République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université ABBES LAGHROUR Khenchela

Ecole doctorale STIC

Sciences et Technologies de l'Information et de la Communication

Mémoire
Pour l'Obtention du Diplôme de Magister En informatique

Option : système d'information et de connaissances

Thème

Une extension de l'architecture BDI : prise en considération du contexte et le partage des croyances

Présenté Par : BECHOUA Khaled

Soutenue le : 02/03/2016

Composition du jury :

Pr. BENMOHAMMED Mohamed	Université Constantine 2-Abdelhamid Mehri	**Président**
Pr MAAMRI Ramdane	Université Constantine 2-Abdelhamid Mehri	**Rapporteur**
Dr. MERAH Elkamel	Université Abbès Laghrour Khenchela	**Examinateur**
Dr. ZITOUNI Abdelhafid	Université Constantine 2-Abdelhamid Mehri	**Examinateur**

2016

DÉDICACE

Je dois remercier Dieu de m'avoir donné le courage pour accomplir ce modeste travail que je dédie à :

Mes très chers parents : Fateh et Zohra

Ma femme : Imène

Mes frères : Walid et Aimen

Mes sœurs : Khadidja et Rahma

Mes amis : Samir, Raouf, Khalil

Toute ma grande famille, tous mes amis et tous mes enseignants

Tous ceux que j'aime et qui m'aiment.

Sans oublier nos frères Palestiniens.

REMERCIEMENTS

Tout d'abord, je présente mes remerciements les plus sincères à l'initiateur de ce travail : Le Professeur **MAAMRI Ramdane**.

Je tiens à remercier le professeur **MAAMRI Ramdane** pour son encadrement et pour l'encouragement et l'intérêt qu'il m'a apportés pour l'accomplissement de ce travail et surtout pour sa grande aide et ses qualités humaines.

Je voudrais également remercier Messieurs, les membres du jury d'avoir accepté de juger mon travail :

- Monsieur **BENMOHAMMED Mohamed,** professeur à l'université Constantine 2-Abdelhamid Mehri qui m'a fait l'honneur de présider le jury de cette thèse.
- Monsieur **MERAH Elkamel**, Maître de conférence à l'université Abbès Laghrour Khenchela, Monsieur **ZITOUNI Abdelhafid**, Maître de conférence à l'université Constantine 2-Abdelhamid Mehri, qui m'ont fait l'honneur, de prendre ce travail en considération en tant qu'examinateurs de cette thèse.

Enfin, je remercie tous ceux qui m'ont aidé de près ou de loin dans ce travail.

Résumé

Les systèmes sensibles au contexte sont des systèmes capables de capter, représenter et gérer le contexte, leurs composants doivent également pouvoir raisonner et adapter leur comportement aux évolutions de leur environnement. Dans ce travail de recherche on s'intéresse à intégrer le concept de contexte aux agents BDI (**B**elief, **D**esire, **I**ntention) pour construire Une extension de l'architecture BDI capable de la prise en considération du contexte. L'architecture que nous avons proposé résoudre trois problèmes essentiels sont:

1. Comment représenter le contexte dans l'agent BDI ?
2. Comment mettre à jour la structure de données de contexte?
3. Comment le contexte affecte dans le processus de prise de décision?

La résolution de ces trois problèmes permet de construire des agents BDI plus sensible au contexte et d'améliorer leur fonctionnement dans les différents environnements

Mots clés : contexte, agent BDI, système multi-agent, système sensible au contexte

ABSTRACT

The Context-aware systems are systems that capture, represent and manage the context, their components must also be able to reason and adapt their behavior to evolution in their environment. In this research we focus on integrating the concept of the context in the BDI agents (Belief, Desire, Intention) to build an extension of the in the BDI architecture capable to take in consideration the context. The architecture that we proposed to solve three key problems are:

1. How to represent the context in the BDI agent?
2. How to update the context data's structure?
3. How does the context affect the process of making decision?

The resolution of these three problems allows building BDI agents more sensitive to context and improving their functioning in the different environments.

Key words: context, BDI agent, multi-agents system, context-aware systems.

ملخص

تعتبر الأنظمة الحساسة للسياق أنظمة قادرة على التقاط، تمثيل وتسيير السياق. أيضا مكوناتها قادرة على التفكير والتكيف مع تغيرات المحيط. في هذه المذكرة نهدف الى إضافة مفهوم السياق لوكلاء من نوع BDI (**B**elief, **D**esire, **I**ntention) وذلك بإنشاء امتداد لهيكل الوكيل BDI يكون قادرا على أخذ السياق بعين الاعتبار. الهيكل المقترح يحل المشاكل الثلاثة التالية:

1. كيفية تمثيل السياق في الوكيل BDI.
2. كيفية تحديث السياق في الوكيل BDI.
3. كيفية تأثير السياق على الوكيل BDI في اتخاذ القرارات.

حل هذه المشاكل الثلاثة يسمح لنا بإنشاء وكيل BDI أكثر حساسية للسياق ويحسن عمل الوكيل في مختلف المحيطات.

الكلمات المفتاحية: السياق، الوكيل BDI، الأنظمة متعددة الوكلاء، الأنظمة الحساسة للسياق.

Table des matières

Liste des figures :

Liste des tableaux :

Introduction générale

La notion d'agent et de système multi-agents (SMA) est relativement récente en informatique, mais elle tend à prendre de plus en plus d'importance. Ce paradigme est né au cours des années 80 de la rencontre de l'Intelligence artificielle et des Systèmes Distribués. Cette discipline est introduite pour remédier aux insuffisances et enrichir les approches classiques, et pour distribuer les connaissances et le contrôle dans les systèmes d'Intelligence Artificielle. Les Systèmes Multi-Agents (SMA) constituent actuellement une nouvelle approche pour la conception, la mise en œuvre, la réalisation des systèmes complexes ou la simulation et la compréhension des systèmes coopératifs, distribués et ouverts. En effet, le paradigme agent est à la connexion de plusieurs domaines comme le génie logiciel pour l'évolution vers des composants logiciels de plus en plus autonomes et proactifs, l'intelligence artificielle pour les aspects décisionnels de l'agent, et l'intelligence artificielle distribuée pour les interactions entre agents et la distribution de la résolution et de l'exécution. Notons que d'autres domaines tels que la vie artificielle, la biologie, la sociologie et les sciences cognitives.

Et dans cette communauté de recherche existe plusieurs modèle d'agent, Un des modèles les plus connues est sans doute le modèle BDI développé par plusieurs chercheurs dans les travaux suivant (Bratman [28], Cohen and Levesque [47], Rao and Georgeff [48], Sadek, [49], Konolige and Pollack [30]..etc), basé sur trois modalités primitives : croyance, désir et intentions en anglais (**B**elief, **D**esir, **I**ntention (BDI)) :

- **Croyance (Belief) :** Les croyances d'un agent sont les informations que l'agent possède sur l'environnement et sur d'autres agents qui existent dans le même environnement.
- **Désir (Desire) :** Les désirs d'un agent représentent les états de l'environnement, et parfois de lui-même, que l'agent aimerait voir réalisés.
- **Intention (Intention) :** Les intentions d'un agent sont les désirs que l'agent a décidé d'accomplir ou les actions qu'il a décidé de faire pour accomplir ses désirs.

Ce modèle est Le but de notre travail. Dans ce travaille nous essayons de proposer une extension des architectures BDI afin qu'elles puissent prendre en considération le contexte et

de partager les croyances pour améliorer son fonctionnement dans les différents environnements.

Plan du mémoire :

Ce mémoire est composé principalement de quatre chapitres, dont les deux premiers présentent un état de l'art du domaine et le troisième chapitre concerne nos contributions. Quant au quatrième chapitre nous présentons une étude de cas.

Chapitre 1 : Concept d'agent et système multi-agents

Dans ce chapitre, nous nous intéressons à des concepts essentiels pour le domaine d'agent et les systèmes multi-agents. Nous donnons quelques définitions sur les agents et les systèmes multi-agents et leur environnement. Nous donnons aussi les concepts d'interaction et de communication entre les agents, et les différentes architectures qui existent dans la littérature, et nous présentons les différentes plateformes de développement d'agent.

Chapitre 2 : Les agents BDI et le Contexte

Nous nous intéressons dans ce chapitre à des concepts essentiels pour notre proposition. Ce chapitre est composé principalement de deux parties. Dans la première partie, nous présentons le modèle BDI notamment les différents formalismes logiques de ce modèle (formalisme de Cohen et Levesque et formalisme de Rao et Georgeff), les architectures les plus connues comme PRS (Procedural Reasoning System) et IRMA (Intelligent Resource-bounded Machine Architecture), et les applications développées basées sur ce modèle. Dans la deuxième partie, nous donnons quelques définitions sur le contexte et les systèmes sensibles au contexte, et nous présentons les différentes catégories, rôle et modèles de représentation de contexte, et enfin nous donnons quelques systèmes tenant compte du contexte tels que Orca et Context-Based Reasoning.

Chapitre 3 : l'architecture BDI proposée

Ce chapitre constitue notre contribution. Il présente de manière approfondie les bases de notre travail, en commençant par la présentation de notre extension d'architecture BDI et de présenter l'importance de contexte dans notre architecture proposée. Ensuite, nous expliquons les différents composants de l'architecture et le mécanisme utilisé pour sélectionner la situation contextuelle pertinente à la situation actuelle de l'agent BDI, cela permet à l'agent de se comporter d'une façon plus approprie à son contexte. Ensuite, nous présentons les différents algorithmes de l'architecture.

Chapitre 4 : Etude de cas Nous terminons par la présentation d'une étude de cas. Pour l'étude de cas, nous utilisons notre architecture pour résoudre le problème du Wumpus décrit dans le livre de Stuart Russel et Peter Norvig.

Chapitre 1 : Concept d’agent et système multi-agents

2.1 Introduction

L'évolution technologique oblige les différents logiciels de devenir de plus en plus décentralisés, organisés et cherche à donner plus d'autonomie et d'initiative aux différents modules de logiciels. Aujourd'hui le concept d'agent et les systèmes multi-agents répond à ces besoins et propose certaines propriétés (autonomie, flexibilité, sociabilité …etc). Ce chapitre montre l'importance de ces concepts.

2.2 Système multi-agents (SMA)

Un SMA peut être défini comme un ensemble d'agents situés dans un environnement commun et interagissent selon une certaine organisation.

Jacques Ferber définit un système multi-agents comme suit [HYPERLINK \l "Jac95" **1**]:

"On appelle système multi-agents (ou SMA) un système composé des éléments suivants :

1) Un environnement **E**, c'est à dire un espace disposant généralement d'une métrique.
2) Un ensemble d'objets **O**. Ces objets sont situés, c'est à dire que, pour tout objet, il est possible à un moment donné d'associer une position dans **E**. Ces objets sont passifs, c'est-à-dire, ils peuvent être perçus, créés, détruits et modifiés par les agents.
3) Un ensemble **A** d'agents, qui sont des objets particuliers (**A** ⊆ **O**), lesquels représentent les entités actives du système.
4) Un ensemble de relations **R** qui unissent des objets (et donc des agents) entre eux.
5) Un ensemble d'opérations **Op** permettant aux agents de **A** de percevoir, produire, consommer, transformer et manipuler des objets de **O**.
6) Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera "les lois de l'univers".

