mis ici sur la généricité de l'architecture, lui permettant d'être utilisée dans des domaines variés, pour des applications très différentes.

6.6.4 Sécurité

Le dernier domaine d'application principal, la sécurité, a fait l'objet d'une démonstration réalisée par *Thales Theresis*, en partenariat avec la Préfecture de Police de Paris. Leur simulateur est capable de simuler une ville entière, grâce à un mécanisme d'agrégation et désagrégation de foules, et permet d'obtenir des simulations très intéressantes dans des domaines comme la simulation de foule (évacuation de grands bâtiments, stations de métro, gares, etc.).

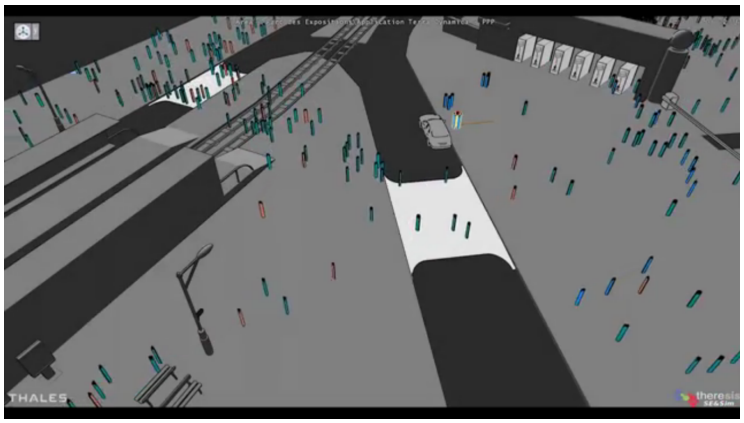

FIGURE 6.17 – Capture d'écran de l'application développée par Theresis

Cette application se focalise surtout sur le passage à l'échelle, en utilisant pleinement les avantages du niveau de détail dynamique en Intelligence Artificielle.

6.6.5 Conclusion sur les applications

A partir d'une même architecture d'agents, il a ainsi été possible de réaliser des applications dans des domaines très variés, en utilisant des capacités différentes du système. Le succès de l'ensemble de ces applications permet d'apporter une validation pratique supplémentaire de nos 4 critères principaux. Utilisée par des systèmes hétérogènes, pour accomplir des objectifs hétérogènes, notre architecture est capable de s'adapter et de répondre aux besoins spécifiques de l'application.

6.7 Conclusion sur les évaluations

Dans cette section nous avons présenté les diverses expérimentations que nous avons réalisées dans le but de valider les différentes composantes de notre modèle. Nous avons tout d'abord réalisé des pré-expérimentations en interne, afin de calibrer le modèle, les divers paramètres, ainsi que les modules de haut-niveau et leur relations.

Ensuite nous avons réalisé des expérimentations objectives, afin de vérifier les différentes thèses que nous défendons concernant l'architecture générale : sa généri-

cité et la facilité de lui intégrer de nouveaux modules. Enfin, concernant le module décisionnel, nous avons vérifié l'ensemble des points défendus par Tyrrell, ainsi que la correcte intégration des divers modules de haut-niveau dans l'architecture et sa capacité à réaliser des compromis entre des modules et des comportements hétérogènes.

Par ailleurs nous avons réalisé une expérimentation subjective, dans laquelle des observateurs humains extérieurs au projet ont donné leur avis sur différentes caractéristiques des comportements observés, notamment sur leur efficacité et crédibilité (le point crucial). De ces expérimentations nous avons pu tirer des résultats intéressants concernant les liens entre efficacité et crédibilité, et nous avons vérifié que l'ajout du module d'anticipation permet une augmentation significative de la crédibilité des comportements.

Pour finir, nous avons également vérifié que notre modèle répond aux critères liés au passage à l'échelle, qu'il peut être utilisé pour modéliser un environnement crédible (simulation de nombreux agents en parallèle au sein d'un environnement complexe), tout en gardant une grande crédibilité grâce à l'ajout du module d'anticipation. Grâce au caractère très flexible de l'architecture, capable d'activer et de désactiver des modules en cours de simulation, il est possible d'adapter la profondeur de réflexion des agents à leur importance à un moment donné, et donc d'économiser des ressources de calcul pour les agents les plus importants.

6.8 Limites et perspectives

Ces expérimentations sont intéressantes, mais de nombreuses autres pourraient être menées. Il reste à traiter certains aspects, et les expérimentations utilisateurs pourraient être plus larges afin d'essayer de valider de manière plus concrète l'existence de la *Canny Hill*. Pour cela il faudrait volontairement fournir aux agents des modèles prédictifs d'une très grande précision, afin qu'ils soient capables d'anticiper leur futur de manière quasi parfaite. C'est d'ailleurs ce qui avait été réalisé lors de pré-expérimentations visant à valider le protocole expérimental. Les commentaires des participants nous avaient alors incités à réduire la qualité des modèles prédictifs afin d'augmenter la crédibilité perçue des comportements.

Il serait également intéressant de comparer notre architecture à des architectures existantes largement utilisées, comme BDI ou SOAR par exemple. Plusieurs types de comparaisons sont envisageables. D'une part, essayer d'intégrer en tant que module de haut-niveau de notre architecture, des sous-systèmes d'une architecture BDI ou SOAR. En ce qui concerne les architectures BDI, cette intégration est simple, puisque la première étape du processus de sélection d'actions de ce type d'architecture, consiste en une génération des options comportementales et d'une sélection des intentions pertinentes. Ces intentions sont aisément assimilables à des propositions de comportements. Le même type de découpage peut être effectué pour les architectures de type SOAR, puisque les premières phases du processus décisionnel

sont : perceptions, mise à jour de la mémoire de travail, propositions d'opérateurs, évaluation des opérateurs. Il suffit de couper avant la phase de sélection pour obtenir un sous-système capable de proposer des comportements dotés d'une utilité. Une fois ces ajouts effectués, on peut vérifier que notre module décisionnel est capable de travailler avec ces nouvelles propositions de comportement, et que notre architecture peut simuler des agents dont les comportements sont issus d'une autre architecture.

D'autre part, on pourrait implémenter des agents SOAR ou BDI et les faire évoluer dans le même environnement que nos propres agents, afin de les soumettre aux mêmes tests utilisateurs portant sur la crédibilités des comportements.

En ce qui concerne le passage à l'échelle, nous n'avons pas évalué la perte de cohérence (contrairement à [Navarro 2013a]) liée à la simplification des agents les moins importants. En effet, lorsqu'un agent voit sa modélisation interne simplifiée (quelles que soient les simplifications effectuées), ses comportements sont modifiés. Même si de légères modifications du comportement d'un agent n'impactent pas fortement la simulation, chaque fois qu'un agent est simplifié, la simulation diffère par rapport à son déroulement "normal". Il aurait été intéressant de mesurer ces différences afin de vérifier qu'elle restent raisonnables.

Enfin, lors des expérimentations utilisateurs, il aurait été particulièrement intéressant d'essayer de comparer les comportements des agents de notre système avec ceux d'avatars, directement contrôlés par des humains.

Chapitre 7

Mise en œuvre dans le cadre du projet Terra Dynamica

Sommaire

Ce chapitre a pour objectif de décrire la mise en œuvre de l'architecture décrite dans les chapitres précédents, ayant été réalisée dans le cadre du projet Terra Dynamica. Nous présenterons notamment le fonctionnement, la paramétrisation, et les interfaces des différents modules développés dans le cadre de cette thèse, ainsi que leur place au sein du logiciel dans son ensemble.

7.1 Informations générales

Cette thèse s'est déroulée dans le cadre d'un accord CIFRE entre le Laboratoire d'Informatique de Paris 6 et THALES. L'ensemble des développements effectués

au cours du travail ont été effectués à *Thales Training and Simulation*, pendant un CDD de 3 ans en tant que doctorant/ingénieur informaticien. Le code est exclusivement écrit en C++, réalisé grâce à l'environnement de développement intégré pour Windows, *Visual C++*. La gestion collaborative du développement étant gérée grâce à la plate-forme intégrée *Rational ClearCase*, sur un réseau interne à THALES, exclusivement sur des ordinateurs sous Windows XP 64 bits. Le logiciel final est une propriété de THALES et du Laboratoire d'Informatique de Paris 6, non libre de droit.

7.2 Intégration de l'architecture décisionnelle au sein du logiciel

7.2.1 Organisation globale du logiciel

Avant toute chose, nous allons très rapidement présenter la structure globale du logiciel développé dans le cadre du projet Terra Dynamica.

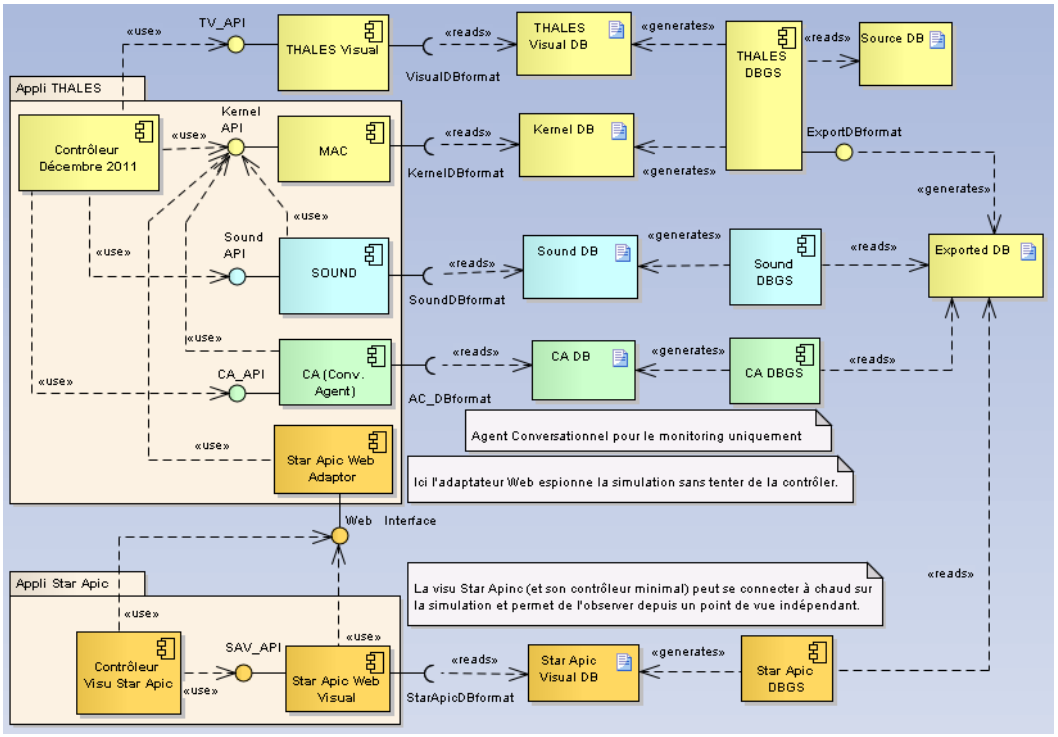

FIGURE 7.1 – Schéma de la structure globale du logiciel

Le schéma 7.1 présente la structure globale du logiciel. Le travail réalisé dans le cadre de cette thèse se trouve dans le "MAC", le Moteur d'Animation Comportemental, situé au sein de l'application THALES (cadre à gauche sur le schéma). Ce moteur d'animation comportemental, étant utilisé par l'ensemble des modules

du projet, est considéré comme son cœur. C'est pourquoi il est également appelé "Kernel". L'ensemble de ses interfaces sont incluses dans le "Kernel API". Le MAC est en relation

- d'une part avec la partie s'occupant de la visualisation (d'un côté le visualisateur Thales View, de l'autre une interface de debug permettant d'afficher dans un environnement en 2 dimensions des informations utiles pour les développeurs) ;
- d'autre part avec la partie sonorisation de l'application, à savoir un son spatialisé en 3 dimensions prenant en compte la distance entre la caméra est les différentes sources sonores de la simulation (piétons, véhicules, événements, etc.) ;
- mais également avec un agent conversationnel animé (ACA), qui permet de surveiller et de contrôler le déroulement de la simulation en lui posant des questions et en récupérant des informations en temps réel ;
- et enfin avec une interface web permettant de suivre et même d'interagir avec la simulation depuis un site internet distant.

7.2.2 Organisation des modules au sein du moteur d'animation comportemental

Le schéma 7.2 indique les différents composants du MAC.

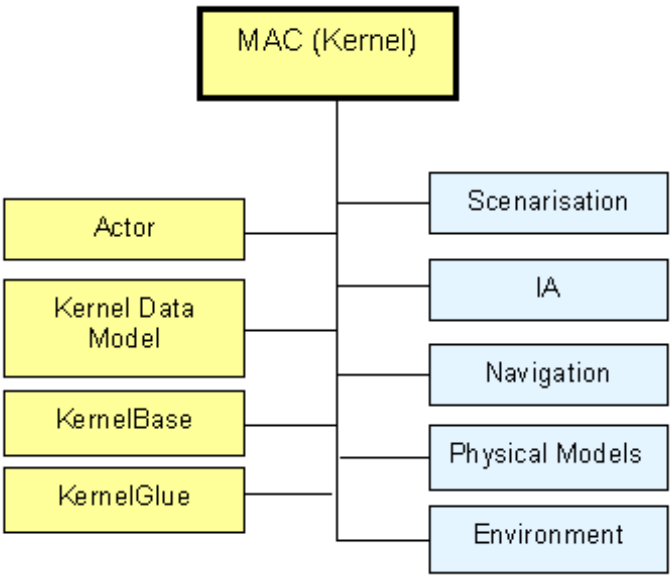

FIGURE 7.2 – Schéma de la structure interne du moteur d'animation comportemental

Le détail de ces différents composants est indiqué dans le tableau 7.1
La figure 7.3 est un diagramme de dépendance de l'architecture décisionnelle.

Module	Détails
KernelBase (KB)	Services de base du MAC
KernelGlue	Ordonnancement des modules (sauf KB et KDM)
KernelDataModel (KDM)	Partage de données et services
Actor	Création et destruction des agents virtuels
Environment	Description et sémantique de l'environnement
Physical Models	Dynamique des acteurs et positionnement
Navigation	Planification et suivi de trajectoire
IA	Génération et sélection des comportements
Scenarisation	Scénarisation scriptée et adaptative

TABLE 7.1 – Tableau présentant la liste des différents composants du MAC

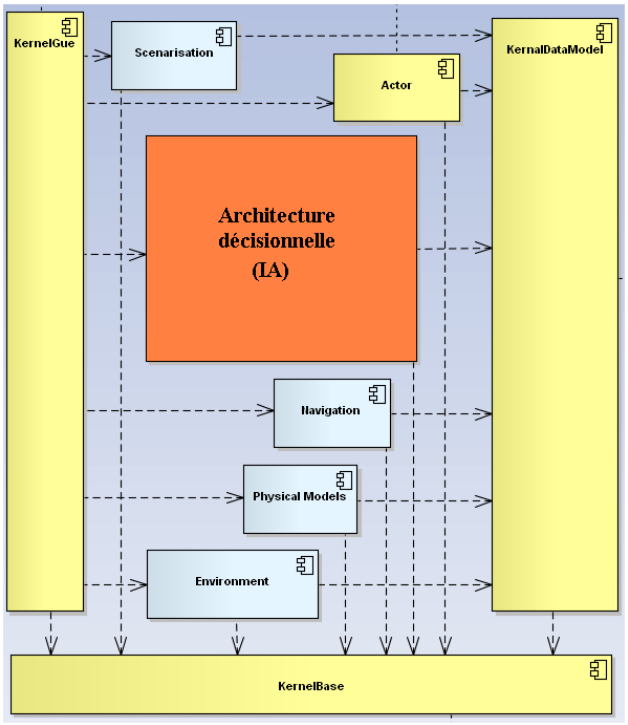

FIGURE 7.3 – Diagramme de dépendance de l'architecture décisionnelle

7.3 Interfaces entre l'architecture décisionnelle et l'architecture d'agents globale

Dans cette section nous allons décrire les différentes interfaces utilisées permettant à l'architecture décisionnelle de communiquer avec le reste de l'architecture d'agents.