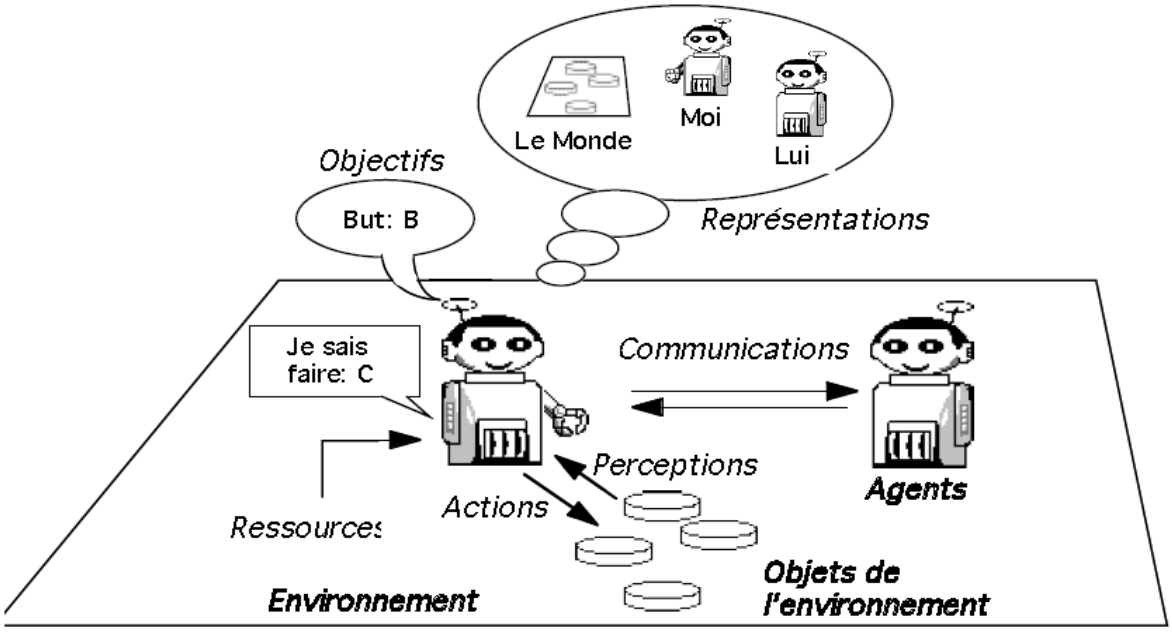


Figure 1: Représentation imagée d'un agent en interaction avec son environnement et les autres agents. [1**]**

D'après cette définition on peut dire qu'un système multi-agents est un ensemble d'agents situés dans un environnement où chacun peut agir sur le monde qui l'environne, c'est à dire sur les autres agents présents dans son univers et sur l'environnement lui-même, et effectuer un ensemble d'actions pour atteindre son objectif.

D'après Yves demazeau un SMA peut être décomposé en quatre axes : un axe Agent, un axe Environnement, des Interactions et une structure Organisationnelle explicite ou non.

SMA = Agent + Environnement + Interaction + Organisation [**2**]

2.3 Agent :

Il n'existe pas de définition unique de ce qui est un agent. Ce terme est utilisé d'une manière assez vague. Dans cette partie on peut constater que certaines définitions sont couramment utilisées dans la littérature :

Nous proposons comme première définition la suivante : « un agent est une entité logicielle ou physique à qui est attribuée une certaine mission qu'elle est capable d'accomplir de manière autonome et en coopération avec d'autres agents » [**3**]

Ferber [**1**] a proposé la définition suivante pour un agent :

« On appelle agent une entité physique ou virtuelle :

A. Qui est capable d'agir dans un environnement (Figure 2).
B. Qui peut communiquer directement avec d'autres agents.
C. Qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voir de survie, qu'elle cherche à optimiser).
D. Qui possède des ressources propres.
E. Qui est capable de percevoir (mais de manière limitée) son environnement.
F. Qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune).
G. Qui possède des compétences et offre des services.
H. Qui peut éventuellement se reproduire.
I. Dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

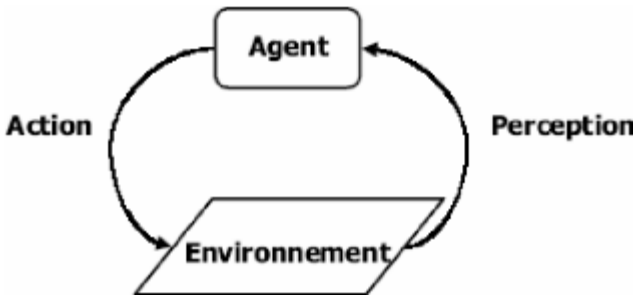


Figure 2: Interaction de l'Agent avec son environnement. [1**]**

Cette définition présente un ensemble de propriétés d'agent, comme la capacité d'agir avec leur environnement et la communication avec d'autres agents, et non pas seulement de raisonner comme dans les systèmes d'IA classique. Une autre propriété essentielle des agents est l'autonomie où l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne. Aussi l'agent possède la capacité d'observer son environnement et d'effectuer des actions autonomes pour atteindre certains objectifs.

Une autre définition d'un agent est proposée par Sycara, Jennings et Wooldridge [**4**]:

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

Les mots « situé », « autonomie », et «flexible » sont définis comme suite :

- **Situé** : L'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement ;
- **Autonomie** : L'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne ;
- **Flexible** : L'agent dans ce cas est :
    - **Capable de répondre à temps :** l'agent doit être capable de percevoir son environnement et d'élaborer une réponse dans les temps requis.
    - **Proactif :** l'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au bon moment.
    - **Social :** l'agent doit être capable d'interagir avec les autres agents (logiciels et humains) quand la situation l'exige afin d'accomplir ses tâches ou aider ces agents à accomplir les leurs.

Toutes ces définitions se fondent sur des notions semblables qui caractérisent l'agent, incluant l'autonomie et la capacité d'agir et de percevoir, et permettent de distinguer plusieurs types d'agents.

2.4 Architecture d'agents :

La communauté multi-agents distingue deux axes de classification, l'un constitué de la coordination entre les agents, et l'autre selon la description de son organisation interne : les données et les connaissances de l'agent, et des processus internes à un agent lui permettant de prendre une décision consistant à choisir une action (vision raisonnement). Pour la vision coordination nous obtenons trois types d'architecture d'agents : architecture d'agent autonome, architecture d'agent interagissant et architecture d'agent social. Et pour la vision raisonnement nous distinguons deux grandes catégories d'architectures d'agents : les architectures cognitives, les architectures réactives. Cependant, il est possible de combiner les deux architectures d'agents pour obtenir des architectures hybrides. [**5**]

2.4.1 Axe de coordination :

2.4.1.1 Architecture d'agent autonome :

Nous appelons architecture d'agent autonome, l'architecture d'un agent possédant des capacités d'action et de perception sur son environnement. Cette architecture est sans capacité explicite de coopération avec d'autres agents.

2.4.1.2 Architecture d'agent interagissant :

Nous appelons architecture d'agent interagissant, l'architecture d'un agent possédant des capacités d'action, de perception sur son environnement, et d'interaction avec les autres agents du système. L'agent interagissant sera capable d'influencer sur d'autres agents via des envois des messages.

2.4.1.3 Architecture d'agent social :

Nous appelons architecture d'agent social, l'architecture d'un agent possédant les capacités d'architecture d'agent interagissant, en plus de ça il possède aussi la capacité de gestion d'organisation; c'est à dire la capacité de gestion des relations avec les autres agents du système.

2.4.2 Axe de raisonnement :

Un agent peut être considéré comme un système de raisonnement visant à déterminer quelles sont les actions possibles en fonction de la situation courante de l'environnement ou des autres agents.

2.4.2.1 Architecture d'agent réactif :

Les agents à capacité réactives basées principalement sur un comportement réactif de l'agent à son environnement, ils ont un mécanisme de décision très simple c'est de type Perception / Action. Ce principe permet aux agents d'agir grâce à des réflexes totalement conditionnés, et cela rend l'agent rapide à répondre, et très sensible à son environnement. Ce type d'agents ne possède pas de représentation explicite de leur environnement, des autres agents qui sont présents dans le système, et d'eux-mêmes; leurs actions dépendent, donc directement, de leurs perceptions. La coordination entre les agents réactifs repose sur des formes de communication indirecte au travers de l'environnement

Les agents réactifs basés sur l'émergence de comportements collectifs à partir de comportements individuels relativement simples. Elles défendent sur l'idée qu'il n'est pas

nécessaire que les agents soient individuellement intelligents pour que le système, dans son ensemble, soit capable de résoudre des problèmes complexes.

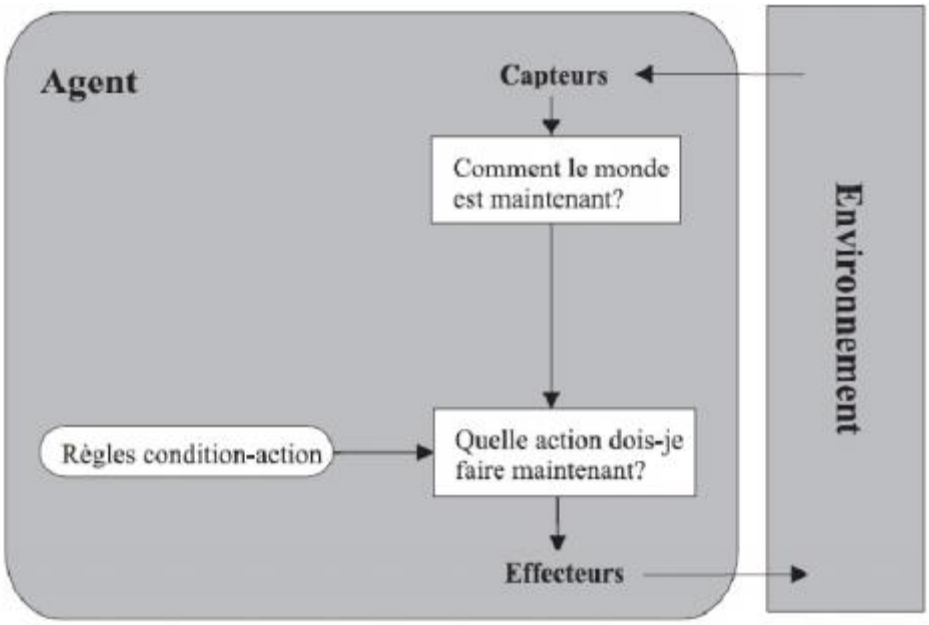


Figure 3: Schéma d'un agent à réflexes simples (Russell & Norvig, 1995) [6]

Les architectures réactives ont l'avantage de la simplicité mais, elles présentent plusieurs limitations, ce qui fait que ces architectures ne peuvent pas être utilisées dans de nombreuses applications. Les principales objections contre les architectures réactives sont les suivantes : les agents ont une vision de courte durée sur la résolution du problème, et ils ne peuvent pas toujours choisir la meilleure action à exécuter à un certain moment ; comme les agents réactifs ne possèdent pas une représentation de l'environnement, ils ne peuvent pas avoir des buts.

2.4.2.2 Architecture d'agent cognitif :

Les agents cognitifs sont des agents qui effectuent une certaine délibération pour choisir leurs actions. Ces architectures possèdent une connaissance propre qui comprend une représentation complète ou partielle de leur environnement, et des autres agents, ainsi que de leur savoir-faire, cela permet à l'agent de gérer ses interactions avec l'environnement et les autres agents.

Ce qui distingue principalement les agents cognitifs à des agents réactifs est leur mécanisme de décision. Et la capacité de représentation explicite et symbolique du monde, cela permet à l'agent de mémoriser des situations, de les analyser, de prévoir les changements induits par leurs actions et finalement de décider à adopter du comportement dans le futur.

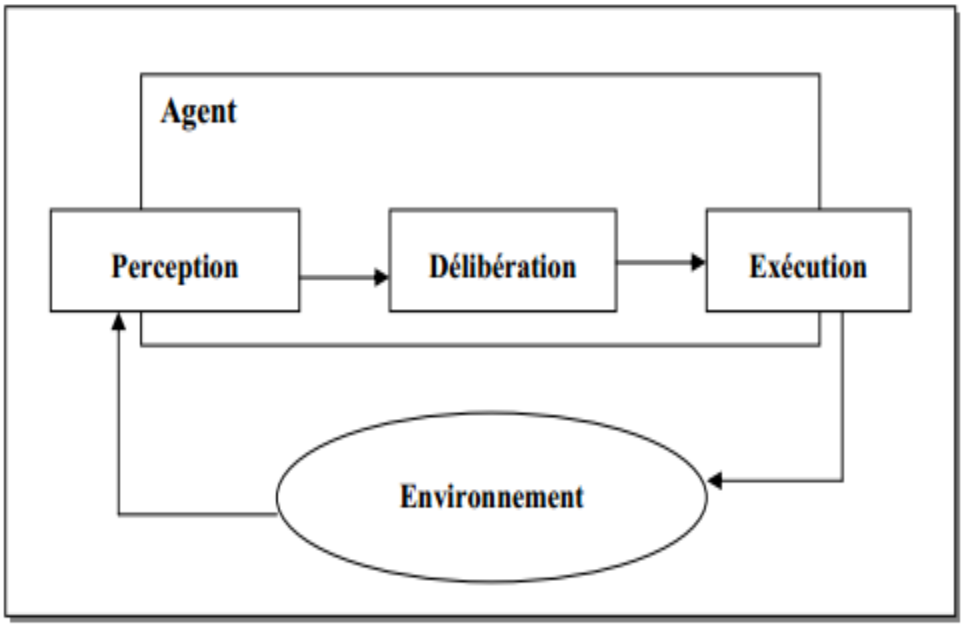


Figure 4: architecture d'un agent cognitif. [7]

Un agent cognitif peut avoir les caractéristiques suivantes:

- Intentionnalité : l'agent ne fonctionne pas en fonction des conditions de son environnement mais en fonction de ses buts propres. D'après Searl : une intention est la déclaration explicite des buts et des moyens d'y parvenir. Elle exprime la volonté d'un agent d'atteindre un but ou d'effectuer une action.
- Rationalité : Un agent rationnel est un agent qui suit le principe suivant « Si un agent sait qu'une de ses actions lui permet d'atteindre un de ses buts, il la sélectionne ». Les agents rationnels sont capables de sélectionner les meilleures actions permettant d'atteindre leurs buts selon des critères d'évaluation de leurs actions, ce qui justifie leurs décisions. De plus, la rationalité permet l'utilisation efficace des ressources par l'agent.
- Adaptabilité : Un agent est dit adaptatif lorsqu'il est capable de changer son comportement en cours de fonctionnement pour l'ajuster dans un environnement dynamique, soit pour réaliser la tâche pour laquelle il a été conçu, soit pour améliorer sa fonction ou ses performances. Un agent adaptatif est capable d'apprendre en fonction de son expérience passée et de son évolution.

2.4.2.3 Différences entre les deux architectures:

Les principaux critères de comparaison dans les systèmes d'agents se résument dans les points suivants : la représentation de l'environnement, la possession de mémoire, la complexité d'agent, et le nombre d'agent dans le système multi-agents. Le tableau ci-dessus résume les différences entre les agents cognitifs et réactifs :

Système Agent cognitif	Système Agent réactif
Représentation explicite de l'environnement	Pas de représentation explicite
Peut tenir compte de son passé	Pas de mémoire de son historique
Agent complexe	Fonction stimulus /action
Petit nombre d'agents	Grand nombre d'agents

Tableau 1:Différence entre les deux architectures [5**]**

Pour avoir le meilleur des deux solutions, à savoir architecture cognitive et architecture réactive, les chercheurs ont conçu des architectures hybrides qui combinent les caractéristiques de ces deux architectures.

2.4.2.4 Architecture hybride :

Une architecture hybride d'un agent intelligent est une architecture composée d'un ensemble de modules organisés dans une hiérarchie, chaque module étant soit une composante cognitive avec représentation symbolique des connaissances et capacités de raisonnement, soit une composante réactive. De cette manière, on combine le comportement pro-actif de l'agent, dirigé par les buts, avec un comportement réactif aux changements de l'environnement. En plus, on espère obtenir simultanément les avantages des architectures cognitives et réactives, tout en éliminant leurs limitations.

Une approche hybride structurée en couches présente plusieurs avantages : [50]

- Elle permet de modulariser un agent ; ainsi les différentes fonctionnalités sont clairement séparées et reliées par des interfaces bien définies ;
- Elle permet une conception de l'agent plus compacte, ce qui augmente sa robustesse;
- Elle accroît les capacités de l'agent car les différentes couches peuvent s'exécuter en parallèle ;

- Elle augmente la réactivité de l'agent car il peut raisonner dans un monde symbolique tout en surveillant son environnement et en réagissant en conséquence ;
- Elle réduit les connaissances nécessaires à une couche individuelle pour prendre ses décisions.

2.4.2.5 Exemple d'architecture :

Le tableau suivant rassemble les différentes architectures qui existent dans la littérature, et en classant selon les dimensions de coordination et de raisonnement présentées dans la section précédente :

Raisonnement / Coordination	Agent réactif	Agent hybride	Agent délibératif
Agent autonome	Subsumption, PENGI, MANTA	Touring Machine	*IRMA*, *PRS/dMARS*
Agent interagissant	PACO, SMARRPS	*InteRRap*	IMAGINE, *AOP*, ARCHON, *COSY*
Agent social	ECO, PACORG		*ADEPT*, *ASIC*, DIMA, CIA

Tableau 2: Architecture d'agents selon la vision coordination et raisonnement. en italique sont indiquées les architectures de type BDI [5**]**

2.4.2.5.1 Architecture Subsumption :

Brooks propose l'architecture de subsumption pour des agents réactifs. Cette Architecture repose sur l'idée qu'un système pouvait être intelligent sans manipuler d'information symbolique. Elle a été utilisée dans le domaine de la robotique.

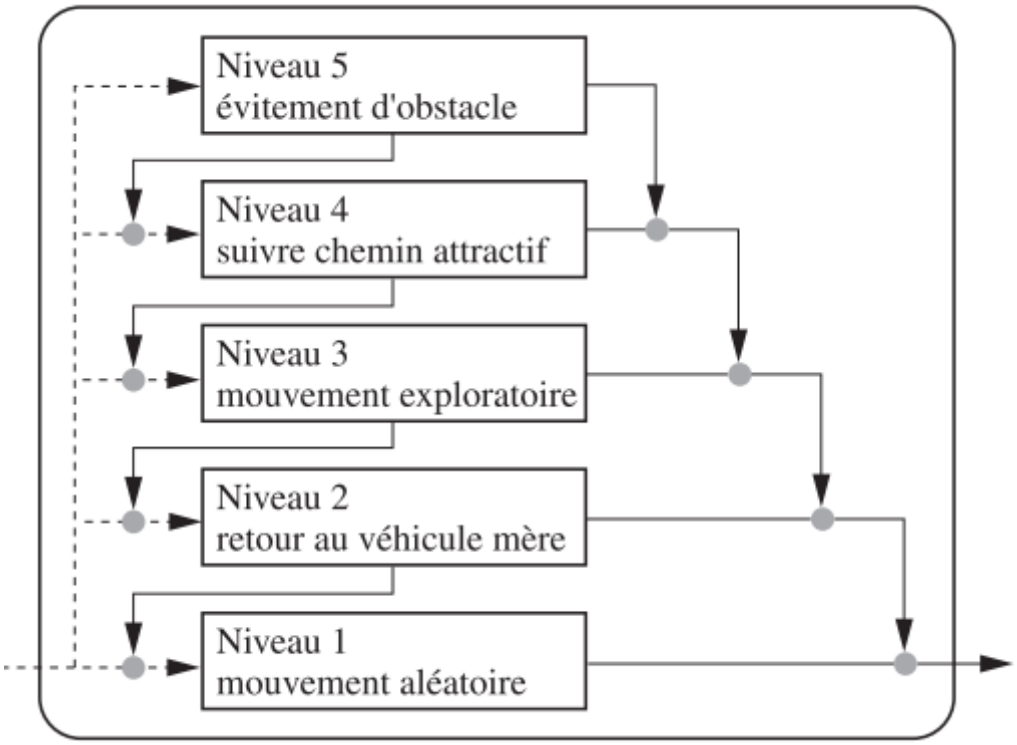


Figure 5: Architecture Subsumption. [5**]**

Dans l'architecture de subsomption, les comportements d'un agent sont regroupés en une hiérarchie de couches organisées de bas en haut, chaque couche peut atteindre un but particulier d'agent. Les couches inférieures sont moins abstraites que les supérieurs, et cela permet aux niveaux les plus élevés d'être capables de subsumer les niveaux inférieurs, ou les comportements des couches supérieures inhibent les comportements appartenant aux couches des niveaux inférieurs.

2.4.2.5.2 Architecture InteRRap :

Cette architecture est basée sur le modèle BDI et combine entre un comportement réactif et coopératif, est constituée :[5]

- Un composant de perception, de communication et d'action représente l'interface de l'agent avec l'environnement.
- Trois couches fonctionnant en parallèle :
  - planification coopérative (PC)
  - planification locale(PL)
  - comportement (C).

- Chacune des couches possède une structure uniforme constituée de deux fonctions générales :
    - Reconnaissance de situation et activation de but (SB),
    - Planification et ordonnancement (PO).
- Une base de connaissance représente hiérarchiquement les informations utilisées par les différentes couches :
    - Croyance sur l'état de l'environnement pour la couche C
    - Croyance sur l'état de l'agent lui-même pour la couche PL
    - Croyance sur les autres agents du système pour la couche PC

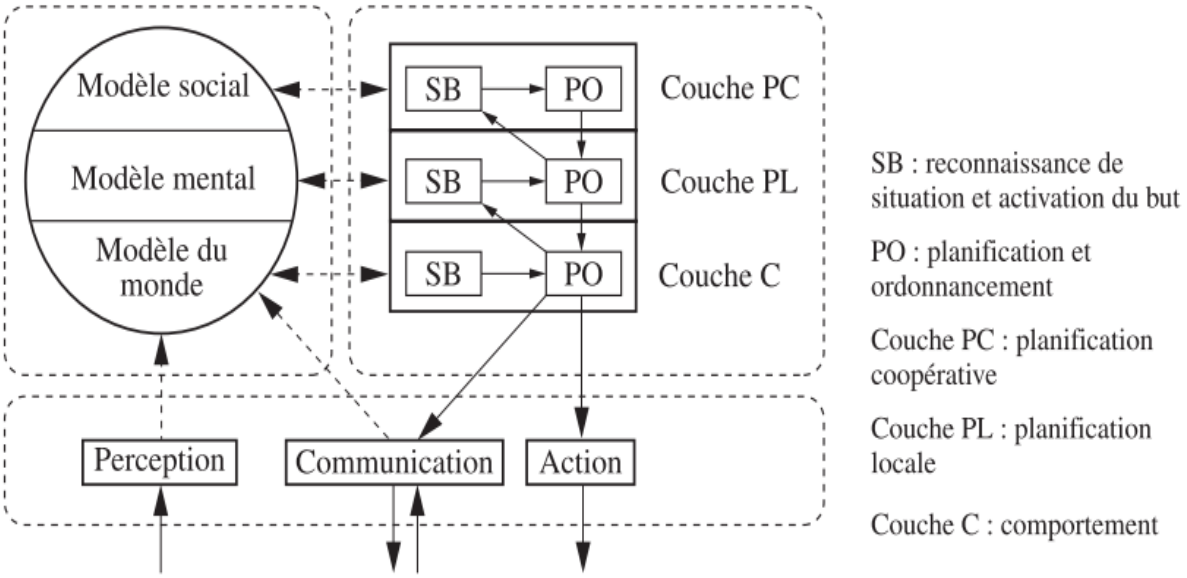


Figure 6: Architecture InteRRap [5]

2.4.2.5.3 Architecture CIA :

Dans cette architecture, quatre modules sont responsables chacun d'une des activités principales de l'agent : [5]

- **Gestionnaire de tâches :** planifie, ordonnance et suit l'exécution des tâches que l'agent doit exécuter.
- **Gestionnaire de contrats :** responsable de la négociation de nouveaux contrats et du suivi des contrats établis.
- **Gestionnaire des communications et interactions :** initie et gère les échanges de messages entre les agents.

- **Gestionnaire de l'exécution des services :** gère les actions locales et les services qu'un agent peut réaliser pour d'autres agents.