7.3.1 Communication entre l'architecture décisionnelle et les autres modules de l'architecture d'agents

Le diagramme 7.4 présente les différentes communications entre l'architecture décisionnelle et le reste de l'architecture d'agents.

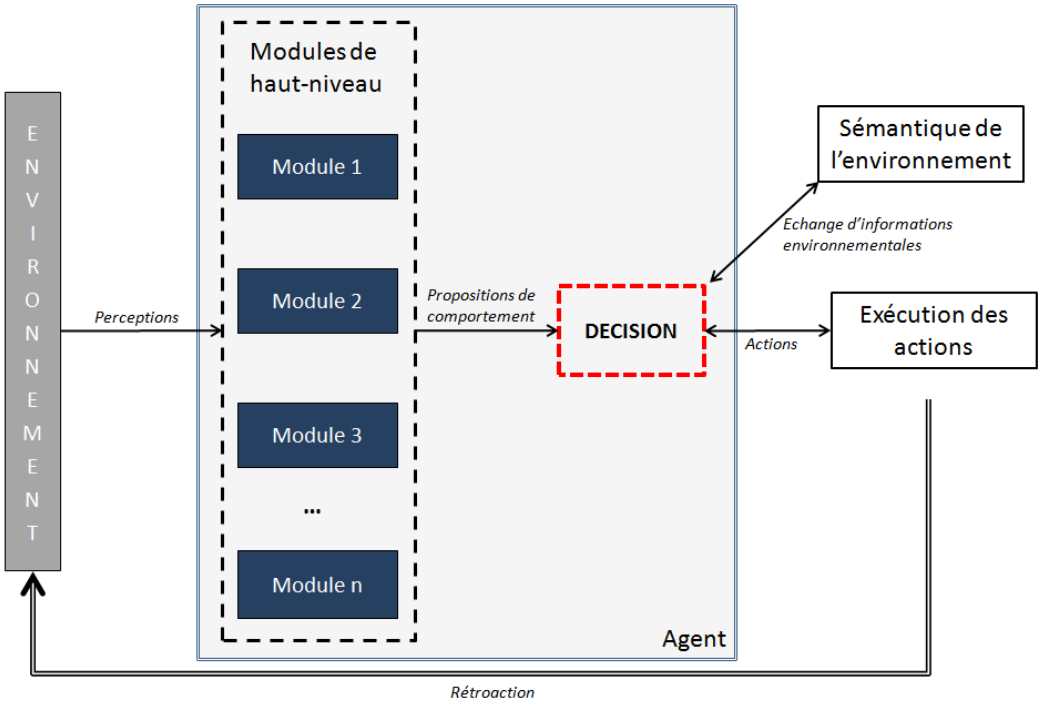

FIGURE 7.4 – Diagramme présentant les entrées et sorties de l'architecture décisionnelle

7.3.2 Perceptions

Les perceptions des agents sont gérées au niveau de l'environnement par un système d'implémentation d'une interface d'abonnement. Nous ne donnerons pas le détail de cette interface, mais au cours de la simulation un agent peut s'abonner ou se désabonner aux différents types de perceptions existantes, c'est-à-dire les différents sens humains (vue, ouïe, odorat, etc.). Ces abonnements peuvent se faire en fonction de ces modules actifs (ce qui permet de faire un pré-tri des informations non pertinentes) ou en fonction de l'importance des agents (cette piste, non explorée, permettrait de réduire la complexité des processus décisionnels en limitant l'accès des agents aux sources d'informations lorsque le comportement de ces agents n'est pas pertinent). C'est donc l'environnement qui se charge de déterminer lorsqu'un agent est à portée suffisante pour recevoir une nouvelle perception, et à partir de quand il ne la reçoit plus.

Un exemple de ce fichier de paramètres ("Perceptions.xml") décrivant les perceptions existantes dans une simulation est donné dans l'annexe A.6.

7.3.3 Exécution des actions

Le module d'exécution des actions est paramétré de la même manière que les modules de haut-niveau, c'est-à-dire qu'il tire ses informations d'un fichier de paramètres (ActionsExecution.xml").

Ce fichier doit permettre au module d'exécution des actions de déterminer les animations qu'il faut lancer lorsque l'agent commence une action et la termine. Dans le cas de nos simulations, nous avons séparé ces informations en deux, d'une part la posture de l'agent (est-il debout, assis, en train de marcher, ou de courir ?) et d'autre part son activité (est-il immobile, en train d'attendre, en train de manipuler un objet, de jeter quelque chose, de prendre quelque chose dans sa poche, etc. ?). A partir de la combinaison de ces deux informations, il est possible de déterminer l'animation correspondante (l'agent marche en mettant la main dans sa poche, il est debout et jette un objet, il est assis et discute, etc.).

Un exemple de ce fichier est donné dans l'annexe A.5.

Chaque fois qu'un agent décide d'exécuter une action, il implémente une interface (voir figure 7.5) permettant au module d'exécution de prendre le contrôle de son déroulement, et de le surveiller.

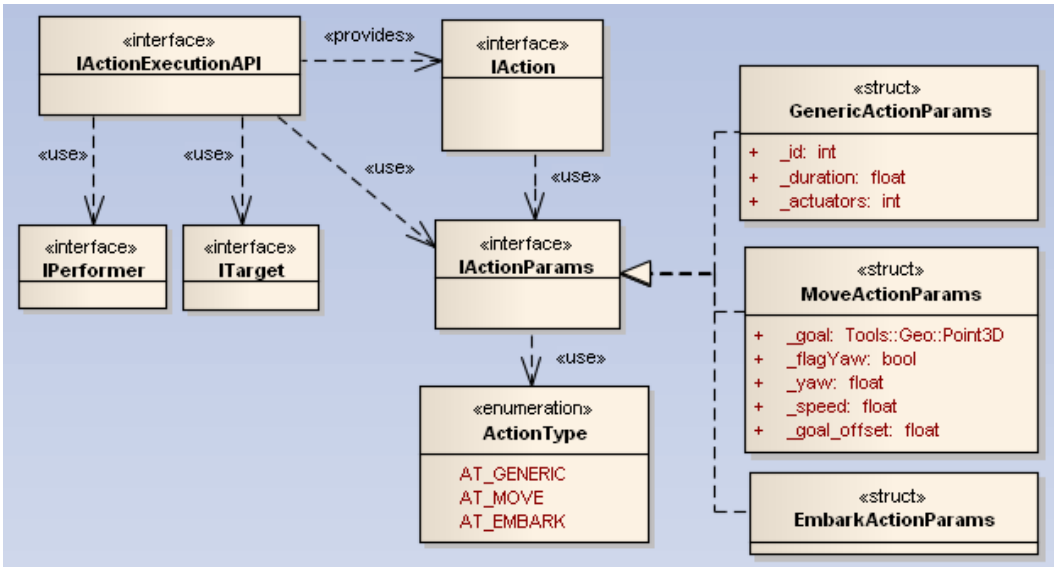

FIGURE 7.5 – Interface d'exécution des actions

L'agent réalisant l'action est le "IPerformer", la cible de l'action (si elle existe) est le "ITarget". L'action peut-être de 3 types différents :

- `AT_MOVE`

indique une action de déplacement.

- `AT_EMBARK`
  indique une action de montée ou de descente d'un véhicule.
- `AT_GENERIC`
  indique une action "classique", c'est-à-dire une action ne rentrant dans aucune des deux catégories précédentes (exemple : "boire un café").

En ce qui concerne les actions classiques, le module de décision doit indiquer au module d'exécution, l'identifiant de l'action en question, sa durée, et les actuateurs utilisés.

Pour les actions de déplacement, le module de décision doit communiquer la destination, l'orientation désirée lorsque la destination est atteinte (typiquement dans le cas d'une interaction avec un objet, par exemple un distributeur de billets, une destination n'est pas suffisante pour que l'action s'effectue de manière cohérente), une vitesse désirée, et une distance à partir de laquelle il est possible de considérer que l'agent est arrivé.

Lorsqu'une action se termine, c'est au module d'exécution d'en informer la décision. Une terminaison peut-être due à l'échec de l'action, à sa réussite, ou à une interruption (par exemple si l'agent change d'avis et décide de faire autre chose).

7.3.4 Sémantique de l'environnement

En plus des perceptions gérées par l'environnement, la sémantique de l'environnement [Harkouken 2013] possède 2 interfaces spécifiques pouvant être utilisées par le module de décision lors du processus décisionnel : le mode interrogatif et le mode proactif.

7.3.4.1 Mode interrogatif

Ce mode d'interaction est utilisé lors de la phase de génération des instanciations de plan, plus précisément dans la méthode "interrogerEnvironnement" de l'algorithme décisionnel (voir 4.5).

L'appel à cette méthode est de la forme :

```
iModeInterrogatif(
  Actor *actor,       // Agent cherchant un(des) service(s)
  const listOfService,// Liste des services recherchés
  const coefQoS,      // Coefficients de qualité de service dans [0;1]
  size_t X,           // Nombre max indicatif de résultats souhaités
  RES &res            // Tableau de résultats
);
```

avec "coefQoS" les coefficients indiquant l'importance des différents critères de sélection de service. Les critères de sélection étant :

- Temps. L'agent cherche un service plus ou moins rapide.
- Coût. L'agent cherche un service dont le prix est plus ou moins faible.
- Efficacité. L'agent cherche un service plus ou moins efficace (correspondant exactement à ce qu'il cherche).
- Difficulté. L'agent cherche un service plus ou moins simple (pouvant être utilisé sans conditions).

Les services correspondant à la recherche sont classés selon leur qualité de service, c'est-à-dire leur adéquation avec ce que l'agent cherche précisément. Les X services les mieux classés sont envoyés. La décision n'étudie donc que les services les plus adaptés aux besoins de l'agent.

7.3.4.2 Mode proactif

Le mode proactif permet d'abonner un agent à la recherche active d'un service. Cette inscription peut être faite pour deux raisons principales. Premièrement il ne connaît aucun lieu lui permettant de réaliser une action qu'il désire réaliser. Deuxièmement, cette action est très peu prioritaire, et les lieux qui lui permettent de la réaliser sont mal placés (loin des autres lieux dans lesquels il souhaite se rendre). Dans ces cas, le module décisionnel peut décider d'inscrire l'agent au mode proactif pour le service recherché. Dès qu'il passera à proximité d'un lieu délivrant le service recherché, ou un service s'en approchant, il recevra une notification l'informant de l'emplacement de ce service découvert. Il pourra ainsi replanifier son comportement en prenant en compte l'existence de ce nouveau service.

7.4 Fonctionnement interne de l'architecture décisionnelle

7.4.1 Propositions de comportements

Les propositions de comportement sont un élément majeur de l'architecture, puisque c'est à travers elles que les modules de haut-niveau communiquent avec le reste de l'architecture. Ces propositions de comportement respectent un formalisme simple. Toute proposition de comportement est composée de 3 parties : l'indice du comportement proposé, la priorité que ce comportement a pour le module le proposant, et éventuellement un paramètre.

7.4.1.1 Indice du comportement proposé

Le graphe de l'ensemble des comportements adoptables par les agents de la simulation est généré automatiquement lors du lancement de chaque simulation (voir 3.3.1). A ce moment, un indice unique est attribué à chaque comportement du graphe. Chacun des modules de l'architecture est capable, à partir de cet indice, de savoir de quel comportement il s'agit. Lorsqu'un module de haut-niveau propose

un comportement, afin d'indiquer au module décisionnel de quel comportement il s'agit, il lui suffit donc d'indiquer cet indice.

7.4.1.2 Priorité des propositions de comportement

Indiquer quel comportement le module de haut-niveau souhaite que l'agent effectue n'est pas suffisant. En effet, il est également impératif d'indiquer l'importance que ce comportement a pour le module de haut-niveau qui le propose. Ce comportement a-t-il une importance vitale ? Ou est-ce que son importance est-elle insignifiante ? De plus il faut que le module décisionnel soit capable de comparer les différentes priorités qui lui parviennent. Il faut donc choisir une échelle commune.

Nous avons choisi une solution simple, quitte à la modifier par la suite s'il s'avère que cette simplicité est contraignante. Les priorités sont des valeurs réelles, comprises entre -1 et 1, 0 indiquant l'absence de priorité (comportement inutile), 1 indiquant une priorité absolue et -1 une répulsion totale. Le choix de valeurs numériques pour caractériser les priorités vient du fonctionnement du module décisionnel sous-jacent, qui lui est basé sur de la manipulation de variables. Le choix des bornes est purement arbitraire.

En pratique les valeurs -1 et 1 sont très particulières. En effet, si les priorités étaient choisies dans l'intervalle $[1; 1]$, cela signifierait qu'un module proposant un comportement avec une priorité de 1 souhaiterait que ce comportement soit sélectionné par le module décisionnel, quel que soit l'ensemble des priorités des autres comportements proposés par les autres modules. Or les modules de haut niveau n'ont pas accès aux connaissances des autres modules, ils ne sont donc jamais capables de savoir si leurs propositions sont plus importantes que celles des autres. Ils peuvent donc proposer des comportements dont la priorité tend vers 1, sans jamais l'atteindre.

Un module peut également désirer qu'un comportement ne soit pas sélectionné. Dans ce cas le module envoie le comportement avec une priorité négative. De la même manière que pour les priorités positives, les modules peuvent envoyer des comportements dont la priorité tend vers -1, mais sans jamais l'atteindre.

Exemple : Soit un module M gérant les relations sociales et les liens d'amitié entre agents. Si Alice est en colère contre Bob, le module M d'Alice peut désirer que les comportements "téléphoner à Bob" ou "discuter avec Bob" ne soient pas activés, et donc les proposer avec des priorités négatives.

Notation : A un instant t, l'ensemble des propositions de comportement envoyées par un module M est un vecteur $\overrightarrow{V}$ de taille n, n étant le nombre des comportements du graphe comportemental (c'est-à-dire le nombre de nœuds, feuilles comprises). Ce vecteur est noté :

$$\overrightarrow{V} = \begin{pmatrix} v_1 \\ v_2 \\ ... \\ v_n \end{pmatrix}$$

Chaque élément v_i de ce vecteur correspond à un comportement. Ces éléments appartiennent à l'intervalle $]-1;1[$. Si le module M ne soumet aucune proposition liée à l'élément i, alors v_i est nul. Les modules n'étant a priori pas intéressés par tous les comportements possibles de l'agent, la majorité des éléments du vecteur sont nuls.