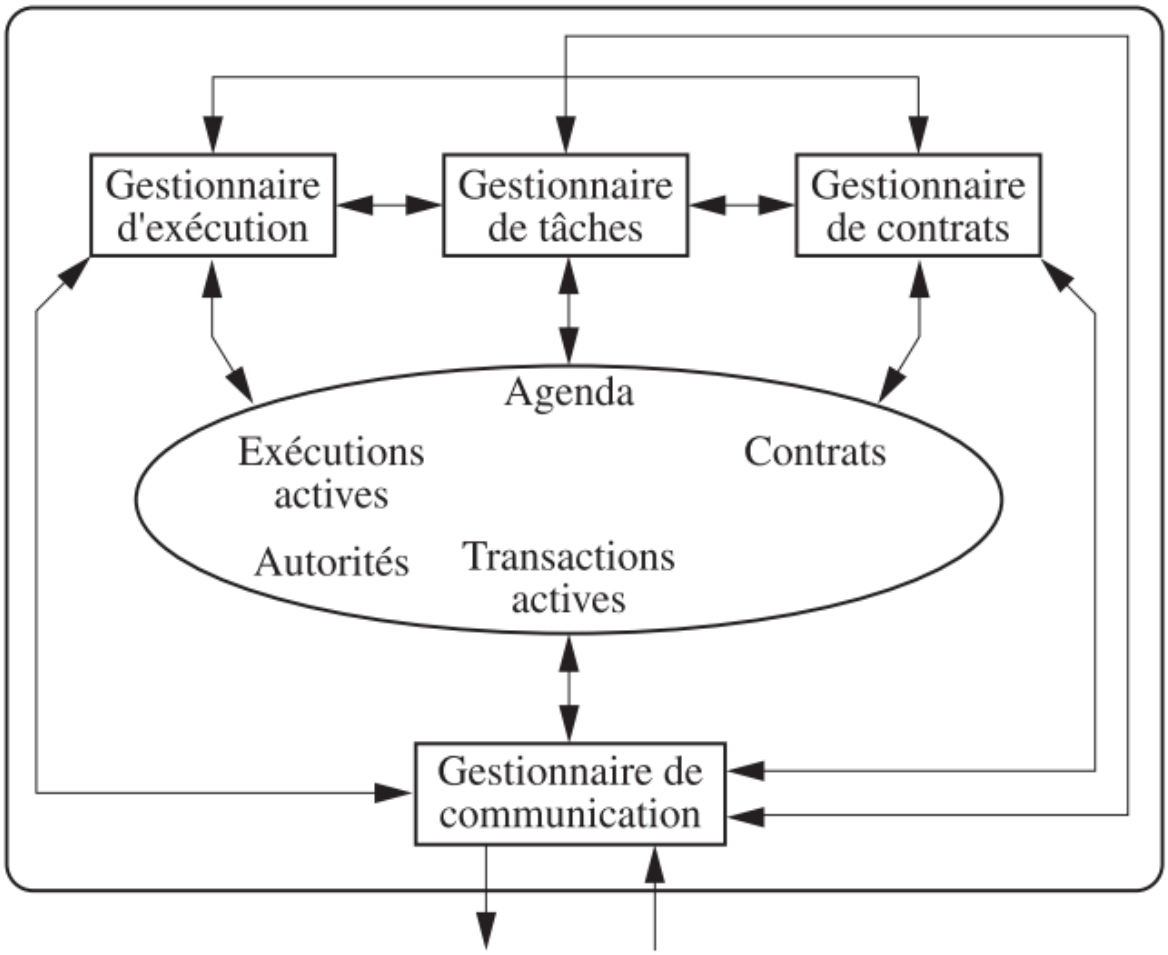


Figure 7: Architecture CIA [5]

2.5 Environnement d'un système multi-agents :

L'environnement d'un SMA est le contexte dans lequel les agents vont évoluer. Il fournit un support commun aux actions des agents, permettant ainsi l'interaction dans le système, et il constitue une source d'informations à laquelle les agents peuvent accéder au travers de leur perception. Dans le cas d'agents réels (comme des robots par exemple), l'environnement est le milieu dans lequel les agents sont plongés, ce milieu étant gouverné par les lois physiques naturelles. Dans le cas d'agents logiciels, un environnement artificiel possédant ses propres lois d'évolution doit être défini. Il est généralement considéré qu'un environnement est composé d'entités actives (les agents), d'entités passives (des objets) et de ressources qui peuvent être produites et consommées. Sans environnement il n'y a pas d'action ou de perception possible, on ne peut donc pas parler d'agent sans parler d'environnement.

2.5.1 Différents types d'environnement :

Les caractéristiques de l'environnement influencent la façon dont on conçoit un agent car il faut tenir compte de l'évolution de l'environnement, de la capacité de l'agent de percevoir cette évolution et de sa capacité à décider en conséquence

1) **Environnement accessible ou inaccessible:**

   Dans un environnement accessible, le système peut obtenir une information complète, exacte et à jour sur l'état de son environnement. Dans un environnement inaccessible, seule une information partielle est disponible.

2) **Environnement déterministe ou non déterministe :**

   Si l'état suivant de l'environnement est déterminé d'une manière unique par l'état courant et l'action de l'agent, alors l'environnement est déterministe. Si le résultat est incertain, notamment si, par suite d'une action de l'agent, l'environnement peut évoluer de différentes manières, alors on est dans le cas non déterministe

3) **Environnement statique ou dynamique :**

   Un environnement est dit statique s'il ne change d'état que sous l'effet des actions des agents. Au contraire, un environnement dynamique possède ses propres processus d'évolution qui peuvent modifier son état sans intervention des agents.

4) **Environnement discret ou continu :**

   On parle d'environnement discret lorsque le nombre de perceptions et d'actions possibles est limité. Si ce n'est pas le cas, l'environnement est alors continu.

2.6 Interactions et Communication :

2.6.1 Interactions :

Une des principales propriétés de l'agent dans un SMA est celle d'interagir avec les autres agents. Ces interactions sont généralement définies comme toute forme d'action exécutée au sein du système d'agents et qui a pour effet de modifier le comportement d'un autre agent. Elles permettent aux agents de participer à la satisfaction d'un but global. Cette participation permet au système d'évoluer vers un de ses objectifs et d'avoir un comportement intelligent indépendamment du degré de complexité des agents qui le composent.

En général, les interactions sont mises en œuvre par un transfert d'informations entre agents ou entre l'environnement et les agents, soit par perception, soit par communication. Par la perception, les agents ont connaissance d'un changement de comportement d'un tiers au travers du milieu. Par la communication, un agent fait un acte délibéré de transfert d'informations vers un ou plusieurs autres agents. L'interaction peut être décomposée en trois phases non nécessairement séquentielles : [52]

- la réception d'informations ou la perception d'un changement,
- le raisonnement sur les autres agents à partir des informations acquises,
- une émission de message(s) ou plusieurs actions (plan d'actions) modifiant l'environnement. Cette phase est le résultat d'un raisonnement de l'agent sur son propre savoir-faire et celui des autres agents.

2.6.1.1 Situations d'interaction :

Plusieurs types d'interaction ont été définis et analysés à travers divers composantes. Ferber donne une classification des situations d'interaction, cette classification présente les différentes situations d'interaction que l'on retrouve en fonction des objectifs des agents (compatibles ou incompatibles), des ressources dont ils disposent (suffisantes ou insuffisantes) et de leurs compétences pour la résolution d'un problème

Buts	Ressources	Compétences	Situation
Compatibles	Suffisantes	Suffisantes	Indépendance
Compatibles	Suffisantes	Insuffisantes	Collaboration simple
Compatibles	Insuffisantes	Suffisantes	Encombrement
Compatibles	Insuffisantes	Insuffisantes	Collaboration coordonnée
Incompatibles	Suffisantes	Suffisantes	Compétition individuelle pure
Incompatibles	Suffisantes	Insuffisantes	Compétition collective pure
Incompatibles	Insuffisantes	Suffisantes	Conflits individuels pour des ressources
Incompatibles	Insuffisantes	Insuffisantes	Conflits collectifs pour des ressources

Tableau 3: Classification des situations d'interactions. [1]

Le tableau expose l'influence des trois critères (objectifs, compétences, ressources) sur la situation d'interaction induite :

- **Indépendance :** La situation d'indépendance ne pose aucun problème du point de vue multi-agents et se résume à la simple juxtaposition des actions des agents pris indépendamment, sans qu'il y ait effectivement d'interaction.
- **Collaboration simple :** La collaboration simple consiste en une simple addition des compétences ne nécessitant pas d'actions supplémentaires de coordination entre les intervenants.
- **Collaboration coordonnée** : La collaboration complexe suppose que les agents doivent coordonner leurs actions pour pouvoir disposer de la synergie de l'ensemble de leurs compétences.
- **Compétition individuelle pure** : Quand les buts sont incompatibles, les agents doivent lutter ou négocier pour atteindre leurs buts.
- **Compétition collective pure :** Lorsque les agents n'ont pas la compétence suffisante, ils doivent se regrouper au sein de coalitions ou d'associations pour parvenir à atteindre leurs objectifs.
- **Conflit individuel pour des ressources :** Lorsque les ressources ne peuvent être partagées, on se trouve dans une situation caractéristique de conflit dont les ressources sont l'enjeu, chacun voulant les acquérir pour lui seul.
- **Conflits collectifs pour des ressources :** Ce type de situation combine la compétition collective aux conflits individuels pour des ressources. Les coalitions luttent les unes contre les autres pour obtenir le monopole d'un bien, d'un territoire ou d'une position.

2.6.1.2 Résolution de conflits :

Les agents coopératifs ont besoin de coordonner leurs activités et négocier leurs actions afin d'éviter les situations conflictuelles pour résoudre un problème.

- **La Coordination :** La coexistence des agents dans un environnement où les ressources sont limitées, et où les objectifs locaux des différents agents peuvent se contredire provoquant l'apparition de situations de conflits qui influencent sur le rendement global de tout le système, et ainsi diminuent les avantages de coopération.

Pour vaincre ces situations de conflits et d'en sortir ou d'en éviter, les agents sont amenés à exécuter des actions supplémentaires (or que les actions productives), ces dernières sont dites actions coordinatrices. Le concept de coordination regroupe l'ensemble d'outils et méthodes qui peuvent être employés pour résoudre des conflits (dus aux ressources partagées limitées ou aux objectifs incompatibles) ou pour optimiser des comportements (éliminer des actions redondantes et inutiles) et plus généralement pour assurer un tout cohérent.

- **La Négociation :** La négociation joue un rôle fondamental dans les activités de coopération. En général, une négociation intervient lorsque des agents interagissent pour prendre des décisions communes, alors qu'ils poursuivent des buts différents. Plus précisément, l'objectif de la négociation est de résoudre des conflits qui pourraient mettre en péril des comportements coopératifs. Il y a d'abord un échange de points de vue, généralement divergents, puis un ajustement réciproque (concessions ou propositions alternatives) pour obtenir le compromis attendu. Le processus de négociation permet d'améliorer les accords (en réduisant les inconsistances et l'incertitude) sur des points de vue communs ou des plans d'action, grâce à l'échange structuré d'informations pertinentes [06].

2.6.2 Communication :

Les communications, dans les SMA comme chez les humains, sont à la base des interactions et de l'organisation sociale. Sans communication, l'agent n'est qu'un individu isolé. Une communication peut être définie comme une forme d'action locale d'un agent vers d'autres agents. Les questions abordées par un modèle de communication peuvent être résumées par l'interrogation suivante : Qui communique Quoi, A qui, Quand, Pourquoi, et Comment ? [53]

1. **Pourquoi les agents communiquent-ils** ? La communication doit permettre la mise en œuvre de l'interaction et par conséquent la coopération et la coordination d'actions.
2. **Quand les agents communiquent-ils** ? Les agents sont souvent confrontés à des situations où ils ont besoin d'interagir avec d'autres agents pour atteindre leurs buts locaux ou globaux. La difficulté réside dans l'identification de ces situations. Par exemple, une communication peut être sollicitée suite à une demande explicite par un autre agent.

3. **Avec qui les agents communiquent-ils** ? Les communications peuvent être sélectives sur un nombre restreint d'agents ou diffusées à l'ensemble des agents. Le choix de l'interlocuteur dépend essentiellement des accointances de l'agent (connaissances qu'a l'agent sur les autres agents).
4. **Comment les agents communiquent-ils** ? La mise en œuvre de la communication nécessite un langage de communication compréhensible et commun à tous les agents. Il faut identifier les différents types de communication et définir les moyens permettant non seulement l'envoi et la réception de données mais aussi le transfert de connaissances avec une sémantique appropriée à chaque type de message.

2.6.2.1 Modèles de communication :

Au sein des SMA, la communication est souvent l'un des moyens utilisés pour échanger des informations entre agents (e.g. plans, résultats partiels, buts, etc.). La capacité de communiquer, et par conséquent, de coopérer des agents, s'appuie sur un élément central des systèmes multi-agents. Dans les systèmes multi-agents, nous distinguons essentiellement deux modes de communication [5] : La communication indirecte qui est une communication par signaux via l'environnement, et la communication directe, qui procède à un échange de messages entre les agents.

2.6.2.1.1 Communication indirecte via l'environnement :

Dans ce cas, la communication se fait à travers l'environnement ou bien par un tableau noir. Cette classe de communication est utilisée par les agents réactifs. Ces agents n'ont pas de communication explicite. Dans ce type de communication, les agents laissent des traces ou des signaux qui seront perçus par les autres agents. Dans ce genre de communication, il n'y a pas de destinataire bien défini.

La technique de communication par tableau noir est très utilisée dans l'intelligence artificielle pour spécifier une mémoire partagée par divers systèmes. Dans un SMA, les agents ne communiquent pas directement entre eux, mais les interactions se déroulent via l'environnement au moyen d'un espace de travail partagé. Dans ce type de communication, Les agents peuvent écrire ou lire des messages dans un espace mémoire partagée, et accessible par les différents agents du système. Le tableau noir est en général partitionné en plusieurs niveaux qui sont spécifiques à l'application. Les agents qui travaillent sur un niveau particulier peuvent accéder aux informations contenues dans le niveau correspondant du tableau noir ainsi que dans des niveaux adjacents. Ainsi, les données peuvent être synthétisées

à n'importe quel niveau et transférées aux niveaux supérieurs alors que les buts de haut niveau peuvent être filtrés et passés aux niveaux inférieurs pour diriger les agents qui œuvrent à ces niveaux.

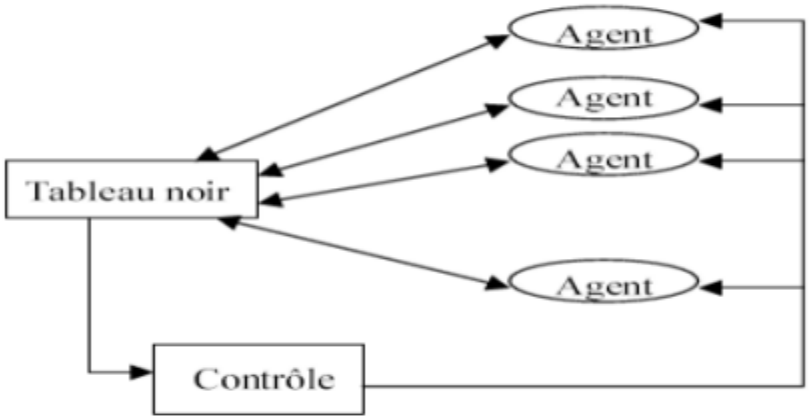


Figure 8: communication par tableau noir [7]

2.6.2.1.2 La communication directe :

Les agents fondés sur la communication par envoi des messages se caractérisent par le fait que chaque agent possède une représentation propre et locale de l'environnement qui l'entoure. Chaque agent va alors interroger les autres agents sur cet environnement ou leur envoyer des informations sur sa propre perception, Il peut envoyer à un ou plusieurs agents et d'interpréter les messages reçus de la part des autres agents. L'agent qui peut communiquer avec d'autres agents, il a les capacités suivantes : capacité d'envoi des messages, capacité de réception des messages et capacité d'interprétation des messages.

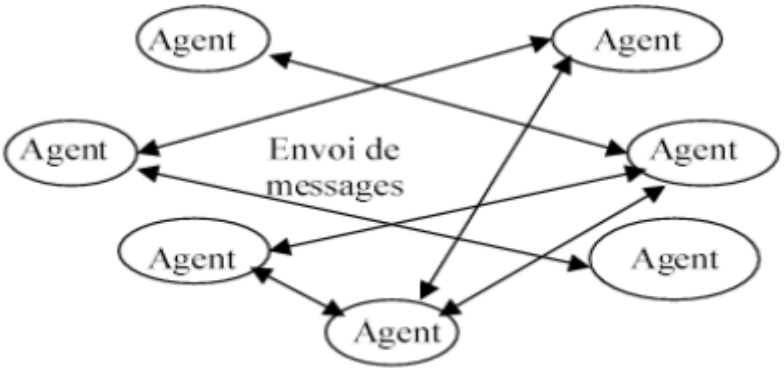


Figure 9: communication par envoi de message. [7]

Le mode de communication par envoi des messages entre agents est propre aux agents cognitifs car ils ont des connaissances sur eux-mêmes et sur autrui. Ce type de communication se base sur trois éléments essentiels : [5]

- **Le langage de communication :** Il permet de structurer les messages échangés entre les agents.
- **L'ontologie :** Elle sert à fournir un vocabulaire et une terminologie compréhensible par tous les agents.
- **Les mécanismes de communication :** Ils permettent de stocker, rechercher et adresser les messages aux agents.

2.6.2.2 Acte de langage :

La théorie des 'actes de langages' ou 'Speech Acts' est introduite par le philosophe Anglais J.-L. Austin dans son livre 'How to do things with words'. Elle constitue un fondement théorique pour la spécification de modules de communication. Elle est basée sur la constatation suivante : Lorsqu'on parle, on effectue des actes. Cela apparaît clairement dans le cas des verbes appelés 'performatifs' (comme promettre, ordonner, demander, etc.) où l'acte est effectué par l'énoncé même : 'je t'ordonne de sortir'. Mais on peut généraliser aux autres verbes appelés 'constatatifs' (par exemple : 'il pleut' devient 'je déclare qu'il pleut').

Austin distingue trois types d'actes dans un énoncé [85] :

- **La composante locutoire :** Elle concerne la syntaxe du message, le medium (écrit, oral, etc.) et les qualités de transmission.
- **La composante illocutoire :** Elle concerne le contenu sémantique qui représente aussi la force illocutoire ou performative. Avec le même contenu propositionnel, on peut former différentes illocutions.
- **La composante perlocutoire :** Il s'agit des effets que le message illocutoire a sur le destinataire ce qui se traduit par des effets sur ses croyances etc.

2.6.2.3 Les langages de communication multi-agents :

Les langages de communication d'agent, souvent désignés par l'acronyme ACL (*Agent Communication Language*). L'intérêt des langages de communication est de faciliter l'échange et l'interprétation des messages et l'interopérabilité entre les agents. Ces langages se focalisent essentiellement sur la manière de décrire des actes de communication d'un point de vue syntaxique et sémantique.

Finin et Al. déclarent que les langages de communication entre les agents doivent satisfaire certaines propriétés :[8]

- Une forme déclarative,
- Une syntaxe simple et lisible,
- Faire la distinction entre le langage de communication et le langage de contenu. Le langage de communication décrit les actes de langage et le langage du contenu permet d'exprimer les informations contenues dans le message,
- Avoir une sémantique du langage ayant un fondement théorique et une description formelle,
- Avoir une implémentation efficace, en vitesse et en bande passante

Il existe un certain nombre de langages qui permettent la communication entre les agents. Le premier langage qui a été introduit est KQML (Knowledge Query and Manipulation Language). A l'origine, KQML a été développé pour échanger des informations et des connaissances entre des systèmes à base de connaissances. Il a été ensuite utilisé pour décrire les messages échangés entre les agents. Le deuxième langage dit FIPA-ACL (FIPA Agent Communication Language), est proposé dans le cadre d'un travail de standardisation mené au sein l'organisation FIPA1 (Foundation of Intelligent Physical Agents). FIPA-ACL est une extension du langage KQML.

2.6.2.3.1 Le language KQML (knowledge query and manipulation language):

Le langage KQML est un langage de communication conçu pour l'échange des informations, des connaissances ou des services entre les agents. Il a été proposé en 1993 par le consortium DARPA-KSE (Knowledge Sharing Effort) dont l'objectif est de développer et de promouvoir des standards pour le partage et réutilisation de base de connaissance et de système à base de connaissance.

Les primitives du langage KQML sont appelées les performatifs. Les performatifs définissent les actions que les agents peuvent faire pour communiquer les uns avec les autres. Un message KQML est constitué essentiellement d'un verbe performatif associé à un contenu dont le format de représentation n'est imposé par le langage ; ce peut être un message en PROLOG ou encore une chaine de caractères.

A.La syntaxe du message KQML :

Conceptuellement, un message KQML se décompose en trois niveaux : [8]

- **Un niveau message :** qui permet d'identifier le type d'acte (nom du performatif), le langage et l'ontologie utilisés.
- **Un niveau communication :** permet d'identifier l'émetteur, le récepteur et le message ;
- **Un niveau contenu.**

La syntaxe générale du message KQML est présentée par la figure10 selon le performatif utilisé

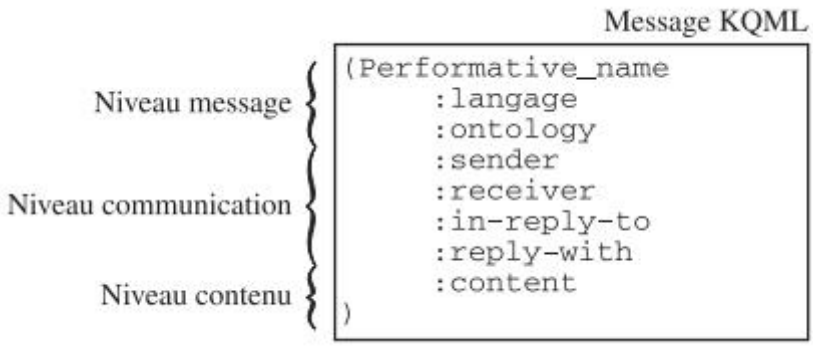


Figure 10: syntaxe générale du message KQML. [8]

Exemple : L'agent A désirant connaitre quels sont les parents de John. Pour cela, il interroge un agent B possédant cette information. La requête est la suivante:

(Stream-all

: Language Prolog

: Ontology Genealogy

: Sender A

: Receiver B

: In-reply id0

: Reply-with id1

: Content "parent(John,X)")

L'agent A veut que l'agent B l'informe de façon exhaustive (all) à propos des parents de John. L'usage de stream-all spécifie que les éléments de réponses doivent être délivrés un par un. Voici, ci-après, une des réponses de l'agent B, dans laquelle, il informe l'agent A que la mère de Johon s'appelle Alice. Cette réponse sera suivie par un deuxième type (pour indiquer le père de John) et un message de type eos (end of stream).

(Tell

: Language Prolog

: Ontology Genealogy

: Sender B

: Receiver A

: In-reply id1

: Reply-with id2

: Content "parent(John,Alice)")

B. Les performatifs :

Dans le langage de communication KQML, on distingue 3 catégories de performatifs : [8]

- **Les performatifs de discours :** sont utilisés pour échanger des informations et des connaissances entre les agents. Les performatifs *ask-if, ask-one, stream-all, deny, tell, insert,… etc* sont quelques exemples des performatifs de discours. A travers ces performatifs, un agent peut demander des informations à un autre agent.
- **Les performatifs d'interconnexion :** aide à la mise en relation des agents entre eux. Ils sont réservés aux communications entre agents. Les performatifs *register, unregister broadcast, broker, …etc* sont quelques exemples des performatifs d'interconnexion.
- **Les performatifs d'exception :** sont utiles pour changer le déroulement normal des échanges ou pour terminer les échanges. Les performatifs *error, sorry, standby,..*etc sont quelques exemples des performatifs d'exception.

2.6.2.3.2 Le langage ACL (Agent Communication Language) :

Le langage ACL a été développé par la FIPA (Foundation for Intelligent Physical Agents), un organisme qui s'occupe de la standardisation des communications entre agents. Il est fondé également sur la théorie des actes de langage. Aussi le langage ACL est plus riche au niveau de la sémantique. Il a été développé aux répondre aux critiques faites à KQML

A.La syntaxe du message ACL :

La syntaxe du message est assez similaire à celle de KQML. La figure11 suivante présente un exemple qui résume les éléments principaux du message ACL.