7.4.1.3 Paramètre

Pour certaines actions, l'indice du comportement ne suffit pas à définir entièrement le comportement souhaité. Dans notre modèle, les paramètres ne peuvent être que de deux types :

- Soit des **lieux**, représentés par un triplet de coordonnées spatiales ;
- Soit des agents, caractérisés par leur identifiant (fourni lors de leur création).

Ces paramètres permettent de gérer les actions dotées d'une cible. Par exemple une action de déplacement doit être accompagnée d'un lieu vers lequel se rendre. D'un autre côté, une action de communication doit être pourvue d'une référence vers un autre agent. Les propositions de comportements peuvent donc être dotées d'un paramètre permettant de compléter les informations.

D'un point de vue technique, ces paramètres supplémentaires facultatifs ne modifient pas la généricité du modèle. En effet, à chaque action est automatiquement associée des coordonnées. Si l'action possède un paramètre de type "lieu", alors ces coordonnées sont utilisées. Si l'action possède un paramètre de type "agent", alors les coordonnées utilisées sont celles de la position actuelle de l'agent cible. Si l'action ne possède pas de paramètre, alors les coordonnées utilisées sont celles correspondant à la position actuelle de l'agent effectuant l'action.

7.4.2 Le fonctionnement des fichiers de paramètres du module décisionnel

Lorsqu'on a pour objectif de développer une architecture générique, cela provoque souvent une multiplication des fichiers de paramètres. En effet, plus un système gagne en généricité, plus il a de paramètres modifiables par le modélisateur. Il faut donc un moyen de rendre ces paramètres accessibles, et les fichiers de paramètres sont le moyen le plus couramment utilisé.

7.4.2.1 Fichier de description du graphe comportemental

Le premier fichier de paramètre de la simulation est celui permettant de décrire le graphe comportemental utilisé. Ce fichier est appelé "BehaviorTree.xml". C'est

le tout premier fichier lu par le module décisionnel. A partir de lui, le module décisionnel va générer le graphe comportemental qui sera utilisé dans la simulation, et ce graphe sera envoyé à tous les autres modules intervenant dans la génération des comportements des agents.

Ce graphe doit indiquer pour chaque comportement quels sont ses comportements parents, quels sont ses comportements fils, et de quel type de nœud il s'agit ("et", "ou", ou bien "action").

Exemple : Le comportement "Manger" est un fils du comportement "Se nourrir", ses fils sont "Manger au restaurant", "Manger dans un fast-food" et "Manger chez soi". C'est un nœud "ou", car pour réaliser le comportement "Manger" il suffit que l'agent fasse l'un des comportements fils (et pas les trois).

Un exemple de ce type de fichier est donné dans l'annexe A.1.1, alors qu'un exemple de graphe comportemental complet est donné dans l'annexe B.

7.4.2.2 Fichier de description des actions

Les actions ont des caractéristiques, qui sont paramétrables. En fonction de la simulation, une même action peut avoir des caractéristiques très différentes. Le fichier "ActionsDecision.xml" décrit l'ensemble de ces caractéristiques.

Les premières à devoir être spécifiées sont celles correspondant aux critères de coût ayant été définis (voir 3.3.3). Chaque action doit pouvoir être comparée aux autres en fonction des critères jugés pertinents pour cette simulation. Dans nos simulations, nous avons choisi comme critères le temps et le prix des actions. De plus nous avons choisi de donner des valeurs uniquement via des fourchettes (temporelle ou monétaire).

D'autres caractéristiques doivent être indiquées. Il s'agit typiquement des effets des actions (voir 3.3.4) qui concernent soit directement l'environnement, soit des variables internes de l'agent qui ne dépendent pas de ses modules de haut-niveau (par exemple son inventaire). Ces modifications internes ou externes peuvent être soumises à de l'aléatoire.

Si l'action a une ou plusieurs préconditions (voir 3.3.6), elles doivent également être indiquées dans ce fichier. Une précondition peut porter sur l'exécution préalable d'une autre action, sur une variable d'inventaire, une caractéristique, etc.

Par ailleurs, ce fichier doit indiquer les actuateurs (voir 3.3.5) qui sont utilisés par l'agent lorsqu'il exécute l'action (nous rappelons que le nombre et la disposition des actuateurs sont également paramétrables). Dans notre cas, nous avons choisi de ne pas augmenter la complexité du modèle, et de ne pas considérer qu'une action peut n'utiliser qu'une fraction d'un actuateur.

Dernière chose (du moins dans le cadre des simulations que nous avons réalisées), les liens entre les actions (voir 3.3.7) sont spécifiés à cet endroit. Il peut s'agir d'un comportement à réaliser immédiatement avant, ou après.

Exemple : L'action "Manger au restaurant" a une durée comprise entre 30 minutes et 90 minutes. Son prix est compris entre 15 et 40 euros. Pour pouvoir réaliser cette action, l'agent doit avoir au préalable réalisé l'action "Réserver". A l'issue de l'action, l'agent obtient dans son inventaire l'objet "ticket de caisse". Il a également 10% de chance d'obtenir l'objet "carte de fidélité". Le restaurant dans lequel l'agent a mangé incrémente de 1 le nombre des repas servis. Pendant son action, l'agent utilise l'ensemble de ses actuateurs, c'est-à-dire ses jambes, ses bras, et sa tête. Immédiatement après avoir réalisé cette action, l'agent doit réaliser l'action "Payer la note".

Un exemple de ce type de fichier est donné dans l'annexe A.1.2.

7.4.3 Le fonctionnement des fichiers de paramètres des modules de haut-niveau

La capacité de l'architecture à intégrer n'importe quel module de haut-niveau oblige chaque module de haut-niveau à être paramétré, afin de connaître les actions et comportements pertinents par rapport à son domaine de compétence, ainsi que la manière dont il doit interpréter les différents stimuli qu'il prend en entrée.

La manière la plus générique de fonctionner est de passer par des fichiers de paramètres spécifiques pour chaque module, indiquant la façon dont le module doit comprendre les informations qui lui sont transmises. En procédant de la sorte, il devient facile de paramétrer ces modules de l'extérieur (sans modification du code source), même sans avoir connaissance des mécanismes internes, et de rajouter l'interprétation de nouveaux stimuli (par exemple lorsque le scénario utilisé se complexifie, ou lorsque le module est appliqué à de nouvelles applications).

Rien n'oblige à adopter le formalisme que nous allons présenter lors de l'implémentation d'un nouveau module dans l'architecture, mais pour des raisons de clarté et d'uniformité, il est fortement recommandé de le faire.

A chaque module de haut-niveau sont associés deux fichiers de paramètres. Le premier, nommé "NomDuModule.xml" sert à décrire le fonctionnement interne générique du module, alors que le deuxième, nommé "ActionsNomDuModule.xml" décrit les effets des événements de la simulation sur le module.

7.4.3.1 Fonctionnement interne du module

Le fichier "NomDuModule.xml" a pour objectif de décrire les caractéristiques du module. Cela peut représenter différentes données en fonction du module, par exemple des variables internes, des fonctions d'évolution, des ressources, des règles comportementales, etc.

Exemple : Soit M un module de motivations. Dans des scénarios de gestion de

crises causées par des catastrophes naturelles, la motivation de "divertissement" peut ne pas être utilisée, puisque l'impact de l'envie de divertissement dans ce genre de cas peut être négligée.

Évidemment, le fonctionnement interne du module ne peut pas être entièrement paramétré, mais plus il est modulable, plus le module est réutilisable pour des applications différentes. Aucun format n'est imposé pour ce document. Étant donné que les informations du fichier ne concernent que le module en question, et qu'il ne sera lu et interprété que par ce module, la structure du fichier, les mots-clés, et l'organisation sont laissés à la discrétion du modélisateur du module. Le format même du document peut être libre, puisque le module lui-même se charge de l'ouverture et de la lecture. Nous recommandons tout de même le format xml en raison de sa facilité d'utilisation, et de son côté standard.

Un exemple de fichier de paramètres de ce type est donné dans l'annexe A.2.1.

7.4.3.2 Effets des actions sur le module

Le fichier "ActionsNomDuModule.xml", a pour but d'interpréter pour le module, chacun des événements de la simulation. Alors que le fichier précédent n'est pas lié à une simulation (dans le sens où un même fichier peut être utilisé dans deux simulations différentes), le fichier de description des événements est lui dépendant de la simulation dans laquelle le module est utilisé. Ce fichier pourrait être assimilé à l'interface entre le module et la simulation. Comme le module et l'environnement n'ont rien en commun (pas même un langage de description), il est nécessaire que quelque chose opère une traduction entre les événements se déroulant dans la simulation et le module. Si un événement n'est pas décrit dans ce fichier, alors le module n'aura aucune visibilité le concernant : il ne saura même pas qu'il s'est produit.

Les événements principaux devant être décrits sont les effets des actions et les perceptions. Chaque événement dont le module prend connaissance peut l'affecter de différentes manières. Pour ce fichier non plus, il n'est pas possible de fixer de standard de forme ou langage, puisque les modèles internes des modèles ne sont pas connus a priori. Pour un module, les événements peuvent modifier la valeur d'une variable interne, ou la fonction permettant de les calculer (c'est par exemple le cas du module motivationnel), alors que pour un autre module un événement peut faire changer une ressource de pile (c'est par exemple le cas du module de comportements affectifs).

Un exemple de ce type de fichier est donné dans l'annexe A.2.2.

7.4.4 Description des agents

Il est également nécessaire de paramétrer le agents, les acteurs de la simulation. Le fichier de paramètres au cœur de la description des agents est le fichier "ActorModels.xml", dont l'annexe A.7 donne un exemple.

Ce fichier permet d'indiquer l'ensemble des actions que les agents de chaque type

peuvent accomplir, ainsi que leurs préférences associées (une fourchette de valeurs). Il donne les modules de haut-niveau actifs et leurs poids (également une fourchette de valeurs). Enfin, le fichier explique l'état initial de l'inventaire.

7.5 Description des scénarios

En plus de l'ensemble des fichiers de paramètres abordés précédemment permettant de configurer la simulation, deux autres fichiers sont spécifiquement utilisés pour définir le scénario.

Cette description se fait principalement grâce au fichier "SourceSink.xml", qui définit l'ensemble des points d'apparition des agents, ainsi que leur fréquence de création, leur type, etc. (plus de détails sur ce fichier sont donnés dans l'annexe A.8).

Ce fichier permet de peupler l'environnement avec l'ensemble des agents qui y évoluent au moment où la simulation commence (piétons, véhicules en stationnement et en circulation, etc.). Il peut par exemple tenir compte de l'heure courante de la simulation pour indiquer la fréquence d'apparition d'agents via une source. Les indications qu'il contient doivent permettre de créer des flux (piétons et véhicules notamment) crédibles, c'est-à-dire ressemblant aux flux réels. Pour arriver à un tel degré de réalisme, des comptages des flux réels sont souvent nécessaires.

7.6 Conclusion

A l'heure actuelle, le logiciel développé n'a qu'un objectif de démonstration, mais il est prévu une intégration des technologies développées dans de réelles applications. En l'état, le logiciel développé peut déjà servir d'outil d'aide à la décision, par exemple pour des projets de modification de l'aménagement urbain. Mais appliquer l'architecture à d'autres applications que celles visées dans le projet permettrait également de tester la solidité et la généricité des modèles utilisés (actions, agents, environnement, etc.).

Chapitre 8

Conclusion et perspectives

Sommaire

8.1 Conclusion

8.1.1 Objectifs

Nous venons de présenter nos travaux, portant sur la proposition d'une architecture d'agents hybride et flexible. Cette architecture combine en son sein les trois contributions principales de la thèse, qui sont :

- L'organisation de l'architecture elle-même, son fonctionnement, sa structure, et sa modélisation.
- Le module décisionnel, réalisant une composition de comportements.
- Le module d'anticipation, permettant de doter les agents de capacités de prédiction et de prise en compte de ces prédictions.

Ces trois contributions permettent à l'architecture proposée de rassembler quatre caractéristiques n'ayant encore jamais été combinées dans une même architecture : elle est à la fois générique, flexible, elle permet de modéliser des comportements crédibles, et elle est capable de passer à l'échelle.

La modélisation d'agents est un sujet étudié depuis longtemps dans des domaines différents (intelligence artificielle, robotique, sciences cognitives, etc.), et utilisé pour des applications variées (urbanisme, sécurité, aide à la décision, etc). Il s'ensuit qu'il existe de très nombreux types d'agents (réactifs, cognitifs, rationnels, émotionnels, virtuels, etc.), fondés sur des théories hétérogènes. En allant plus loin,

au sein d'un même domaine, l'intelligence artificielle, la définition d'un agent intelligent n'est pas clairement délimitée. Il peut s'agir "d'un agent qui pense comme un humain", "d'un agent qui pense de manière rationnelle", "d'un agent qui agit comme un humain", ou encore "d'un agent qui agit rationnellement" [Russell 2010]. Ces quatre définitions montrent bien qu'en fonction des objectifs que l'on souhaite que les agents accomplissent, toute la modélisation, et toute la structure même de l'agent sont modifiées. Un agent est toujours construit en vue de la réalisation de quelque chose, et ce but justifie bien souvent la manière dont il est modélisé.

Ainsi, il existe de nombreuses théories, soutenues par des domaines de recherches distincts, qui permettent de modéliser des agents exhibant telle particularité comportementale, ou telle méthode de réflexion, autorisant les agents à remplir les objectifs leur ayant été préalablement fixés. L'idée que nous défendons est la proposition d'une méta-architecture d'agents, capable d'intégrer n'importe quels composants, systèmes, ou sous-architectures provenant de théories diverses. De cette manière il devient possible de modéliser des agents aussi différents les uns des autres que désirés (afin de répondre à des objectifs hétérogènes), tout en gardant une même structure interne. L'intérêt de cette **généricité** est énorme. D'une part, elle permet de combiner au sein d'un même agent des composants issus de théories différentes, ce qui permet d'obtenir des agents exhibant plusieurs caractéristiques intéressantes, et donc de modéliser un large spectre d'agents, de comportements, et donc de scénarios, et d'applications hétérogènes. Mais cela permet d'autre part, de dresser des ponts entre des domaines jusqu'alors isolés, mais travaillant sur les mêmes types d'applications, en leur offrant une plate-forme de collaboration. Pour finir, la comparaison et les tests des différentes théories sont simplifiés puisque l'impact de la qualité de la modélisation des agents est grandement diminué.