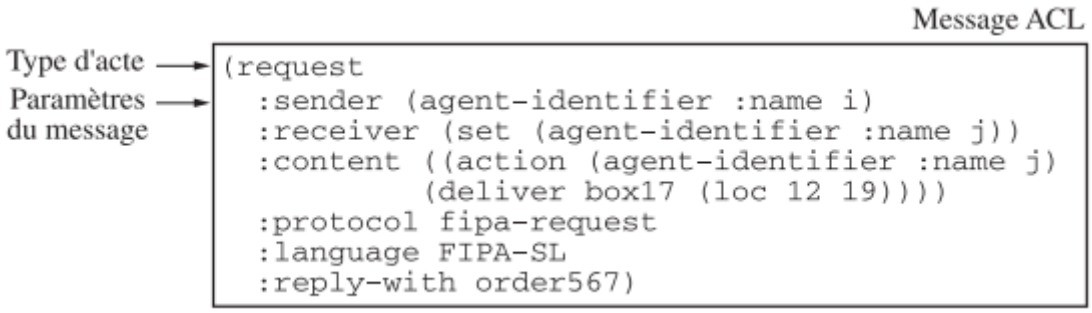


Figure 11: exemple d'un message ACL. [8]

2.7 Organisations multi-agents :

L'organisation d'un système multi-agents est la manière dont le groupe est constitué, à un instant donné, pour pouvoir fonctionner. Elle décrit l'ensemble des composants fonctionnels du système, leurs natures, leurs responsabilités et leurs besoins en ressources ainsi que les liens de communication entre agents.

Dans un système multi-agents, les agents ont des rôles. Le cadre organisationnel fixe les rôles des agents dans un groupe et donc les règles.

Selon Ferber, il existe deux architectures d'organisations pour les sociétés d'agents : [1]

- Structure horizontale : Dans de telles sociétés, tous les agents sont au même niveau, il n'y a pas d'agents maîtres et d'agents esclaves. C'est le cas, par exemple, d'un groupe d'agents ayant des spécialités différentes travaillant pour la résolution d'un même problème.
- Structure verticale : Dans une architecture verticale, les agents sont structurés par niveaux. Dans un même niveau on retrouve localement une structure horizontale. Le

fonctionnement des agents dans de telles sociétés est le suivant : l'agent reçoit le problème à résoudre d'un autre agent qui lui est supérieur dans la hiérarchie, il le décompose-en :

- des sous-problèmes auxquels il peut répondre localement,
- des sous-problèmes qu'il pourrait résoudre en coopérant avec les autres agents du même niveau que lui,
- Des sous-problèmes qu'il fait suivre aux agents du niveau inférieur dans la hiérarchie.

2.8 Méthodes de conception de système multi-agents :

Le développement de système multi-agents nécessite des méthodes et des outils qui permettent leur réalisation. Dans la littérature, il y a plusieurs méthodes de développement orientées agent.

Une méthode de développement est constituée d'un processus de développement et des notations permettant la modélisation ainsi que des outils d'aide à la conception, à la vérification et/ou au suivi du processus. L'intérêt d'une méthode pour les SMA consiste en la proposition d'une démarche qui permettra au concepteur d'être guidé pour modéliser, implémenter et vérifier un système multi-agents. A cet effet, les chercheurs dans ce domaine se sont basés sur des modèles, méthodes et outils déjà existants. Ces méthodes peuvent être classées en trois catégories : [9]

- Les extensions d'approches orientées objets : comme AAII, Gaia, Aalaadin, ADELFE
- Les approches dérivées de l'ingénierie de la connaissance : DESIRE, MAS-CommonKADS
- Le troisième type, consiste en les méthodes moins classiques qui proposent de prendre dès le départ un point de vue centré sur la notion de système multi-agents, ce qui permet d'éviter de passer sous silence les aspects sociaux ou d'émergence de comportement comme voyelles et CASSIOPEE

2.9 Les plateformes de développement :

Une plate-forme de développement des systèmes multi-agents est une infrastructure de logiciels utilisée comme environnement pour le déploiement et l'exécution d'un ensemble d'agents. D'autre part, les environnements de développement multi-agents sont nécessaires pour atteindre le grand succès dans ce paradigme. Ces environnements permettent aux

utilisateurs de concevoir et de réaliser leur système multi-agents sans perdre de temps à comprendre les différents concepts.

Il existe actuellement de nombreuses plates-formes multi-agents qu'on peut classer en plusieurs catégories. [**9**]

- **Les plateformes de simulation :** Leur intérêt est de fournir un ensemble d'outils et de bibliothèques pour faciliter le développement de simulation multi-agents. Ces outils de simulation permettent :
  - D'observer, de comprendre et de contrôler l'évolution des différents agents.
  - De simuler l'environnement dans lequel ces agents évoluent.
- **Les plateformes d'implémentation :** Implémentent des architectures d'agents. Ces architectures sont conçues et implémentées dans le but de faciliter le développement des agents. Les architectures existantes varient des simples objets actifs ou des objets actifs dotés d'un langage de communication tel que KQML à des architectures plus complexes basé sur des propositions conceptuelles telles que les états mentaux.
- **Les plateformes de conception :** Leur principal intérêt est de fournir un ensemble de composants logiciels pour la conception des systèmes multi-agents et proposent une démarche pour la conception de ces systèmes.
- **Les plateformes de conception et d'implémentation :** Offrent un ensemble d'utilitaires pour guider la définition d'un groupe d'agents ou l'introduction d'un nouvel agent. Par exemple : JACK et DECAF sont des Frameworks d'agents. JACK est basé sur l'architecture BDI et DECAF est basé sur une architecture modulaire. Ces outils fournissent également un ensemble de facilité de développement indépendant de l'architecture.
- **Les plateformes de conception, d'implémentation et de validation** : Proposent des outils pour concevoir et implémenter des applications multi-agents et une infrastructure d'exécution répartie et de test des applications multi-agents développées (validation, test, exploitation).

Nous présentons dans ce qui suit quelques plates-formes de développement SMA les plus usuelles.

2.9.1 La plate-forme MACE :

MACE est le premier environnement de conception et d'expérimentation de différentes architectures d'agents dans divers domaines d'application. L'objectif était de fournir des supports de haut niveau pour des applications dont les composantes sont de granularités variées.

2.9.1.1 Agents :

Dans MACE, un agent est un objet actif qui communique par envoi de messages. Un agent peut effectuer trois types d'actions : changer son état interne, envoyer des messages aux autres agents, envoyer des requêtes au noyau MACE pour contrôler les évènements internes. Chaque agent est doté d'un moteur qui représente la partie active de l'agent. Ce moteur détermine l'activité de l'agent et la façon dont les messages sont interprétés.

2.9.1.2 Outils de développement :

Les principales composantes de MACE sont :

- Une collection d'agents liés au domaine d'application ;
- Une bibliothèque d'outils et de fonctions que tous les agents peuvent utiliser : un simulateur, plusieurs moteurs d'agents, la gestion des erreurs et le traitement de messages ;
- Une base de données; ainsi qu'un agent qui gère les interactions avec cette base de données. Cette dernière est utilisée pour mémoriser les descriptions des différents agents, traiter ces descriptions et éventuellement recréer des agents à partir de ces descriptions.

2.9.1.3 Applications :

Le premier domaine d'application de MACE est la conception de systèmes multi-agents. MACE a été également utilisé pour développer des simulations d'applications distribuées.

2.9.2 DIMA (développement et implémentation SMA) :

DIMA est principalement fondée sur la réutilisation de *frameworks*. La première version de DIMA a été implémentée en Smalltalk-80 et a été ensuite portée en JAVA.

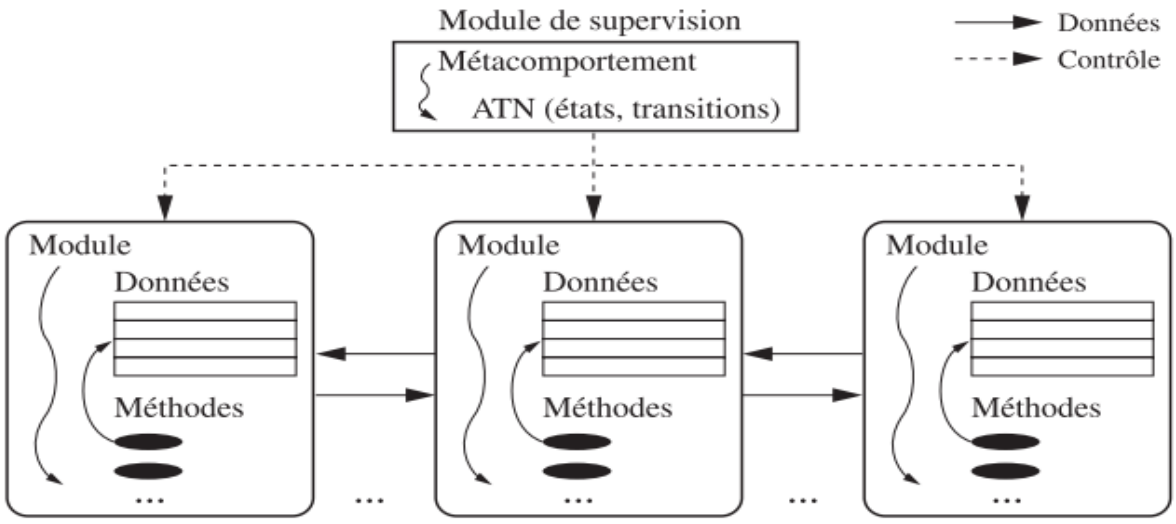


Figure 12: architecture d'agent DIMA [9]

2.9.2.1 Agents

L'architecture d'agents D IMA propose de décomposer chaque agent en différents modules dans le but d'intégrer des paradigmes existants notamment des paradigmes d'intelligence artificielle. Ces modules représentent les différents comportements d'un agent tels que la perception (interaction agent-environnement), la communication (interaction agent-agent) et la délibération. Un agent peut ainsi avoir un ou plusieurs modules qui peuvent être réactifs ou cognitifs.

Pour gérer les interactions entre ces différents comportements, DIMA propose un module de supervision représentant le méta-comportement de l'agent. Ce méta-comportement gère les interactions entre les différents modules et permet à l'agent d'observer ces comportements. Un agent est ainsi une entité proactive et autonome.

Cette architecture offre la possibilité de concevoir et de réaliser les différents types d'agents : [9]

- Des agents réactifs, décrits par des comportements réactifs,
- Des agents cognitifs, décrits par des comportements cognitifs,
- Des agents hybrides, conçus pour allier des capacités réactives à des capacités cognitives, ce qui leur permet d'adapter leur comportement en temps réel à l'évolution de leur univers.

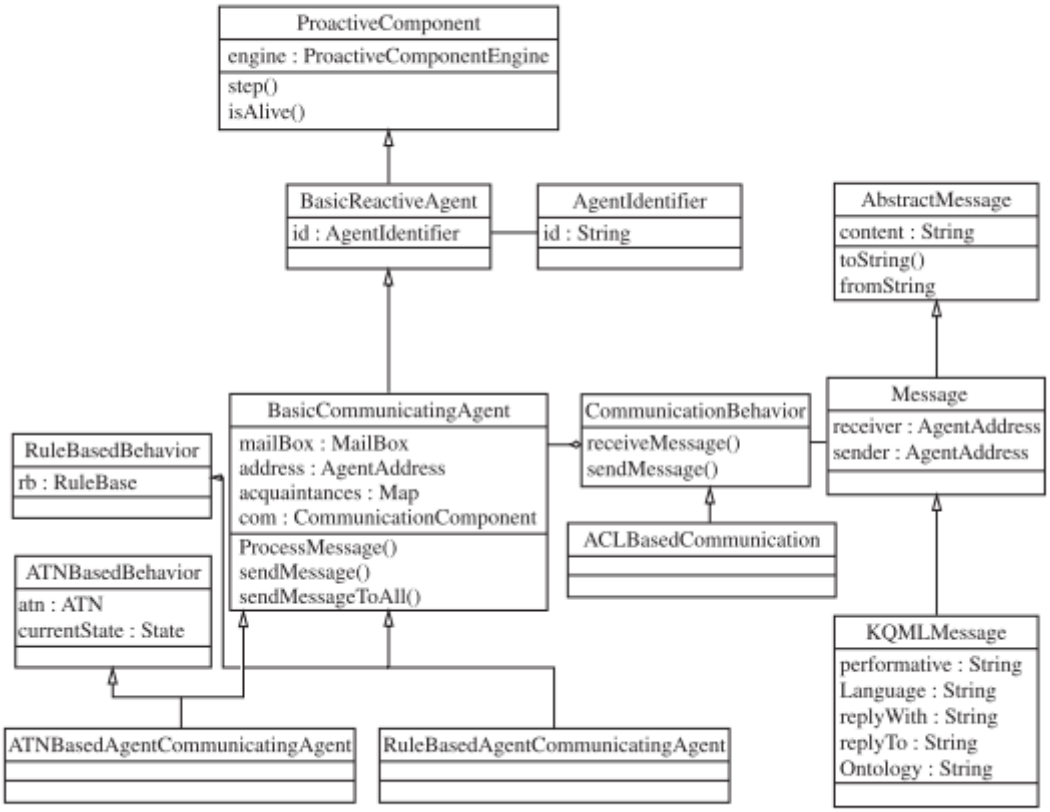


Figure 13: Exemple de bibliothèques de DIMA. [9]

2.9.2.2 Outils de développement :

La principale caractéristique de DIMA est son architecture modulaire et des bibliothèques offrant les briques de base pour construire des modèles d'agents divers. Ces différentes briques ont pour but d'offrir à l'utilisateur une implémentation des différentes propositions conceptuelles introduites par la communauté muni-agent (BDI, KQML, ACL, etc.). Les bibliothèques de DIMA regroupent des classes qui peuvent être réutilisées et/ou adaptées pour construire facilement des agents. Ces classes peuvent être instanciées pour implémenter un comportement.

L'implémentation d'un SMA consiste en l'implémentation des différents agents et éventuellement l'implémentation des objets représentant leur environnement. Les principales étapes pour implémenter un système multi-agent sont les suivantes : [9]

- implémentation des agents :
  - détermination du type d'agents, c'est-à-dire le choix de l'ensemble des modules de l'agent ainsi que les classes de la bibliothèque décrivant ces modules,
  - éventuellement, implémentation de la classe décrivant les modules choisis ou spécialisation de classes de la bibliothèque,
  - description de l'ATN décrivant le méta-comportement de l'agent ;
- description du réseau d'accointances ;
- description ou instanciation des protocoles d'interactions.

2.9.2.3 Applications :

DIMA a été utilisée pour développer plusieurs applications réelles comme la ventilation artificielle, la simulation des modèles économiques, la simulation d'un marché électrique, etc. Ces applications peuvent être des simulations, des résolutions de problèmes ou des systèmes de contrôle ayant éventuellement des contraintes temps réel.

2.9.3 MASK (Multi-Agent System Kernel):

MASK est fondé sur l'approche VOYELLES et constitue le support logiciel de cette méthode. Il fournit des bibliothèques d'agents (A), de manipulation d'environnement (E), d'interaction (I) et d'organisation (0) ainsi que les outils d'aide à la programmation.

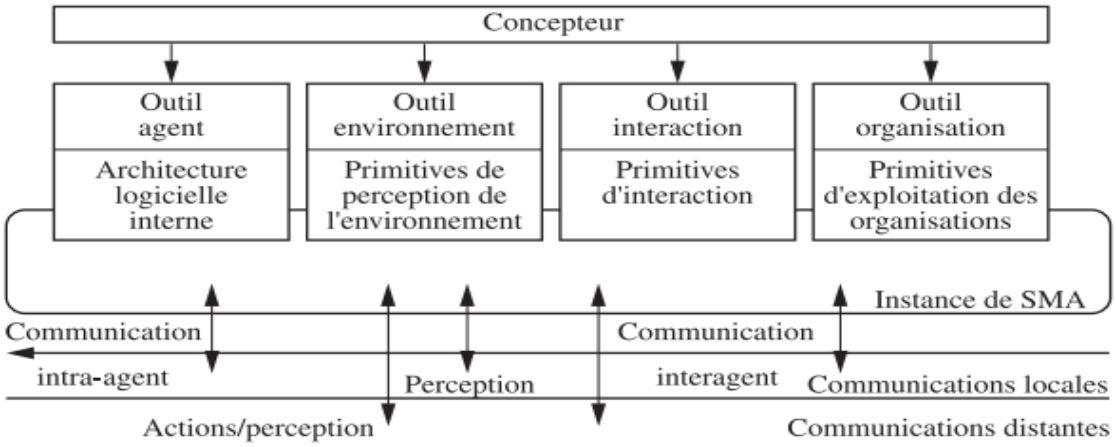


Figure 14: Vue générale de MASK. [9]

2.9.3.1 Agents :

MASK présente une approche multi-modèle. Plusieurs modèles sont disponibles, les plus marquants étant ASTRO (agent hybride) et PACORG (agent réactif). Dans le même esprit, la structure des agents est complétée par des composants de base définis sur les boîtes à outil AEIO. Ces différents composants s'intègrent aux architectures d'agents pour réaliser des structures d'agents opérationnelles.

2.9.3.2 Outils de développement :

Le principal but de MASK est de fournir au concepteur de nombreux outils intégrés au sein d'un même environnement. MASK est en effet composé de boîtes à outils couvrant les différents aspects du paradigme multi-agent.

2.9.3.3 Principes et architecture des SMA :

La boîte à outils Agent fournit à l'utilisateur des éditeurs d'agents instances de modèles d'agents prédéfinis ou permet au concepteur de modèles d'ajouter de nouveaux modèles avec leurs éditeurs associés.

La boîte à outils Environnement fournit à l'utilisateur des bibliothèques de fonctions pour manipuler les environnements (simulés) où évoluent les agents. Le concepteur peut définir de nouveaux environnements et associer les bibliothèques correspondantes. Cette boîte dépend très fortement des applications.

La boîte à outils Interaction est chargée de la présentation des fonctions d'interaction. Cette boîte se décompose en deux parties pour chaque modèle disponible : une bibliothèque de fonction et un éditeur d'interaction. Ainsi un éditeur de protocoles permettant de construire le protocole propre à notre application est fourni avec la boîte IL. On pourra aussi disposer par défaut des protocoles consacrés (Contract Net, entre autres).

La boîte à outils Organisation se décompose comme la boîte interaction en deux parties : un ensemble d'outils de modélisation de l'organisation fournissant des primitives d'exploitation et des éditeurs permettant d'instancier des organisations sur notre SMA. Le modèle RESO propose une interface graphique de construction de la représentation d'une organisation de SMA. De même, on pourra fournir par défaut les organisations classiques telles que la hiérarchie simple.

On peut construire de façon indépendante les parties déclaratives de chacune des instances d'environnement, interaction et organisation. En fonction des types d'agent choisis, un outil d'assemblage adapté est lancé pour intégrer les AEIO. Dans le cadre des modèles d'agents cognitifs, peu nombreux et complexes, l'outil d'assemblage guide la construction des parties déclaratives de l'agent (spécification des plans, etc.) et assiste la programmation des parties opératoires par l'aide à l'intégration des primitives des autres bibliothèques choisies. Dans le cas des modèles d'agents réactifs, l'outil d'assemblage propose l'intégration automatique des AEIO. La programmation des agents réactifs se fait par configuration des paramètres qui gèrent leur décision. Après instanciation des EIO, l'outil propose une instanciation automatique des agents (leur nombre interdit habituellement dans ce contexte de les créer à la main) et une configuration automatique des paramètres.

2.9.3.4 Applications :

MASK est un prototype de recherche. La plate-forme a été principalement utilisée pour simuler quelques applications dans le domaine des systèmes coopératifs (agent hybrides) ou pour des applications de résolution de problèmes en cartographie (agents réactifs). Sa structure très ouverte permet de viser de nombreux domaines d'application, sous réserve pour le concepteur de définir ses propres bibliothèques d'environnement.

2.9.4 Zeus :

Zeus est un environnement conçu et réalisé par British Telecom pour développer des applications collaboratives. Il est fondé sur les travaux de la FIPA. L'architecture des agents ZEUS est similaire à la majorité des architectures d'agents collaboratifs. Elle regroupe principalement les composants suivants :

- une boîte aux lettres et un gestionnaire de messages qui analyse les messages de la boîte aux lettres et les transmet aux composants appropriés,
- un moteur de coordination,
- un planner et un scheduler qui planifie les tâches de l'agent en fonction des décisions du moteur de coordination, des ressources disponibles et spécifications des tâches,
- plusieurs bases de données représentant les accointances de l'agent, l'ontologie utilisée, etc.,

  un contrôleur d'exécution qui gère l'horloge locale de l'agent et les tâches actives.

2.9.4.1 Outils de développement :

Zeus offre une méthodologie de développement principalement fondée sur la notion de rôle. Cette méthodologie comprend quatre étapes.

- L'analyse du domaine consiste à comprendre et modéliser le domaine. Cette étape est principalement fondée sur la notion de rôle. Le modèle de rôle définit la manière dont sont définis et liés les rôles, sans se préoccuper du comportement effectif des agents.
- La conception d'agents permet de dériver le comportement de l'agent en fonction du rôle. Cette étape permet, en effet, d'étudier la réutilisabilité des solutions existantes (ontologies, services...).
- La réalisation détermine l'implémentation du modèle conceptuel.
- Le suivi d'exécution permet à l'utilisateur de suivre l'exécution afin de tester, déboguer et optimiser.

Des éditeurs sont associés à chacune de ces étapes. Par exemple, dans la partie analyse, un éditeur de modèle de rôle est fourni Cet éditeur est issu du diagramme de classes de UML.

2.9.4.2 Applications :

Zeus a été utilisé pour développer plusieurs applications réelles (vente aux enchères, simulation de la fabrication des ordinateurs, etc.).

Les caractéristiques des domaines d'applications de Zeus ont été clairement définies par les concepteurs. Parmi ces caractéristiques :

- chaque agent crée un plan qui nécessite un raisonnement explicite pour atteindre son but ;
- la résolution de problème nécessite une coopération entre agents ;
- le rôle de chaque agent consiste à contrôler un système externe qui réalise une tâche du domaine d'application, la résolution de problème est ainsi effectuée par ce système externe et contrôlée par les agents.

Chapitre 2 : Les agents BDI et le Contexte

3.1 Première partie : les agents BDI

Dans les années 70, Mc. Carthy et Hayes ont proposé d'associer aux systèmes informatiques des états mentaux comme cela est fait pour les êtres humains [34] [26]. L'idée sous-jacente est de décrire l'´etat interne ainsi que le comportement d'un système informatique. Ceci pousse à affecter aux systèmes informatiques des attitudes mentales comme par exemple des croyances, des intentions ou des obligations. L'agent utilise ces attitudes mentales pour modéliser son comportement rationnel.

Ainsi, Newell a proposé un système informatique à base de la notion de « croyance »Newell [24]. Par la suite, de nombreux chercheurs en Intelligence Artificielle ont éprouvé le besoin d'utiliser des attitudes mentales supplémentaires (comme: Bratman [28], Cohen and Levesque [47], Rao and Georgeff [48], Sadek [49], Konolige and Pollack [30], Brazier et al [86], van der Torre and Weydert [87], Georgeff et al.[42], Padgham and Lambrix [38], Dastani et al [45],Aujourd'hui, le terme *BDI* est utilisé comme raccourci pour désigner les modèles décrivant le fonctionnement d'un agent en utilisant le concept d'attitude mentale et en particulier en utilisant des Croyance, des Désir et des Intention. [**10**]

Le terme originel de l'abréviation *BDI* est introduit par les travaux de Bratman et al [14]. Il désignait alors une architecture bien particulière permettant de modéliser des agents aux ressources limitées. Cette dernière a été désignée plus tard par Pollack sous le nom de IRMA pour « Intelligent Resource-Bounded Machine Architecture » afin de réserver le terme *BDI* pour désigner l'ensemble des modèles cognitifs d'agents Georgeff et al [42]. Ce modèle s'appuie sur l'hypothèse originellement proposée par Bratman dans Bratman [28] selon laquelle les intentions jouent, en pratique, un rôle important dans le raisonnement d'un agent humain.

Plus particulièrement, Bratman soutient qu'un agent rationnel est un agent qui met en œuvre à chaque instant la réaction qu'il juge la meilleure, il fonde son raisonnement sur ses intentions en mettant de côté les comportements qui sont en conflit avec ses intentions. Dans cette approche, l'intention peut être vue comme un moyen pour l'agent de focaliser son raisonnement. Ces intentions peuvent être indépendantes les unes des autres, mais elles appartiennent à une structure hiérarchique de plan et elles sont liées par les ressources limitées dont dispose l'agent. En effet, il faut prévoir que telle suite d'actions a1-a2 ne peut pas être exécutée car l'exécution d'acte a1 ne laisse pas assez de ressources à a2 donc il vaut mieux choisir la suite d'action a1-a3, même si elle semble moins appropriée. Bratman insiste aussi

sur la cohérence des intentions avec la connaissance que l'agent a du monde, en déclarant que : [14]

- il n'est pas rationnel pour un agent d'avoir l'intention de réaliser un acte a, tout en croyant qu'il ne réalisera pas a (inconsistance croyance-intention).
- mais il est rationnel pour un agent d'avoir l'intention de réaliser un acte a et de ne pas croire qu'il réalisera a (croyances incomplètes sur les intentions : incomplétude croyance-intention).