Cette généricité est importante, mais elle n'est pas suffisante en tant que telle. En effet, inclure dans un même agent des composants issus de théories différentes est intéressant, mais encore faut-il que le processus décisionnel de l'agent soit capable de traiter correctement ces sources comportementales hétérogènes. En effet, des théories comportementales distinctes peuvent traiter de comportements extrêmement différents et ces différences peuvent agir sur plusieurs plans. Tout d'abord sur le plan de la complexité : le processus décisionnel peut se retrouver confronté à des comportements aussi complexes que des objectifs de très haut-niveau (préparer un voyage, essayer d'obtenir une promotion, etc.) et dans le même temps à des comportements de très bas niveau (tendre la main, se baisser, etc.). Les différences peuvent également porter sur l'importance des comportements, allant de comportements vitaux (fuir un ennemi, se protéger d'une menace) à des comportements triviaux ou peu utiles (regarder les nuages, compter les dalles d'un carrelage, etc.). Mais, et c'est peut-être encore plus difficile à traiter, les comportements proposés peuvent porter sur des domaines n'ayant rien à voir les uns avec les autres (ce qui est cohérent avec l'idée que les modules de comportements sont issus de théories distinctes). Ainsi, le processus décisionnel peut avoir à traiter dans le même temps des comportements portant sur la vie sentimentale et/ou familiale de l'agent (télépho-

ner à quelqu'un, demander conseil, etc.), sur ses motivations élémentaires (faim, soif, fatigue, etc.) et sur sa vie professionnelle (faire des heures supplémentaires, chercher un travail, etc.). Plus l'agent doit traiter de sources comportementales différentes, plus le travail décisionnel sera important et complexe. Il faut donc trouver un moyen de comparer et d'intégrer ces comportements de manière cohérente.

Par ailleurs, si l'agent n'est capable que de sélectionner une source comportementale à la fois, et de suivre ses propositions de comportement (principe "*Winner-take-all*"), l'intégration des différents modules comportementaux n'est pas complète. Il est également nécessaire que l'agent puisse combiner les comportements entre eux, afin que le comportement de sortie de l'agent (le comportement que l'agent décide d'adopter, après réflexion) soit issu d'une collaboration, d'une mise en commun des différents raisonnements effectués par les modules de comportement. Cette composition de comportement permet au processus décisionnel de devenir **flexible**, et de s'adapter aux différentes théories.

Comme cette architecture a pour objectif de modéliser le plus grand nombre d'agents possible, il est intéressant d'étudier le cas des humains virtuels, qui sont l'un des types d'agents les plus complexes à simuler, en raison de leur objectif d'imitation du comportement humain. Afin de juger de la qualité de simulation d'un humain virtuel, nous avons choisi comme méthode de mesure la **crédibilité** des comportements des agents, jugée par un observateur humain. Afin de valider la capacité de notre système à proposer des comportements jugés crédibles, nous avons donc instancié la méta-architecture proposée, en la dotant d'un certain nombre de modules comportementaux simples. Ces modules ne suffisant pas à garantir une bonne crédibilité comportementale, nous avons ajouté au processus décisionnel des capacités d'anticipation, en raison de l'importance croissante leur étant attribuée [Pezzulo 2009a].

Ces capacités d'anticipation permettent aux agents de produire des prédictions, grâce à des modèles prédictifs, ce qui leur permet d'agir immédiatement en prenant en compte des informations sur un état futur de l'environnement, ce qui augmente grandement à la fois l'efficacité des comportements, mais également leur crédibilité. Les observateurs sont capables de reconnaître ces comportements spécifiques, et leur attribuent des scores de crédibilité plus importants, en raison de leur ressemblance avec des comportements humains.

Le cadre applicatif dans lequel ce travail a été réalisé est celui de la simulation urbaine. Dans ce genre de simulations, il est important de pouvoir simuler un grand nombre d'agents en parallèle (les villes regroupent de nombreux habitants). Il faut donc que l'architecture d'agents utilisée dans la modélisation des agents de la simulation soit capable de **passer à l'échelle**. Cette capacité n'est pas gérée grâce à une partie particulière de l'architecture traitant spécifiquement de cet aspect, mais c'est l'ensemble de ses composants qui ont été conçus de manière à rendre le passage à l'échelle possible, en exploitant notamment le concept du niveau de détail en Intelligence Artificielle [Wissner 2010]. Ainsi, il est possible d'adapter la com-

plexité décisionnelle des agents, d'une part aux ressources disponibles, et d'autre part à l'importance de ces agents. L'importance d'un agent pouvant dépendre de son rôle au sein de la simulation, de sa visibilité (pour un observateur), de sa localisation, etc. Ainsi il est possible de rendre certains agents importants particulièrement complexes dans leur modélisation et leur profondeur de réflexion, et de simplifier d'autres agents moins importants afin de gagner en temps de calcul.

8.1.2 Choix de modélisation

Afin de répondre aux quatre points ci-dessus, nous avons opté pour une architecture modulaire, composée d'un nombre quelconque de modules de haut-niveau (chacun pouvant être basé sur des théories différentes) envoyant des propositions de comportement potentiellement intéressant à un module décisionnel chargé de leur composition. Ce module décisionnel est inspiré des travaux de [Tyrrell 1993a] sur la hiérarchie à libre-flux. Mais ce modèle est largement modifié de manière à ce qu'il compare non pas des actions les unes par rapport aux autres, mais des instanciations de plans (à savoir des enchaînement d'actions concrètes). Cette planification est dynamique, c'est-à-dire que les plans sont construits lors du lancement de la simulation (ce qui permet d'obtenir un gain de temps de calcul important). Une fréquence de replanification élevée (de l'ordre de la seconde, donc proche d'un temps de réaction humain) permet cependant de garder l'agent réactif, et de lui permet d'adapter ses plans à l'évolution du contexte. Afin d'augmenter la crédibilité des comportements, un module de haut-niveau spécifique a été ajouté (inspiré des travaux de [Davidsson 1994]). Il prend en charge les capacités d'anticipation de l'agent. Pour cela, il se sert de modèles prédictifs, fournis par le modélisateur, mais pouvant (à terme) être appris.

8.1.3 Validation

L'ensemble des quatre caractéristiques principales de l'architecture ont été testées et validées en détail, grâce à différents types d'expérimentations. Des expérimentations objectives et quantitatives ont été menées, afin de calibrer puis valider le fonctionnement de base de l'architecture, et de ses différents composants (modules de haut-niveau réactifs, et module décisionnel). Les capacités du système par rapport à l'objectif du passage à l'échelle ont également été mesurées.

Ensuite, des expérimentations utilisateurs subjectives ont été réalisées (grâce à plus de 150 participants). Ces dernières expérimentations ont permis de mettre en évidence l'intérêt des capacités d'anticipation intégrées dans l'architecture, notamment grâce à des études comparatives d'efficacité et de crédibilité des comportements. Un lien entre l'efficacité et la crédibilité d'un comportement a ainsi pu être déterminé, et nous avons suggéré l'existence d'un *Canny Hill*, c'est-à-dire un pic de crédibilité d'un comportement, lorsque son efficacité est jugée comme bonne, sans être *trop* importante [Reynaud 2014]. Nous avons également vérifié la difficulté éprouvée par un observateur à se rendre compte de l'intelligence des agents d'une

foule (à ce niveau, la crédibilité porte plus sur le comportement de la foule en temps qu'entité, plutôt que sur les comportements des agents qui la composent).

Cette architecture a également été testée et validée par le biais du projet Terra Dynamica (dans laquelle elle a été développée), sur l'ensemble des quatre points abordés. En effet, les différents partenaires académiques et industriels du projet ont développé des applications spécifiques à leurs domaines de compétence, et ont démontré leur intérêt lors d'un festival présentant les nouvelles technologies numériques en Juin 2013 à Paris.

8.1.4 Difficultés rencontrées

Au cours de ce travail, le premier obstacle a été de se placer par rapport à l'état de l'art, très richement fourni en architectures d'agents. Le caractère hybride de l'architecture est rapidement apparu comme indispensable, mais aucune des hybridations réalisées dans les travaux précédents ne permettaient de répondre à l'ensemble de nos objectifs. Grâce à un travail collaboratif, notamment avec Etienne de Sevin, Jean-Yves Donnart, et Vincent Corruble, une nouvelle hybridation entre comportements réactifs et cognitifs a été proposée [de Sevin 2012b].

Une des forces principales du travail est la dualité ingénierie/recherche de sa conception, issue du partenariat industrie/recherche, à la fois dans le projet Terra Dynamica, et dans l'encadrement de la thèse (thèse CIFRE entre Thales Training and Simulation et le Laboratoire d'Informatique de Paris 6). Cette spécificité est à la fois une force et une contrainte. Une force puisque ce travail a débouché sur le développement d'un logiciel ayant fait l'objet de nombreuses démonstrations académiques (lors de conférences), et industrielles (lors de festivals, ou de présentations officielles). Une contrainte, puisque le développement de l'architecture a été réalisé avec les contraintes habituellement associées au milieu industriel : cycles de développement lourds, normes à respecter, langage et environnement de développement imposés, démonstrations intermédiaires régulières, etc. ce qui a provoqué un allongement du temps initialement prévu pour le développement.

Une autre difficulté rencontrée est liée à la multiplicité des pistes de recherches abordées (généricité de l'architecture, composition de comportement, anticipation, passage à l'échelle). Il a fallu fixer un calendrier assez strict (et le suivre) afin de pouvoir passer suffisamment de temps sur chacune. Au final, l'ensemble est cohérent, et le travail réalisé gagne en profondeur grâce aux interactions entre les différentes pistes.

8.2 Perspectives

L'objectif premier de cette thèse était très ambitieux, puisqu'il s'agissait de proposer une architecture d'agents pouvant être utilisée dans des applications aussi différentes que la sécurité, les transports, le jeu vidéo, et l'urbanisme (c'est-à-dire les quatre applications concernées par le projet Terra Dynamica). En réalité, ce

sont même une adaptation à l'ensemble des domaines d'application de la simulation agent, et une incorporation au sein de la même architecture des différents domaines travaillant sur la simulation agent, qui sont visés. Il serait présomptueux de considérer ce double objectif comme entièrement atteint, mais nous sommes convaincus que l'architecture proposée permet de faire un premier pas vers un rapprochement des différents domaines de recherches travaillant sur la simulation agent. Nous pensons que la constitution d'une méta-architecture, largement reconnue dans des domaines différents, dans laquelle des composants (issus de différentes théories), seraient incorporés sous forme d'une bibliothèque de modules, est une des meilleures manières d'unifier l'ensemble des théories de description (ou de production) des comportements. L'architecture présentée dans cette thèse, propose d'ouvrir ce nouvel axe de recherche. De nombreuses perspectives restent à être traitées, d'une part à court terme pour que cette architecture gagne en intérêt et en attractivité pour des spécialistes d'autres domaines de recherche ; et d'autre part à long terme, car ce nouvel axe de recherche mène à la rencontre de nouveaux problèmes.

8.2.1 Perspectives appliquées

8.2.1.1 Perspectives à court terme

A court terme, le travail réalisé dans le cadre du projet Terra Dynamica pourrait faire l'objet de nombreux développements et améliorations.

Il existe tout d'abord de nombreuses possibilités d'enrichissement du modèle. Le modèle d'actions pourrait par exemple être plus complet. Dans la version actuelle du modèle, toutes les actions ont une durée. Or il serait judicieux de considérer également des actions sans durée définie à l'avance. Il serait également intéressant d'envisager que les individualités des agents aient un impact sur les durées des actions (par exemple, un agent aimant énormément réaliser une action pourrait y passer plus de temps), voire rajouter une variable d'individualisation (la rapidité ?) indiquant la tendance de l'agent à prendre son temps, ou au contraire à se dépêcher.

Les conditions d'échec des actions sont encore au stade embryonnaire. Le modèle gagnerait énormément en utilisabilité si ces conditions étaient approfondies. De nombreuses pistes sont envisageables : pourcentages prédéfinis d'échec pouvant être en lien avec l'état de l'environnement ou de l'agent, totale maîtrise de la réussite ou de l'échec des actions donnée au modélisateur de la simulation, etc.

La manière dont les effets des actions se produisent pourrait également être améliorée. A l'heure actuelle, les effets se produisent uniquement à la fin de l'action, lorsqu'elle est réussie. Il est possible de créer différents types d'actions, dont certaines appliquent leurs effets de manière continue tout au long de leur accomplissement.

Sur les actions toujours, la notion de collaboration en vue de la réalisation d'une action est une piste à étudier. Il s'agir d'actions pouvant (ou devant) être réalisées en commun (faire la vaisselle, jouer au rugby, etc.) et dont les bénéfices

(effets) peuvent être partagés par l'ensemble des participants.

Les agents pourraient eux-aussi être plus complexes. Un modèle peut toujours être enrichi bien sûr, et à partir d'un certain point cela apporte plus de difficultés que d'intérêt, mais nos agents restent pour l'instant assez synthétiques. L'aspect relationnel notamment est peu approfondi (sauf dans le cadre de la patrouille [Poulet 2013]. Les agents n'ont ni amis, ni connaissances, ni famille. Par rapport aux simulations que nous avons réalisées cela ne posait pas de problème, mais c'est une limitation importante du modèle.

Pour finir, afin d'enrichir les évaluations et de compléter les tests réalisés, il serait également intéressant de rendre l'environnement utilisé dans le projet Terra Dynamica plus complet. L'intérieur des bâtiments est peu souvent accessible (et jamais visible par un observateur), les conditions climatiques n'ont aucun réel effet sur les agents, il existe trop peu de types différents de points d'intérêt, et le nombre des animations disponibles est beaucoup trop faible : autant d'améliorations pertinentes afin d'augmenter le réalisme des simulations. Par ailleurs, le graphe comportemental utilisé pourrait être à la fois plus large et plus haut, c'est-à-dire doté de plus de comportements réalisables, et de plus grande complexité.

8.2.1.2 Perspectives à long terme

En plus de ces perspectives à "court terme", il est également possible de citer quelques perspectives à plus long terme sur lesquelles nos travaux peuvent déboucher.

Le projet Terra Dynamica était situé dans le prolongement de deux autres projets : Terra Numerica, et Terra Magna. Un nouveau projet a déjà pris la suite, Terra Mobilita [1]. Ce projet vise à améliorer la modélisation 3D de la voirie et de l'espace public. S'il ne se sert pas du travail accompli dans Terra Dynamica, l'inverse pourra être le cas. C'est-à-dire qu'il sera possible d'utiliser dans des simulations des environnements beaucoup plus complexes, prenant en compte de nombreux détails de la voirie et de l'aménagement urbain pouvant avoir un impact sur les comportements des agents. Et il n'est pas exagéré de penser que cette suite de projet ne s'arrêtera pas là, et que petit à petit, ils déboucheront sur des outils permettant de simuler une ville entière de manière fortement réaliste et crédible. L'architecture d'agents proposée dans cette thèse continuera donc à être exploitée sur ce plan. D'un autre côté, les deux jeux vidéos de démonstration développés dans le cadre du projet serviront peut-être de base à de réelles applications commerciales.

1. www.terradynamica.com/les-projets-terrax

8.2.2 Perspectives théoriques

8.2.2.1 Perspectives à court terme

D'un point de vue plus théorique, il existe des perspectives de travail plus importantes encore, notamment une réelle prise en compte de la dimension multi-agents des simulations. En effet, bien que le modèle rende possible l'interaction d'agents entre eux (les agents peuvent se voir, se parler, échanger des informations, etc.), il a tout de même été conçu dans un cadre mono-agent. C'est-à-dire que seul l'agent qui réfléchit est pris en compte, les autres n'étant soit pas considérés, soit considérés comme des concurrents potentiels (dans le cas des ressources à accès limité). Il serait donc vraiment pertinent de se poser la question de la coordination multi-agents, dans un cadre plus général que celui de la patrouille.

Une manière potentiellement efficace de traiter ce sujet, serait l'intégration, comme module de haut-niveau, d'un module de coordination. Ce module aurait la charge des comportements nécessitant une coordination entre agents, et pourrait éventuellement intervenir pour proposer des solutions pour des comportements traités par d'autres modules (à l'image de la couche de coordination de l'architecture InteRRaP [Müller 1993]). Cependant cette proposition nécessiterait une restructuration importante de l'architecture. En effet, pour que le système fonctionne avec un module de coordination de haut-niveau intervenant dans les comportements proposés par les autres modules, il faut que ce module soit tenu informé des buts de l'agent (c'est-à-dire des propositions de comportement). Il faudrait donc mettre une place une interaction entre les modules de haut-niveau. Cette interaction avait été écartée en première approche à cause du gain de complexité qu'elle engendrait (et donc de l'augmentation du temps de calcul nécessaire à l'ensemble du processus de génération du comportement). Mais elle semble une perspective de premier plan de l'architecture, maintenant que la capacité du modèle à simplifier les agents les moins importants afin de passer à l'échelle, a été validée.

Cette perspective est d'autant plus intéressante qu'une interaction entre modules de haut-niveau pourrait être utile pour bien d'autres modules qu'un module de coordination. On peut d'ores et déjà penser au module d'anticipation, qui pourrait inclure les dernières propositions de comportements (notamment celles non sélectionnées par le module décisionnel), dans ses calculs. On peut également réfléchir à un module d'émotions, qui pourrait appliquer un filtre émotionnel aux propositions de comportements (avant leur intégration au processus décisionnel).

Afin de réaliser cette interaction entre modules comportementaux plusieurs solutions sont envisageables. La plus prometteuse serait peut-être d'adopter un fonctionnement proche d'un tableau noir ([Erman 1980]), dans lequel une mémoire de travail centraliserait l'ensemble des informations publiées par les modules. Les modules pourraient ainsi mettre à jour leurs modèles internes après prise en compte de l'ensemble de ces données.

D'un point de vue purement pratique, cette idée d'interactions entre modules de haut-niveau est intéressante, car elle augmenterait uniquement le temps nécessaire

aux modules de haut-niveau pour envoyer leurs propositions de comportements. Il serait donc possible d'éviter un allongement de la durée totale du processus de génération comportementale, en rendant les processus de propositions des comportements et de décision asynchrones. Ainsi le processus de proposition des comportements pourrait être déclenché plus tôt, afin que les propositions soient immédiatement prêtes lorsque le module décisionnel entame son processus.

Par ailleurs, une meilleure gestion de l'aspect multi-agents des simulations permettrait de s'intéresser à une autre piste de recherche : la **narration interactive** (*Interactive Storytelling*) [Cavazza 2002, Porteous 2013] et ses relations avec les relations sociales entre agents. La narration interactive permet de donner aux modélisateurs des simulations, des outils visant à garder un certain contrôle sur le déroulement du scénario, sans pour autant le rendre trop rigide. C'est donc une capacité de choix à ajouter à noter système. Un module de narration interactive est justement en cours d'incorporation à l'architecture [Chauvin 2014]. En plus d'être une aide pour les modélisateurs, ces travaux peuvent également augmenter l'immersion des observateurs dans l'environnement virtuel et donc améliorer la crédibilité des simulations [Lugrin 2010].

Une autre perspective de première importance est l'apprentissage. Plusieurs apprentissages sont possibles, mais le plus important est sans doute celui qui pourrait être réalisé par le module d'anticipation, afin d'apprendre lui-même les modèles prédictifs qu'il utilise. Cet apprentissage permettrait de simplifier le travail du modélisateur, qui n'aurait plus besoin de fournir ces modèles en entrée du module d'anticipation. Mais cela augmenterait également fortement la généricité de ce module, puisqu'il pourrait fonctionner à partir de n'importe quels modules de haut-niveau. Cet apprentissage serait réalisé en comparant les prédictions ayant été réalisées avec le déroulement effectif de la simulation. Il serait ainsi possible de corriger les prédictions, et d'affiner les degrés de confiance des différents modèles en multipliant les observations croisées. Plusieurs pistes de travail ont été abordées à ce sujet dans la section 5.2.4. Il nous semble qu'il s'agit d'une des pistes d'amélioration du modèle les plus prometteuses, même si les résultats d'un tel processus appliqué à des comportements non prédictibles ne sont pas évidents à prévoir (est-ce que les degrés de confiance des prédictions concernant ces comportements tendraient effectivement vers 0 ? Est-il possible que le module d'anticipation tombe dans des pièges de type "surapprentissage" ?).

8.2.2.2 Perspectives à long terme

Comme nous l'avons souligné à maintes reprises, l'architecture proposée dans le cadre de cette thèse prend toute son ampleur en tant que méta-architecture dotée d'une bibliothèque de composants issus de différentes théories, pouvant être intégrés comme modules de haut-niveau. La perspective la plus intéressante est donc sûrement la constitution de cette bibliothèque. Au stade actuel, 6 modules de haut-

niveaux sont présents : 3 modules réactifs ("motivations", "instinct", "emploi du temps") et 1 module cognitif ("anticipation") ont été proposés par un des concepteurs de l'architecture, et 2 modules "externes" ont été ajoutés ("coordination" et "affects") par des chercheurs du Laboratoire d'Informatique de Paris 6, n'ayant pas travaillé sur l'architecture. Jusqu'à présent, tous ces modules ont pu être couplés dans le processus décisionnel d'un même agent. Mais il est nécessaire de vérifier qu'avec d'autres composants, issus de théories venant de domaines plus éloignés, cette composition se réaliserait aussi facilement.

Il semble un peu irréaliste d'espérer que n'importe quel point fort de n'importe quelle théorie, puisse être ainsi incorporé au modèle sans qu'aucune modification du modèle ne soit nécessaire. Cependant, cela n'est pas une limite de notre système, au contraire. Chaque ajout d'un nouveau composant issu d'une nouvelle théorie (même si cet ajout nécessite une modification du modèle), est un pas de plus vers la généralisation du modèle. A l'heure actuelle, rien ne laisse présager une éventuelle incompatibilité dans la prise en compte d'une théorie dans notre système, mais cette hypothèse ne peut pas définitivement être écartée. Cela remettrait en cause les fondations de notre modèle, mais permettrait peut-être dans le même temps de poser les bases d'une nouvelle méta-architecture toujours plus universelle...

Annexe A

Fichiers de paramètres

Dans cette annexe nous présenterons un exemple de chacun des fichiers de paramètres utilisés par les différents modules d'IA. Nous avons pris comme scénario de référence un scénario standard utilisé en particulier pour les expérimentations réalisées dans le chapitre 6.

A.1 Fichiers de paramètres généraux

A.1.1 BehaviorTree

Le fichier de paramètres "BehavirTree.xml" contient l'ensemble des informations nécessaires à la construction du graphe décisionnel. Le graphe décisionnel sera construit lors du chargement du scénario, puis partagé avec tous les modules d'IA.

Exemple :

```
<Behaviors>
  <Behavior name="seNourrir"          service="SeNourrir"
            type="or"                 childOf=""/>

  <Behavior name="seNourrirEnVille"   service="SeNourrir"
            type="or"                 childOf="seNourrir"/>

  <Behavior name="mangerFastFood"     service="MangerVite"
            type=""                   childOf="seNourrir"/>

  <Behavior name="mangerRestaurant"   service="MangerBien"
            type=""                   childOf="seNourrir"/>

  <Behavior name="mangerUnSandwich"   service="MangerSandwich"
            type=""                   childOf="seNourrir"/>

  <Behavior name="seNourrirChezsoi"   service=""
            type="and"                childOf="seNourrir"/>

  <Behavior name="mangerChezSoi"      service=""
            type=""                   childOf="seNourrirChezsoi"/>

  <Behavior name="sHydrater"          service="SHydrater"
            type="or"                 childOf=""/>

  <Behavior name="boireCafe"          service="BoireCafe"
            type=""                   childOf="sHydrater"/>

  <Behavior name="boireSoda"          service="Boire"
            type=""                   childOf="sHydrater"/>

  <Behavior name="jeterObjet"         service="JeterObjet"
            type=""                   childOf=""/>

  <Behavior name="retirerArgent"      service="RetirerArgent"
            type=""                   childOf=""/>

   etc.
</Behaviors>
```

Remarques : "service" indique le nom du service, offert par la sémantique de l'environnement, qui est associé au comportement.

"type", indique le type du nœud associé à ce comportement : soit un nœud "et", soit un nœud "ou", soit une feuille.

"childOf" indique le nom du nœud parent.

A.1.2 ActionsDecision

Le fichier de paramètres "ActionsDecision.xml" décrit l'ensemble des actions réalisables dans la simulation, avec leurs caractéristiques décisionnelles. Il est lu par le module décisionnel.

Exemple :

```
<Actions>
  <Action name="mangerFastFood">
    <Duration min="10" max="30"/>
    <Cost min="5" max="10"/>
    <Poubelle val="0.1" />
    <Actuateur val="3"/>
  </Action>

  <Action name="mangerRestaurant">
    <Duration min="30" max="90"/>
    <Cost min="10" max="50"/>
    <Poubelle val="0.1" />
    <Actuateur val="3"/>
  </Action>

  <Action name="sasseoir">
    <Duration min="10" max="30" />
    <Actuateur val="2"/>
  </Action>

  <Action name="seDetendre">
    <Duration min="10" max="30" />
    <Actuateur val="2"/>
    <Previous name="boireSoda" V "boireCafe" />
  </Action>

  <Action name="acheterRevue">
    <Duration min="1" max="5" />
    <Actuateur val="3"/>
    <Poubelle val="0.1"/>
    <Inventory plus="journal"/>
  </Action>

  <Action name="jeterObjet">
    <Duration min="0" max="1" />
    <Poubelle val="-1" />
    <Actuateur val="3"/>
  </Action>

  <Action name="retirerArgent">
    <Duration min="1" max="2" />
    <Cost min="-100" max="-50"/>
    <Poubelle val="0.1" />
    <Actuateur val="3"/>
  </Action>

  etc.
</Actions>
```

Remarque : La durée est indiquée en minutes.

Le coût est indiqué en euros.

Le champs "Poubelle" indique la probabilité qu'à la fin de l'action l'acteur se retrouve avec un objet à devoir jeter (reçu, ticket, etc.). Une valeur de -1 indique que l'agent se débarrasse de ses objets.

Le champs actuateur indique quels actuateurs sont utilisés par l'action. Un champs de bits est utilisé : la valeur 1 indique que l'agent utilise seulement ses bras, la valeur 2 indique que l'acteur utilise seulement ses jambes, et la valeur 3 indique que l'agent utilise ses bras et ses jambes.

A.2 Fichiers de paramètres du module de motivations

A.2.1 Motivations

Le fichier de paramètres "Motivations.xml" décrit les motivations et leur évolution standard.

Exemple :

```
<Motivations>
  <Motivation name="soif"     rate="0.05"
              T1="0.2"        T2="0.8"        behavior="sHydrater"/>
  <Motivation name="faim"     rate="0.03"
              T1="0.3"        T2="0.9"        behavior="seNourrir"/>
  <Motivation name="hygiene"  rate="0.01"
              T1="0.05"       T2="0.8"        behavior="wc"/>
  <Motivation name="fatigue"  rate="0.02"
              T1="0.1"        T2="0.9"        behavior="seReposer"/>
  <Motivation name="loisir"   rate="0.01"
              T1="0.05"       T2="0.7"        behavior="seDivertir"/>
</Motivations>
```

Remarque : Le "rate" indique l'évolution par heure de la variable interne associée.

Les valeurs "T1" et "T2" indiquent les seuils de confort et de danger de la motivation.

A.2.2 ActionsMotivations

Le fichier de paramètres "ActionsMotivations.xml" permet d'indiquer les effets des actions sur le module de motivations.

Exemple :

```
<Actions>
  <ActionDesc name="mangerRestaurant">
    <Effect type="add" var="faim" val="-1" />
    <Effect type="add" var="soif" val="-1" />
  </ActionDesc>

  <ActionDesc name="mangerFastFood">
    <Effect type="add" var="faim" val="-0.5" />
    <Effect type="add" var="soif" val="-0.5" />
  </ActionDesc>

  <ActionDesc name="mangerUnSandwich">
    <Effect type="add" var="faim" val="-0.5" />
  </ActionDesc>

  <ActionDesc name="mangerChezSoi">
    <Effect type="add" var="faim" val="-1" />
    <Effect type="add" var="soif" val="-1" />
  </ActionDesc>

  <ActionDesc name="boireCafe">
    <Effect type="add" var="soif" val="-0.2" />
    <Effect type="add" var="hygiene" val="0.2" />
  </ActionDesc>

  <ActionDesc name="boireSoda">
    <Effect type="add" var="soif" val="-1" />
    <Effect type="add" var="hygiene" val="0.1" />
  </ActionDesc>

  <ActionDesc name="seDetendre">
    <Effect type="add" var="fatigue" val="-0.2" />
  </ActionDesc>

  <ActionDesc name="sasseoir">
    <Effect type="add" var="fatigue" val="-0.2" />
  </ActionDesc>

  <ActionDesc name="lire">
    <Effect type="add" var="loisir" val="-0.1" />
  </ActionDesc>

  <ActionDesc name="regarderTV">
    <Effect type="add" var="loisir" val="-0.5" />
  </ActionDesc>

  <ActionDesc name="wc">
    <Effect type="add" var="hygiene" val="-1" />
  </ActionDesc>

  <ActionDesc name="travailler">
    <Effect type="add" var="loisir" val="-0.1" />
  </ActionDesc>

  etc.
</Actions>
```

A.3 Fichier de paramètres du module d'instinct

Dans le cadre de nos simulations, un seul fichier de paramètres a été utilisé pour décrire le fonctionnement du module d'instinct : le fichier de paramètres "Reaction.xml". Il indique des comportements de type "perception-action".

Exemple :

```
<Actions>
  <Action name="fuir" perception="explosion" strength="0.8" />
  <Action name="fuir" perception="arme" strength="0.7" />
  etc.
</Actions>
```

Nous remarquons qu'aucune prise en compte de l'exécution des actions n'est effectuée. En effet, nous avons jugé qu'un module d'instinct aussi simple n'avait pas besoin de gérer de variables internes.

A.4 Fichier de paramètres du module d'emploi du temps

Pour ce module également, un seul fichier de paramètres est utilisé : le fichier "EmploiDuTemps.xml". Il indique les comportements liés à l'emploi du temps d'un agent.

Exemple :

```
<Actions>
  <Action name="travailler" timeMin="8h" timeMax="12h" strength="0.6" />
  <Action name="travailler" timeMin="14h" timeMax="18h" strength="0.6" />
  etc.
</Actions>
```

Nous considérons que ce module d'emploi du temps n'a pas besoin d'avoir de variables internes. Bien sûr, on pourrait employer un module plus complexe qui en utiliserait.

A.5 Fichier de paramètre du module d'exécution des actions

Le fichier de paramètre "ActionsExecution.xml" décrit les informations nécessaires au module d'exécution afin qu'il puisse mettre en œuvre correctement les actions.

Exemple :

```
<Actions>
  <Action    name="boireCafe">
    <Begin> <Anim posture="STAND"     activity="MANIPULATE"/> </Begin>
    <End>   <Anim posture="STAND"     activity="IDLE"/>       </End>
  </Action>
  <Action    name="sasseoir">
    <Begin> <Anim posture="SIT_CHAIR" activity="IDLE"/>       </Begin>
    <End>   <Anim posture="STAND"     activity="IDLE"/>       </End>
  </Action>
  <Action    name="jeterObjet">
    <Begin> <Anim posture="STAND"     activity="DROP_TRASH"/> </Begin>
    <End>   <Anim posture="STAND"     activity="IDLE"/>       </End>
  </Action>
  <Action    name="fuir">
    <Begin> <Anim posture="RUN"       activity="IDLE"/>       </Begin>
    <End>   <Anim posture="STAND"     activity="IDLE"/>       </End>
  </Action>
  etc.
</Actions>
```

Les lignes "Begin" et "End" indiquent respectivement les postures ("posture") et activité ("activity") de l'agent pendant l'action, et lorsqu'elle se termine.

A.6 Fichier de paramètres du module de perception

Le fichier "Perceptions.xml" décrit l'ensemble des perceptions que les agents peuvent recevoir.

```
<Effects>

  <SpecialEffect type="TRASHCAN_FIRE" >
    <Perception    mode="Sound"     radius="10" name="trashcanFire" />
    <Perception    mode="View"      radius="30" name="trashcanFire" />
  </SpecialEffect>

  <ActorEffect type="RUNNING">
    <Perception    mode="Sound"     radius="15" name="running" />
    <Perception    mode="View"      radius="30" name="running" />
  </ActorEffect>

  <ActorEffect type="ACTION"                   name="manipuler" >
    <Perception    mode="ViewFront" radius="10" name="manipulation" />
  </ActorEffect>

</Effects>
```

Le mot-clef "SpecialEffect" indique un perception associée à un événement, par exemple un feu de poubelle, une explosion, une alarme, etc.

Le mot-clef "ActorEffect" indique une perception associée à un autre agent de la simulation. Elle peut être liée à une action qu'un agent effectue, ou à un mode de déplacement, à un objet qu'il porte, etc.

Le "mode" correspond au mode de perception (visuel, auditif, etc.), et le "radius" correspond à la distance en mètres depuis la source, jusqu'à laquelle la perception est détectable.

A.7 Fichier de paramètres des agents

Le fichier "ActorModels.xml" donne toutes les informations nécessaires à la création des différents types d'agents (voir 3.5.1). Voici un exemple d'un tel fichier :

```
<ActorModels>
  <ActorModel  name="Passant">
    <TypeRef   ref="casual1_m" p="1"/>
    ...
    <TypeRef   ref="child1_f" p="1"/>
    <ActionRef ref="mangerFastFood" min="0.3" max="0.9"/>
    ...
    <ActionRef ref="retirerArgent"  min="0.4" max="0.5"/>
    <VarInit   ref="cash" min="0.0" max="50.0"/>
    ...
    <VarInit   ref="ticketMetro"    min="50"  max="100"/>
    <ModuleRef ref="reaction"       min="0.7  max="0.9"/>
    <ModuleRef ref="motivation"     min="0.2  max="0.6"/>
    <ModuleRef ref="emploiDuTemps"  min="0.7  max="0.9"/>
    <ModuleRef ref="anticipation"   min="0.4  max="0.6"/>
  </ActorModel>

  <ActorModel  name="Touriste">
    <TypeRef   ref="casual1_m" p="1"/>
    ...
    <TypeRef   ref="child1_f" p="1"/>
    <ActionRef ref="mangerFastFood" min="0.3" max="0.9"/>
    ...
    <ActionRef ref="retirerArgent"  min="0.4" max="0.5"/>
    <VarInit   ref="cash"           min="0.0" max="50.0"/>
    <ModuleRef ref="reaction"       min="0.7  max="0.9"/>
    <ModuleRef ref="motivation"     min="0.7  max="0.9"/>
  </ActorModel>
</ActorModels>
```

"ActorModel" indique le nom du type d'acteur.

"TypeRef" indique des références de modèles 3 dimensions d'humains virtuels, qui serviront à donner un corps aux agents (parmi les modèles des touristes il y aura plus de gens en chemises à fleurs que parmi les passants).

"ActionRef" indique les noms des actions que les agents de ce type peuvent accomplir, ainsi qu'une fourchette dans laquelle leurs préférences seront choisies aléatoirement.

"VarInit" permet d'initialiser l'inventaire.

"ModuleRef" indique les modules de haut-niveau initialement actifs lors de la création de l'agent, ainsi qu'une fourchette indicative de leurs poids.

A.8 Fichier de paramètres du scénario

Le fichier "SourceSink.xml" décrit l'ensemble des sources (création des agents) et des puits (destruction des agents), ainsi que les informations liées.

```
<SourcesAndSinks>
<!-- Vehicules en stationnement -->
  <ParkedVehicles model="BUS">
    <Pos x="1720.3" y="2561.5" z="36.9" />
    ...
    <Pos x="1845.2" y="2448.8" z="36.7" />
  </ParkedVehicles>

  <ParkedVehicles model="Cars">
    <Pos x="1922.726162" y="2483.030099" z="36.354409" />
    ...
    <Pos x="1893.184139" y="2509.634816" z="36.132822" />
  </ParkedVehicles>

<!-- Positions des puits et sources de vehicules -->
  <PositionTable>
    <Position id="P1" x="2145.610000" y="2439.38" z="35.81" />
    ...
    <Position id="P99" x="1948.300000" y="2396.80" z="36.85" />
  </PositionTable>

<!-- VEHICULES LEGERS -->
  <Sink id="P1vc" pos="PP1v"> <Cylinder r="7.00" h="10.00"/> </Sink>
  ...
  <Sink id="P9vc" pos="PP9v"> <Cylinder r="7.00" h="10.00"/> </Sink>

...

<!-- Positions des puits et sources de pietons -->
  <PositionTable>
    <Position id="p1p" x="2195.300000" y="2753.00" z="41.20" />
    ...
    <Position id="p99p" x="1762.76" y="2667.01" z="35.36" />
  </PositionTable>

  <Sink id="P1p"  pos="pP1p">  <Cylinder r="1.5" h="10.00"/> </Sink>
  ...
  <Sink id="P15p" pos="pP15p"> <Cylinder r="1.5" h="10.00"/> </Sink>

  <Source name="S1p" pos="p1p" model="Touriste" unit="h" freq="60" >
    <DestSink sink="P1p"  p="25"/>
    <DestSink sink="P2p"  p="55"/>
    <DestSink sink="P3p"  p="10"/>
    <DestSink sink="P5p"  p="92"/>
    <DestSink sink="P6p"  p="36"/>
    <DestSink sink="P9p"  p="10"/>
    <DestSink sink="P10p" p="25"/>
    <DestSink sink="P12p" p="109"/>
  </Source>

  ...

  <Source name="S2p" pos="S2" model="Passant" unit="h" freq="300" >
    <DestSink sink="P1p"  p="25"/>
    <DestSink sink="P2p"  p="55"/>
    <DestSink sink="P3p"  p="10"/>
    <DestSink sink="P5p"  p="92"/>
    <DestSink sink="P6p"  p="36"/>
    <DestSink sink="P9p"  p="10"/>
    <DestSink sink="P10p" p="25"/>
  </Source>

  <Limits>
    <Limit model="Passant" min="100" max="500" />
    <Limit model="Touriste" min="100" max="500" />
    <Limit model="Cars" min="3000" max="3010" />
  </Limits>

</SourcesAndSinks>
```

"ParkedVehicles" indique les positions des véhicules de type "model" garées dans l'environnement au début de la simulation.

Une source d'agents est caractérisée par un nom ("name"), une position ("pos"), le nom du type des agents créés ("model"), une fréquence ("freq") indiquée en fonction de l'unité "unit" (h pour heure, m pour minute). Ensuite une liste de puits ("DestSink") indique les destinations originales des agents.

Annexe B

Exemple de graphe comportemental complet

Remarque sur le graphe : Pour des raisons de mise en page, la graphe est coupé en deux. Les actions situées sous le trait de séparation (de "retirerArgent" à "éteindreFeu") se trouvent en réalité sur la même ligne que les actions situées au dessus du trait (de "mangerResto" à "protégerRessource").

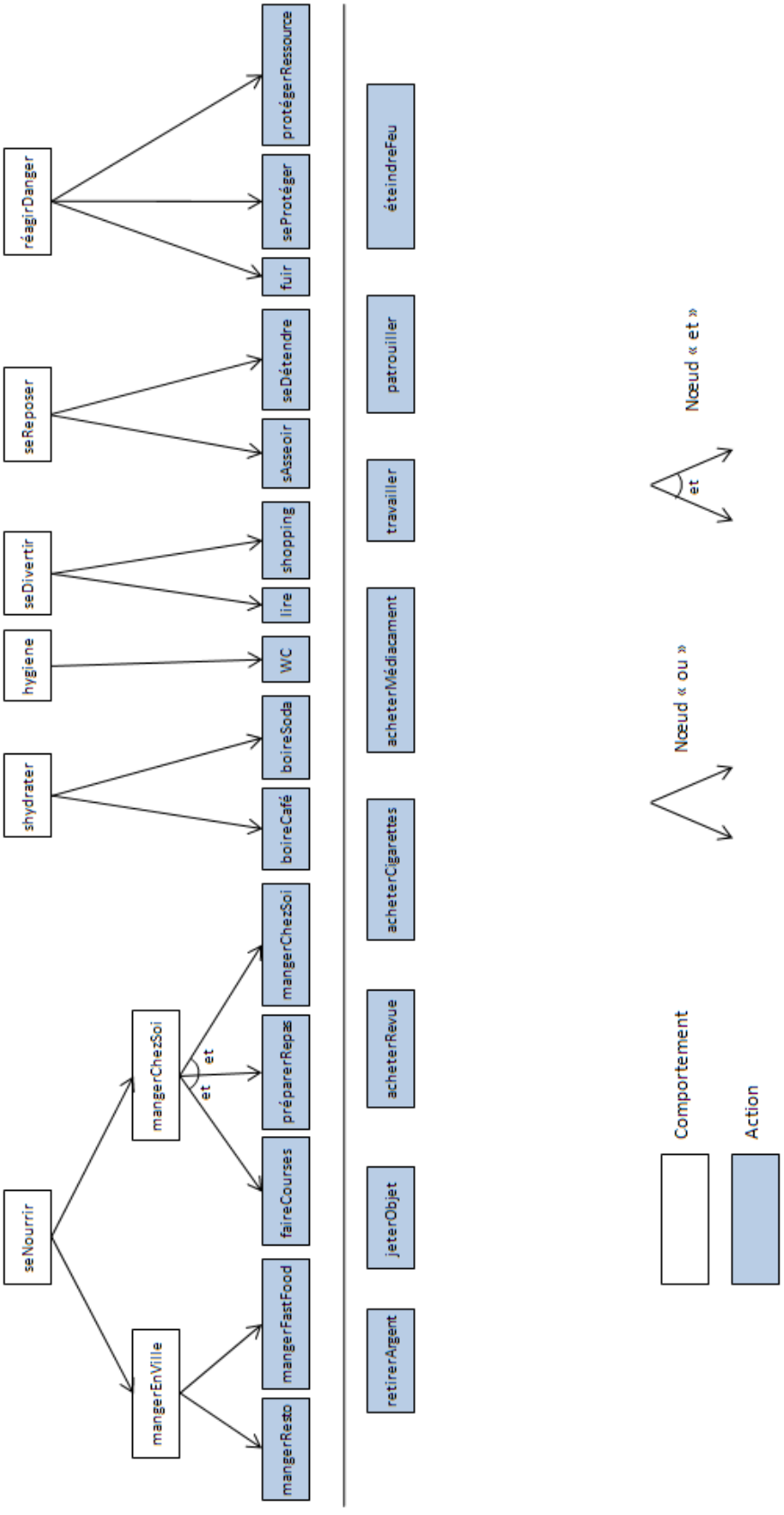

FIGURE B.1 – Exemple de graphe comportemental complet

Bibliographie

[Alcock 2001] J Alcock et P Farley. Animal behavior : an evolutionary approach. 2001. (Cité en page 68.)

[Allen 2007] IE Allen et CA Seaman. *Likert scales and data analyses*. Quality Progress, 2007. (Cité en page 155.)

[Almeida 2004] A Almeida, Geber Ramalho et H Santana. *Recent advances on multi-agent patrolling*. In Advances in Artificial Intelligence, pages 474–483. Springer Berlin Heidelberg, 2004. (Cité en page 90.)

[Amouroux 2009] E Amouroux, TQ Chu, A Boucher et A Drogoul. *GAMA : an environment for implementing and running spatially explicit multi-agent simulations*. In Aditya Ghose, Guido Governatori et Ramakoti Sadananda, editeurs, Agent Computing and Multi-Agent Systems, pages 359–371. Springer Berlin Heidelberg, 2009. (Cité en page 14.)

[Amouroux 2013] E Amouroux, T Huraux et F Sempé. *Simulating human activities to investigate household energy consumption*. In International Conference on Agents and Artificial Intelligence (ICAART), 2013. (Cité en page 14.)

[Anderson 2004] John R Anderson, Daniel Bothell, Michael D Byrne, Scott Douglass, Christian Lebiere et Yulin Qin. *An integrated theory of the mind*. psychological review, vol. 111, pages 1036–1060, 2004. (Cité en page 29.)

[Andrade 2005] G Andrade, Geber Ramalho, H Santana et Vincent Corruble. *Automatic computer game balancing : a reinforcement learning approach*. In Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, pages 1111–1112. ACM, 2005. (Cité en page 53.)

[Andriamasinoro 2003] Fenintsoa Andriamasinoro. *Proposition d'un modèle d'agents hybrides basé sur la motivation naturelle*. PhD thesis, Université de La Réunion, Août 2003. (Cité en pages 25 et 83.)

[Barbosa 2011] Pablo Barbosa, Danielle Silva, Geber Ramalho et Patricia Tedesco. *Simulating the Emergence of Social Relationship Networks in Groups of Believable Agents : The X-BARIM Model*. In Antônio Carlos Rocha Costa, Rosa Maria Vicari et Flavio Tonidandel, editeurs, Advances in Artificial Intelligence, volume 6404 of *Lecture Notes in Computer Science*, pages 133–142. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. (Cité en page 2.)

[Bates 1994] Joseph Bates. *The role of emotion in believable agents*. Communications of the ACM, vol. 37, no. 7, pages 122–125, Juillet 1994. (Cité en page 2.)

[Batty 2007] Michael Batty. Cities and Complexity : Understanding Cities with Cellular Automata, Agent-Based Models, and Fractals. The MIT Press, Septembre 2007. (Cité en page 15.)

[Batty 2013] Michael Batty. The New Science of Cities. MIT Press, Cambridge, MA, 2013. (Cité en page 15.)

[Blumberg 1996] BM Blumberg. *Old tricks, new dogs : ethology and interactive creatures*. PhD thesis, Massachusetts Institute of Technology, 1996. (Cité en pages 22 et 83.)

[Bonasso 1997] R Peter Bonasso et R James Firby. *Experiences with an architecture for intelligent, reactive agents*. Journal of Experimental & Theoretical Artificial Intelligence, vol. 9, no. 2-3, pages 237–256, 1997. (Cité en page 38.)

[Braubach 2005] L Braubach, A Pokahr et W Lamersdorf. *Jadex : A BDI-agent system combining middleware and reasoning*. Software Agent-Based Applications, Platforms and Development Kits, vol. Whitestein, pages 143–168, 2005. (Cité en page 33.)

[Brockington 2002] M Brockington. *Level-of-Detail AI for a Large Role-Playing Game*. In Steve Rabin, editeur, AI Game Programming Wisdom, pages 419–425. Charles River Media, Inc., 2002. (Cité en page 53.)

[Brooks 1985] Rodney A Brooks. *A Robust Layered Control System For a Mobile Robot*. Rapport technique, Massachusetts Institute of Technology, Cambridge, MA, USA, 1985. (Cité en page 16.)

[Bryson 2000] J Bryson. *Cross-Paradigm Analysis of Autonomous Agent Architecture*. Journal of Experimental and Theoretical Artificial Intelligence, vol. 12, no. 2, pages 165–190, 2000. (Cité en page 15.)

[Butz 2003a] Martin V. Butz, O Sigaud et P Gerard. *Internal models and anticipations in adaptive learning systems*. In Anticipatory behavior in adaptive learning systems, volume LNAI 2684. 2003. (Cité en page 46.)

[Butz 2003b] Martin V. Butz, Olivier Sigaud et Pierre Gérard. Anticipatory Behavior in Adaptive Learning Systems : Foundation, Theories, and Systems, volume 2684 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. (Cité en page 114.)

[Butz 2007a] Martin V. Butz, Olivier Sigaud, Giovanni Pezzulo et Gianluca Baldassarre. *Anticipations, brains, individual and social behavior : An introduction to anticipatory systems*. In Anticipatory Behavior in Adaptive Learning Systems : From Psychological Theories to Artificial Cognitive Systems, pages 1–18. 2007. (Cité en page 114.)

[Butz 2007b] Martin V. Butz, Olivier Sigaud, Giovanni Pezzulo et Gianluca Baldassarre. Anticipatory Behavior in Adaptive Learning Systems : From Brains to Individual and Social Behavior. 2007. (Cité en page 114.)

[Campano 2012] Sabrina Campano, Nicolas Sabouret, Etienne de Sevin et Vincent Corruble. *The "Resource" Approach to Emotion*. In Autonomous Agents and MultiAgent Systems, Valencia, Spain, 2012. (Cité en page 89.)

[Campano 2013a] Sabrina Campano. *COR-E : un modèle pour la simulation d'agents affectifs fondé sur la théorie COR*. PhD thesis, Université Pierre et Marie Curie, 2013. (Cité en page 89.)

[Campano 2013b] Sabrina Campano, Nicolas Sabouret, Etienne De Sevin et Vincent Corruble. *An evaluation of the COR-E computational model for affective behaviors.* Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems, pages Pages 745–752, 2013. (Cité en pages 89 et 138.)

[Capdepuy 2007] P Capdepuy, D Polani et CL Nehaniv. *Construction of an internal predictive model by event anticipation.* Anticipatory Behavior in Adaptive learning systems, 2007. (Cité en pages 47 et 122.)

[Carpenter 1991] GA Carpenter, S Grossberg et JH Reynolds. *ARTMAP : Supervised real-time learning and classification of nonstationary data by a self-organizing neural network.* Neural networks, vol. 4, pages 565–588, 1991. (Cité en page 45.)

[Cassimatis 2005] Nicholas L. Cassimatis. *Integrating Cognitive Models Based on Different Computational Methods.* In Proceedings of the Twenty-Seventh Annual Conference of the Cognitive Science Society, 2005. (Cité en page 41.)

[Castelfranchi 2006] C Castelfranchi. *Agents with Anticipatory Behaviors : To Be Cautious in a Risky Environment.* In Proc. of European Conf. on Artificial Intelligence, pages 538—-546, 2006. (Cité en page 47.)

[Cavazza 2002] Marc Cavazza, F Charles et SJ Mead. *Interacting with virtual characters in interactive storytelling.* In Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems : Part 1, pages 318–325, Bologna, Italy, 2002. ACM. (Cité en pages 53 et 193.)

[Chauvin 2014] Simon Chauvin, Guillaume Levieux, Jean-Yves Donnart et Stéphane Natkin. *An Out of Character Approach to Emergent Game Narratives.* In Foundations of Digital Games, 2014. (Cité en page 193.)

[Coleridge 1817] Samuel Taylor Coleridge. Biographia Literaria. 1817. (Cité en page 2.)

[Davidsson 1994] Paul Davidsson, E Astor et B Ekdahl. *A framework for autonomous agents based on the concept of anticipatory systems.* Cybernetics and Systems, 1994. (Cité en pages 117, 123 et 188.)

[Davidsson 2003] Paul Davidsson. *A Framework for Preventive State Anticipation.* In Anticipatory Behavior in Adaptive Learning Systems, pages 151–166. Lecture Notes in Computer Science, 2003. (Cité en pages 47 et 115.)

[de Sevin 2006] Etienne de Sevin. *An action selection architecture for autonomous virtual humans in persistent worlds.* PhD thesis, Ecole polytechnique fédérale de Lausanne, 2006. (Cité en pages 25, 83 et 84.)

[de Sevin 2012a] Etienne de Sevin, Quentin Reynaud et Vincent Corruble. *FlexMex : Flexible Multi-Expert Meta-Architecture for Virtual Agents.* In First Annual Conference on Advances in Cognitive Systems, 2012. (Cité en pages 71, 72 et 90.)

[de Sevin 2012b] Etienne de Sevin, Quentin Reynaud et Vincent Corruble. *Flex-Mex : Flexible Multi-Expert Meta-Architecture for Virtual Agents*. In Advances in Cognitive Systems, Palo Alto, USA, 2012. (Cité en page 189.)

[Depristo 2001] Mark A Depristo, Hughes Hall et Robert Zubek. *Being-in-the-World*, 2001. (Cité en page 38.)

[Devigne 2007] D Devigne. *Approche centrée interaction pour la simulation d'agents cognitifs situés*. PhD thesis, Université Lille1, 2007. (Cité en page 39.)

[Doniec 2008] A Doniec, R Mandiau, S Piechowiak et Espié S. *Anticipation based on constraint processing in a multi-agent context*. Autonomous Agents and Multi-Agent Systems, vol. 17, no. 2, pages 339–361, 2008. (Cité en pages 47 et 123.)

[Donnart 1996] Jean-Yves Donnart et J.A. Meyer. *Learning reactive and planning rules in a motivationally autonomous animat*. IEEE Transactions on Systems, Man, and Cybernetics, vol. 26, no. 3, pages 381–395, 1996. (Cité en pages 22 et 83.)

[Donnart 1998] Jean-Yves Donnart. *Architecture cognitive et proprietes adaptatives d'un animat motivationnellement autonome*. PhD thesis, Université Paris VI, 1998. (Cité en pages 22 et 66.)

[Dorigo 1993] M Dorigo et U Schnepf. *Genetics-based machine learning and behavior-based robotics : a new synthesis*. Systems, Man and Cybernetics, IEEE Transactions on, vol. 23, no. 1, pages 141–154, 1993. (Cité en page 22.)

[Drogoul 1993] A Drogoul. *De la simulation multi-agents à la résolution collective de problèmes*. PhD thesis, Université Paris VI, 1993. (Cité en pages 14 et 15.)

[Drogoul 2003] A Drogoul, D Vanbergue et T Meurisse. *Multi-agent based simulation : Where are the agents ?* In Multi-agent-based simulation II, pages 1–15. Springer Berlin Heidelberg, 2003. (Cité en page 14.)

[Duch 2008] W. Duch, R.J. Oentaryo et M. Pasquier. *Cognitive Architectures : Where do we go from here ?* In Proceeding of the 2008 conference on Artificial General Intelligence, pages 122–136. IOS Press, 2008. (Cité en pages 8 et 15.)

[Dujardin 2010a] Tony Dujardin. *Construction d'individualité par un mécanisme de sélection d'action basé sur les motivations*. PhD thesis, Université Lille 1, 2010. (Cité en pages 77, 83 et 101.)

[Dujardin 2010b] Tony Dujardin et Jean-Christophe Routier. *Behavior Design of Game Character's Agent*. In 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pages 423–430. IEEE, Août 2010. (Cité en page 39.)

[Edward 2010] Lydie Edward, Domitile Lourdeaux et Jean-Paul Barthès. *Une architecture cognitive pour la simulation de comportements d'agents autonomes. Intégration de facteurs physiques, physiologiques et de personnalité*. Revue d'Intelligence Artificielle, vol. 24, no. 5, pages 673–694, 2010. (Cité en page 34.)

[Erman 1980] LD Erman et F Hayes-Roth. *The Hearsay-II speech-understanding system : Integrating knowledge to resolve uncertainty.* ACM Computing Surveys ..., 1980. (Cité en pages 21 et 192.)

[Erol 1996] K. Erol. *Hierarchical Task Network Planning : Formalization, Analysis, and Implementation*, 1996. (Cité en page 30.)

[Evans 2009] Richard Evans. *AI Challenges in Sims 3.* In Artificial Intelligence for Interactive Digital Entertainment, 2009. (Cité en page 53.)

[Ferguson 1992] Innes A Ferguson. *Touring Machines : Autonomous Agents with Attitudes.* Computer, vol. 25, no. 5, pages 51–55, 1992. (Cité en page 35.)

[Ferrell 1993] C Ferrell. *Robust agent control of an autonomous robot with many sensors and actuators.* PhD thesis, Massachusetts Institute of Techology, 1993. (Cité en page 67.)

[Fikes 1972] RE Fikes et NJ Nilsson. *STRIPS : A new approach to the application of theorem proving to problem solving.* Artificial intelligence, 1972. (Cité en pages xi, 27, 28 et 83.)

[Fleischer 2003] J Fleischer, S Marsland et J Shapiro. *Sensory anticipation for autonomous selection of robot landmarks.* In Anticipatory Behavior in Adaptive Learning Systems : Foundations, Theories, and Systems, pages 201–221. 2003. (Cité en page 47.)

[Garcia 2000] Carlos Gershenson Garcia, Pedro Pablo Gonzalez Perez et Jose Negrete Martinez. *Action Selection Properties in a Software Simulated Agent.* In Proceedings of the Mexican International Conference on Artificial Intelligence : Advances in Artificial Intelligence, pages 634–648, London, UK, 2000. Springer-Verlag. (Cité en page 21.)

[Georgeff 1987] MP Georgeff et AL Lansky. *Reactive reasoning and planning.* In Association for the Advancement of Artificial Intelligence, 1987. (Cité en page 31.)

[Georgeff 1999] Michael P Georgeff, B. Pell, M. Pollack, M. Tambe et Michael Wooldridge. *The belief-desire-intention model of agency.* Intelligent Agents V : Agents Theories, Architectures, and Languages, pages 1–10, 1999. (Cité en page 32.)

[Goldberg 1988] DE Goldberg et JH Holland. *Genetic algorithms and machine learning.* Machine learning, vol. 3, no. 2, pages 95–99, 1988. (Cité en page 22.)

[Guillot 2001] Agnès Guillot. *The animat contribution to cognitive systems research.* Cognitive Systems Research, vol. 2, no. 2, pages 157–165, Mai 2001. (Cité en page 22.)

[Harkouken-Saiah 2012] Kenza Harkouken-Saiah, Nicolas Sabouret et Jean-Yves Donnart. *Modélisation de l'information sémantique pour la simulation d'un environnement virtuel sémantique urbain.* In 23 ème Journées Francophones d'Ingénierie des Connaissances, 2012. (Cité en pages 103 et 106.)

[Harkouken 2013] K Harkouken. *An Ontology for Semantic Representation of an Urban Virtual Environment.* In The International Conference on Artificial Intelligence, 2013. (Cité en page 177.)

[Hawes 2007] Nick Hawes, Aaron Sloman, Wyatt Jeremy et Michael Zillich. *Towards an Integrated Robot with Multiple Cognitive Functions.* In Proceedings of the 22nd national conference on Artificial intelligence - Volume 2, pages 1548—-1553, Vancouver, British Columbia, Canada, 2007. AAAI Press. (Cité en page 41.)

[Holland 1977] JH Holland et JS Reitman. *Cognitive systems based on adaptive algorithms.* ACM SIGART Bulletin, pages 49–49, 1977. (Cité en page 22.)

[Holmes 2002] JH Holmes et PL Lanzi. *Learning classifier systems : New models, successful applications.* Information Processing Letters, vol. Volume 82, pages Pages 23–30, 2002. (Cité en page 46.)

[Huber 1999] MJ Huber. *JAM : A BDI-theoretic mobile agent architecture.* In Proceedings of the third annual conference on Autonomous Agents, pages 236–243, 1999. (Cité en page 33.)

[Jones 2009] Hazaël Jones, Julien Saunier et Domitile Lourdeaux. *Personality, Emotions and Physiology in a BDI Agent Architecture : The PEP-> BDI Model.* In The 2009 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pages 263–266, Milan, 2009. (Cité en page 34.)

[Kaelbling 1996] LP Kaelbling, ML Littman et AW Moore. *Reinforcement learning : A survey.* Journal of Artificial Intelligence Research, 1996. (Cité en pages 45, 46 et 111.)

[Kelly 2008] JP Kelly, A Botea et S Koenig. *Offline Planning with Hierarchical Task Networks in Video Games.* In AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2008. (Cité en page 30.)

[Kubera 2011] Y Kubera, P Mathieu et S Picault. *IODA : an interaction-oriented approach for multi-agent based simulations.* In Autonomous Agents and Multi-Agent Systems, pages 303–343, 2011. (Cité en pages 2 et 40.)

[Laird 1987] J. E. Laird, Allen Newell et Paul S Rosenbloom. *SOAR : an architecture for general intelligence.* Artif. Intell., vol. 33, no. 1, pages 1–64, 1987. (Cité en page 28.)

[Laird 2001] J. E. Laird. *It knows what you're going to do : adding anticipation to a Quakebot.* Proceedings of the fifth international conference on Autonomous agents, pages Pages 385–392, 2001. (Cité en page 47.)

[Laird 2012] J. E. Laird. The Soar cognitive architecture. 2012. (Cité en page 28.)

[Langley 2006] P. Langley et Dongkyu Choi. *A unified cognitive architecture for physical agents.* In proceedings of the 21st national conference on Artificial intelligence - Volume 2, pages 1469–1474. AAAI Press, 2006. (Cité en page 39.)

[Langton 1997] CG Langton. Artificial life : An overview. 1997. (Cité en page 2.)

[Lind 1999] J Lind. *MASSIVE : Software Engineering for Multiagent Systems.* 1999. (Cité en page 51.)

[Lorenz 1985] Konrad Lorenz. *My family and other animals.* In Leaders in the study of animal behavior : autobiographical perspectives, pages 259–287. 1985. (Cité en page 68.)

[Lugrin 2010] Jean-Luc Lugrin, Marc Cavazza, David Pizzi, Thurid Vogt et Elisabeth André. *Exploring the usability of immersive interactive storytelling.* In Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology - VRST '10, pages 103–110, New York, New York, USA, Novembre 2010. ACM Press. (Cité en page 193.)

[Machado 2003] A Machado et Geber Ramalho. *Multi-agent patrolling : An empirical analysis of alternative architectures.* In Multi-Agent-Based Simulation II, volume 2581, pages 155–170. Springer Berlin Heidelberg, 2003. (Cité en page 90.)

[Maes 1989] P Maes. *The dynamics of action selection.* In International Joint Conferences on Artificial Intelligence, pages 991–997, 1989. (Cité en page 68.)

[Maes 1990] Pattie Maes. *Situated agents can have goals.* Robotics and Autonomous Systems, vol. 6, no. 1-2, pages 49–70, Juin 1990. (Cité en page 18.)

[Maim 2009] J Maim et B Yersin. *Yaq : An architecture for real-time navigation and rendering of varied crowds.* Computer Graphics and Applications, vol. 29, no. 4, pages 44–53, 2009. (Cité en page 51.)

[Maslow 1943] AH Maslow. *A theory of human motivation.* Psychological review, 1943. (Cité en pages 22, 25 et 83.)

[McFarland 1993] David McFarland et Tom Bösser. Intelligent behavior in animals and robots. MIT Press, 1993. (Cité en page 83.)

[Meyer 1996] J.A. Meyer. *Artificial Life and the Animat Approach to Artificial Intelligence.* Artificial Intelligence, Boden,(Ed), pages 325–354, 1996. (Cité en page 22.)

[Millington 2009] I Millington et J Funge. Artificial intelligence for games. 2009. (Cité en page 63.)

[Minsky 1965] M Minsky. *Matter, mind and models.* 1965. (Cité en page 1.)

[Miranda 2011] J Martínez Miranda. *Modelling human behaviour at work : an agent-based simulation to sopport the configuration of work teams.* PhD thesis, Universidad Complutense de Madrid, 2011. (Cité en page 15.)

[Mori 1970] M Mori. *The uncanny valley.* Energy, 1970. (Cité en pages viii et 160.)

[Müller 1993] Jörg P Müller et Markus Pischel. *The Agent Architecture InteRRaP : Concept and Application.* Rapport technique, DFKI, 1993. (Cité en pages 35 et 192.)

[Navarro 2011] Laurent Navarro, Fabien Flacher et Vincent Corruble. *Dynamic Level of Detail for Large Scale Agent-Based Urban Simulations*. In The Tenth International Conference on Autonomous Agents and Multiagent Systems, pages 701–708. International Foundation for Autonomous Agents and Multiagent Systems, 2011. (Cité en pages 54 et 110.)

[Navarro 2013a] Laurent Navarro. *Niveau de Détail Dynamique et Progressif de l'Intelligence Artificielle pour le passage à l'Echelle dans la Simulation de Comportements Humains à Base d'Agents*. PhD thesis, Université Pierre et Marie Curie, 2013. (Cité en pages 82 et 170.)

[Navarro 2013b] Laurent Navarro, Vincent Corruble, Fabien Flacher et Jean-Daniel Zucker. *A flexible approach to multi-level agent-based simulation with the mesoscopic representation*. In Ito, Jonker, Gini et Shehory, editeurs, Proceedings of the Twelfth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), pages 159–166, Saint Paul, Minnesota, USA, 2013. International Foundation for Autonomous Agents and Multiagent Systems. (Cité en pages 110 et 163.)

[Newell 1961] Allen Newell, H.A. Simon et rand corp Santa Monica Calif. GPS, a program that simulates human thought. Defense Technical Information Center, 1961. (Cité en page 27.)

[Newell 1972] Allen Newell et Simon H A. *Human problem solving*. 1972. (Cité en page 27.)

[Nuttin 1985] J Nuttin. *Théorie de la motivation humaine : du besoin au projet d'action*. Presses Universitaires de France PUF (Coll. Psychologie d'aujourd'hui), page 383, 1985. (Cité en page 22.)

[Ochs 2009] M. Ochs, Nicolas Sabouret et Vincent Corruble. *Simulation of the Dynamics of Nonplayer Characters' Emotions and Social Relations in Games*. IEEE Transactions on Computational Intelligence and AI in Games, vol. 1, no. 4, pages 281–297, Décembre 2009. (Cité en page 89.)

[Orkin 2006] J Orkin. *Three states and a plan : the AI of FEAR*. In Game Developers Conference, 2006. (Cité en page 52.)

[Öztürk 2009] P Öztürk. *Levels and types of action selection : The action selection soup*. Adaptive behavior, 2009. (Cité en page 35.)

[Pan 2006] X Pan, C Han, KH Law et JC Latombe. *A computational framework to simulate human and social behaviors for egress analysis*. In Joint International Conference on Computing and Decision Making in Civil and Building Engineering, 2006. (Cité en page 14.)

[Pew 1998] RW Pew et AS Mavor. Modeling human and organizational behavior : Application to military simulations. 1998. (Cité en page 15.)

[Pezzulo 2008] Giovanni Pezzulo, Martin V. Butz, Christiano Castelfranchi et Rino Falcone. The Challenge of Anticipation : A Unifying Framework for the Analysis and Design of Artificial Cognitive Systems. Springer, 2008. (Cité en pages 9 et 114.)

[Pezzulo 2009a] Giovanni Pezzulo, Martin V. Butz, Olivier Sigaud et Gianluca Baldassarre. Anticipatory Behavior in Adaptive Learning Systems : From Psychological Theories to Artificial Cognitive Systems. 2009. (Cité en page 187.)

[Pezzulo 2009b] Giovanni Pezzulo, Martin V. Butz, Olivier Sigaud et Gianluca Baldassarre. *From Sensorimotor to Higher-Level Cognitive Processes : An Introduction to Anticipatory Behavior Systems.* In Anticipatory Behavior in Adaptive Learning Systems : From Psychological Theories to Artificial Cognitive Systems, pages 1–9. 2009. (Cité en page 114.)

[Piunti 2007] M Piunti, C Castelfranchi et R Falcone. *Surprise as Shortcut for Anticipation : Clustering Mental States in Reasoning.* International Joint Conferences on Artificial Intelligence, 2007. (Cité en page 47.)

[Porteous 2010] J Porteous, Marc Cavazza et F Charles. *Applying planning to interactive storytelling : Narrative control using state constraints.* ACM Transactions on Intelligent Systems and Technology, vol. 1, no. 2, 2010. (Cité en page 53.)

[Porteous 2013] J Porteous, F Charles et Marc Cavazza. *NetworkING : using character relationships for interactive narrative generation.* In Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems, pages 595–602, St. Paul, MN, USA, 2013. International Foundation for Autonomous Agents and Multiagent Systems. (Cité en page 193.)

[Poulet 2012a] Cyril Poulet, Vincent Corruble et Amal El Fallah Seghrouchni. *Auction-based strategies for the open-system patrolling task.* In Iyad Rahwan, Wayne Wobcke, Sandip Sen et Toshiharu Sugawara, editeurs, PRIMA - 15th International Conference on Principles and Practice of Multi-Agent Systems, Proceedings, pages 92–106. Springer, 2012. (Cité en pages 90 et 138.)

[Poulet 2012b] Cyril Poulet, Vincent Corruble et Amal El Fallah Seghrouchni. *Working as a team : using social criteria in the timed patrolling problem.* In Tools with Artificial Intelligence (ICTAI), 2012 24th IEEE International Conference on. IEEE, 2012. (Cité en page 90.)

[Poulet 2013] Cyril Poulet. *Coordination dans les systèmes multi-agents : Le problème de la patrouille en système ouvert.* PhD thesis, Université Pierre et Marie Curie, 2013. (Cité en pages 90 et 191.)

[Railsback 2006] SF Railsback, SL Lytinen et SK Jackson. *Agent-based simulation platforms : Review and development recommendations.* Simulation, vol. 82, no. 9, pages 609–623, 2006. (Cité en page 14.)

[Rao 1995] Anand S Rao et Michael P Georgeff. *BDI Agents : From Theory to Practice.* In in proceedings of the first international conference on multi-agent systems (ICMAS-95), pages 312–319, 1995. (Cité en page 71.)

[Reynaud 2012] Quentin Reynaud, Etienne De Sevin, Jean-Yves Donnart et Vincent Corruble. *A cognitive module in a decision-making architecture for agents in urban simulations.* In AAMAS 2012 Workshop on Cognitive

Agents in Virtual Environments (CAVE), numéro 1, Valencia, Spain, 2012. (Cité en page 131.)

[Reynaud 2013] Quentin Reynaud et Vincent Corruble. *Un mécanisme de composition de comportements pour agents virtuels.* In Journées Françaises sur les Systèmes Multi-Agents, 2013. (Cité en page 110.)

[Reynaud 2014] Quentin Reynaud, Jean-Yves Donnart et Vincent Corruble. *Evaluating the Efficiency and Believability of Virtual Agents with Anticipatory Abilities.* In Autonomous Agents and Multi-Agent Systems, Paris, France, 2014. (Cité en pages 161 et 188.)

[Riegler 2001] A Riegler. *The role of anticipation in cognition.* In Proceedings of the Fourth International Conference on Computing Anticipatory Systems, pages 534—-541, 2001. (Cité en page 45.)

[Riegler 2003] A Riegler. *Whose anticipations ?* In Martin V. Butz, Olivier Sigaud et Pierre Gérard, editeurs, Anticipatory Behavior in Adaptive Learning Systems, volume 2684, pages pp 11–22. Springer Berlin Heidelberg, lecture no édition, 2003. (Cité en page 45.)

[Robert 2003] G. Robert et Agnès Guillot. *MHiCS, a modular and hierarchical classifier systems architecture for bots.* In 4th International Conference on Intelligent Games and Simulation, pages 140–144, 2003. (Cité en page 24.)

[Rosen 2012] R Rosen. Anticipatory systems. Springer, 2nd ed édition, 2012. (Cité en page 45.)

[Rosenblatt 1989] J.K. Rosenblatt et D.W. Payton. *A fine-grained alternative to the subsumption architecture for mobile robot control.* In International Joint Conference on Neural Networks, pages 317–323 vol.2. IEEE, 1989. (Cité en page 18.)

[Rosenblatt 1995] Julio Rosenblatt. *DAMN : A Distributed Architecture For Mobile Navigation - Thesis Summary.* In Journal of Experimental and Theoretical Artificial Intelligence, pages 339–360. AAAI Press, 1995. (Cité en pages 19 et 71.)

[Russell 2010] S.J. Russell et P. Norvig. Artificial intelligence : a modern approach. Prentice hall, 2010. (Cité en pages 1, 4, 5, 6, 60 et 186.)

[Schmidt 2005] Bernd Schmidt. *Human factors in complex systems : The modelling of human behaviour.* In Simulation in wider Europe, 19th European Conferance on Modelling and Simulation, pages 5–14, 2005. (Cité en pages 37 et 72.)

[Sengers 1999] Phoebe Sengers. *Designing comprehensible agents.* In Proceedings of the 16th international joint conference on Artificial intelligence - Volume 2, pages 1227–1232, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. (Cité en page 2.)

[Seth 1998] Anil K Seth. *Evolving action selection and selective attention without actions, attention, or selection.* In Proceedings of the fifth international

conference on simulation of adaptive behavior, pages 139–146, Cambridge, MA, USA, 1998. MIT Press. (Cité en page 2.)

[Shao 2007] Wei Shao et Demetri Terzopoulos. *Autonomous pedestrians*. Graphical Models, 2007. (Cité en pages 50 et 51.)

[spi 2005] *http ://www.spirops.com/SpirOpsAI.php*, 2005. (Cité en page 52.)

[Stolzmann 1997] Wolfgang Stolzmann. Antizipative Classifier Systems. Shaker, 1997. (Cité en page 45.)

[Student 1908] Student. *The probable error of a mean*. Biometrika, 1908. (Cité en page 153.)

[Sutton 1991] RS Sutton. *Reinforcement learning architectures for animats*. In From Animals to Animats : Proceedings of the First International Conference on Simulation of Adaptive Behavior, pages 288–296, 1991. (Cité en page 45.)

[Sutton 1998] RS Sutton et AG Barto. *Reinforcement learning : An introduction*. In Jonathan H. Connell et Sridhar Mahadevan, editeurs, Robotica, pages 229–235. Cambridge University Press, Boston, 1998. (Cité en pages 46 et 111.)

[Thalmann 2009] Daniel Thalmann, Helena Grillon, Jonathan Maim et Barbara Yersin. *Challenges in crowd simulation*. In International Conference on CyberWorlds, 2009, pages 1–12, Bradford, 2009. (Cité en page 50.)

[Tisue 2004] S Tisue et U Wilensky. *NetLogo : A simple environment for modeling complexity*. In International Conference on Complex Systems, Boston, 2004. (Cité en page 14.)

[Tyrrell 1993a] Toby Tyrrell. *Computational Mechanisms for Action Selection*. PhD thesis, University of Edinburgh, 1993. (Cité en pages 43, 72, 94, 135 et 188.)

[Tyrrell 1993b] Toby Tyrrell. *The use of hierarchies for action selection*. Adaptive Behavior, vol. 1, no. 4, page 387, 1993. (Cité en page 18.)

[Vanbergue 2002] D Vanbergue et A Drogoul. *Approche multi-agent pour la simulation urbaine*. Actes des 6ème Journées CASSINI, pages 95–112, 2002. (Cité en pages 2, 3 et 15.)

[Wilson 1995] SW Wilson. *Classifier fitness based on accuracy*. Evolutionary computation, vol. 3, no. 2, pages 149–175, 1995. (Cité en page 22.)

[Wissner 2010] M Wissner, F Kistler et E André. *Level of detail ai for virtual characters in games and simulation*. Motion in Games, 2010. (Cité en pages 53, 80 et 187.)

[Wooldridge 1995] Michael Wooldridge et NR Jennings. *Agent theories, architectures, and languages : a survey*. Intelligent agents, 1995. (Cité en page 15.)

[Yadav 2011] Nitin Yadav et Sebastian Sardina. *Decision theoretic behavior composition*. In The 10th International Conference on Autonomous Agents and Multiagent Systems, Taipei, Taiwan, 2011. International Foundation for Autonomous Agents and Multiagent Systems. (Cité en page 43.)