Rao et Georgeff dans ses auteurs épreuve qu'un agent a besoin d'un mécanisme pour sélectionner la réaction à mettre en œuvre. Ce mécanisme représente le composant délibératif de l'agent, il est basé sur les états mentaux pour guider l'agent à comporter d'une façon plus approprie à son situation (état interne, état d'environnement).

3.1.1 Le modèle *BDI* :

Un modèle BDI (Belief – Desire – Intention) est modèle qui permet de décrire et de spécifier le comportement d'un agent qui possède trois attitudes mentales : [10]

- Croyance (Belief) Les croyances d'un agent sont les informations que l'agent possède sur l'environnement et sur d'autres agents qui existent dans le même environnement. Les croyances peuvent être incorrectes, incomplètes ou incertaines et, à cause de cela, elles sont différentes des connaissances de l'agent, qui sont des informations toujours vraies. Les croyances peuvent changer au fur et à mesure que l'agent, par sa capacité de perception ou par l'interaction avec d'autres agents, recueille plus d'information
- Désir (Desire) Les désirs d'un agent représentent les états de l'environnement, et parfois de lui-même, que l'agent aimerait voir réalisés. Un agent peut avoir des désirs contradictoires ; dans ce cas, il doit choisir parmi ses désirs un sous-ensemble qui soit consistant. Ce sous-ensemble consistant de ses désirs est parfois identifié avec les buts de l'agent.
- Intention (Intention) Les intentions d'un agent sont les désirs que l'agent a décidé d'accomplir ou les actions qu'il a décidé de faire pour accomplir ses désirs. Même si tous les désirs d'un agent sont consistants, l'agent ne peut pas être capable d'accomplir tous ses désirs à la fois.

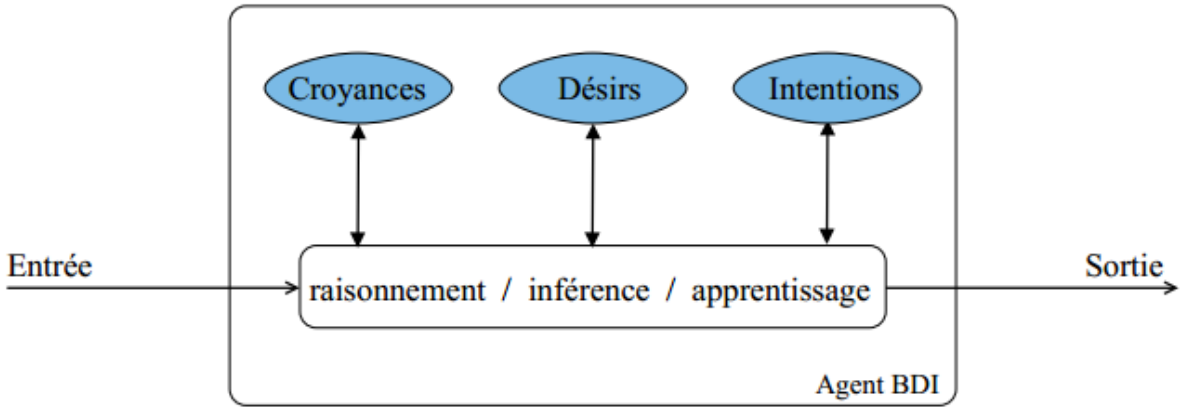


Figure 15: Vue simplifiée des attitudes mentales d'un agent BDI. [10]

3.1.2 Caractéristiques d'agent BDI :

L'architecture BDI offre les caractéristiques de base suivantes : [69]

- Adaptabilité - Les plans ne sont pas complètement spécifiés et peuvent être constitués de sous-buts imbriqués, ce qui rend l'agent flexible et lui permet de s'adapter à un environnement changeant.
- Robustesse - L'utilisation d'une hiérarchie de plans signifie que si un plan échoue, potentiellement à cause de changements dans l'environnement pendant son exécution, l'agent est capable de se ressaisir en exécutant un autre plan applicable s'il en a un disponible.
- Programmation abstraite - Le programmeur spécifie les croyances, désirs et intentions (concept de haut niveau d'abstraction), ce qui permet de créer des systèmes complexes tout en maintenant un code transparent et facile à comprendre.
- Orientation vers les buts - L'utilisation d'une approche basée sur les buts par rapport à une approche basée sur la tâche signifie qu'un agent sait pourquoi il exécute une tâche, ce qui lui permet d'expliquer son comportement d'une manière intuitive.

Les interactions entre les trois attitudes suivent le schéma suivant (voir la figure16) :

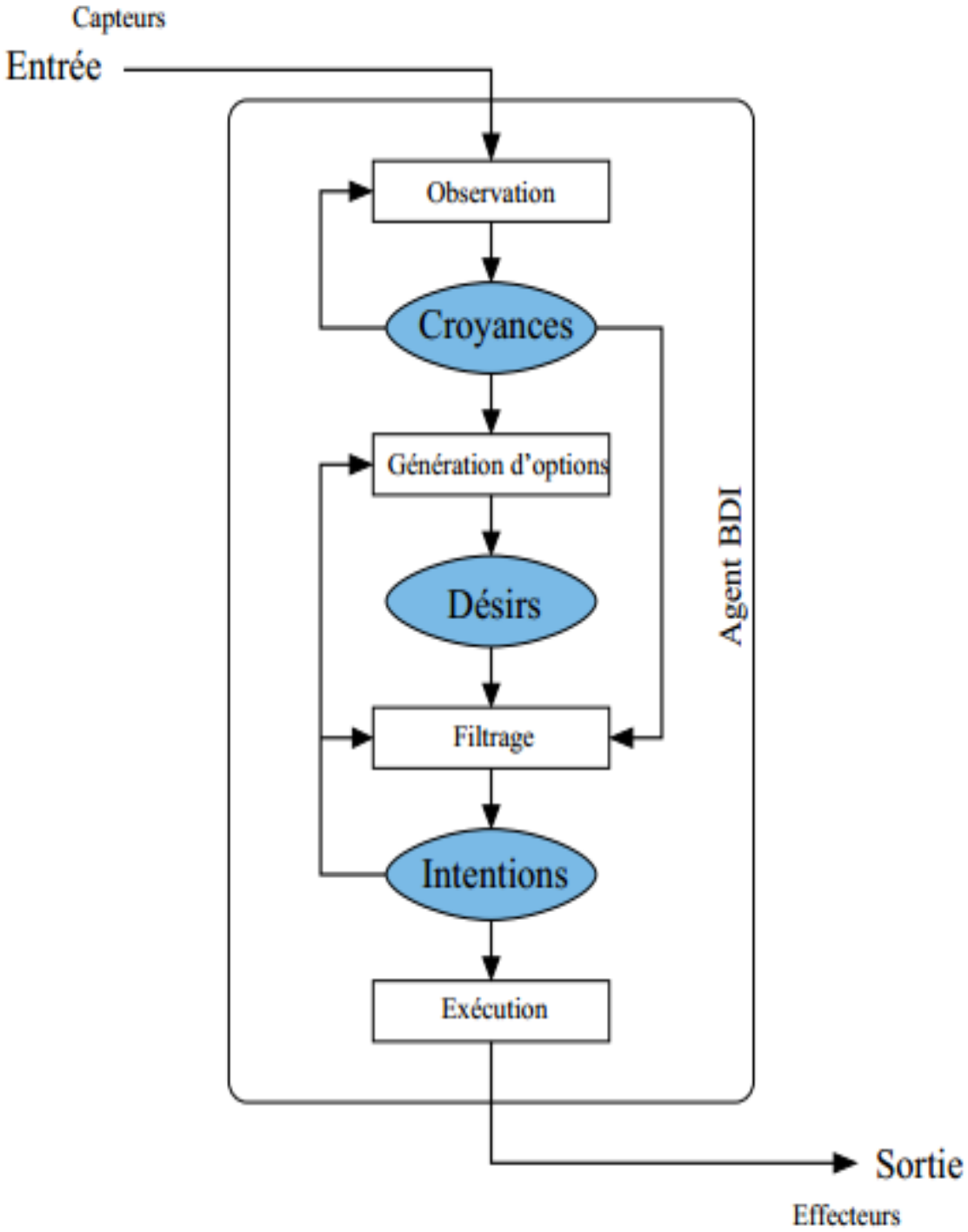


Figure 16: Schéma général de fonctionnement d'un agent BDI [10].

- un ensemble de croyances courantes (B) sur son environnement ;
- une fonction de révision de ses croyances (brf) qui calcule ses nouvelles croyances à partir des courantes et de ses nouvelles perceptions de son environnement ; brf :℘ (Bel) XP → ℘ (Bel).

- une fonction de génération de ses options pertinentes (options) qui représentent en fait ses désirs ou choix possibles conformément à ses intentions ; elle se base sur les croyances qu'il a de son environnement et sur ses intentions ; cette fonction est responsable des actions mises en œuvre ; elle doit produire des options consistantes (vis-à-vis des croyances et intentions courantes) ; enfin elle doit être opportuniste, c'est à dire être capable de détecter des changements de l'environnement qui sont favorables à la conduite de nouvelles intentions ou d'intentions laissées car irréalistes : options : $\wp$ (Bel) X $\wp$ (Inten) → $\wp$ (Des).
- une fonction de filtre (filtre) qui représente la phase initiale (quoi faire) de son processus de raisonnement ; elle active ses nouvelles intentions en fonction de ses croyances, options et intentions courantes ; c'est donc un processus continu ; cette fonction doit éliminer (rendre inactives) les intentions devenues irréalistes (trop coûteuses par rapport aux gains escomptés) ou incohérentes (entre elles) : filtre : $\wp$ (Bel) X $\wp$ (Des) X $\wp$ (Inten) → $\wp$ (Inten). Cette fonction ne crée pas de nouvelles intentions mais active/désactive des intentions existantes.
- un ensemble d'intentions courantes (I), représentant ses centres d'intérêts actuels ;
- une fonction de sélection (exécuter) de l'action à exécuter. Cette fonction renvoie une intention exécutable, qui correspond donc à une action : exécuter : $\wp$ (Inten) → A.

Un agent BDI doit donc mettre à jour ses croyances avec les informations qui lui proviennent de son environnement, décidé quelles options lui sont offertes, filtrer ces options afin de déterminer de nouvelles intentions et poser ses actions au vu de ses intentions. Le processus de décision permet de sélectionner les actions à effectuer pour atteindre ses objectifs. Dans ces modèles, le processus se décompose en deux phases : La première est appelée phase de délibération, elle consiste à fixer certains buts. La seconde correspond à une phase de planification, elle consiste à définir la manière d'atteindre ces buts.

Ce modèle fournit une bonne décomposition fonctionnelle qui identifie clairement les sous-systèmes nécessaires à la conception d'un agent.

3.1.3 Les formalisations du modèle *BDI :*

3.1.3.1 La formalisation de Cohen et Levesque du modèle *BDI :*

Cohen et Levesque ont largement utilisé les principes énoncés par Bratman, afin de formaliser une logique de l'intention plus adéquate et utilisable dans le cadre de la conception d'agents BDI. Cohen et Levesque ont identifié sept propriétés spécifiques qui correspondent aux idées directrices de la théorie de l'intention de Bratman, et qu'ils ont appliqué dans leur modèle logique [**11**]:

- Les intentions sont comme des problèmes pour les agents qui doivent déterminer un moyen de les résoudre. (Raisonnement pratique)
- un agent ne peut adopter une intention en conflit avec celles qu'il a déjà. (filtre d'admissibilité des intentions)
- les agents évaluent l'état d'avancement de leurs intentions, et sont enclins à essayer encore si leurs tentatives échouent (les intentions sont des motivateurs de l'action)
- Les agents croient que leurs intentions sont réalisables. (il est nécessaire d'avoir une croyance de la possibilité de ses intentions)
- les agents ne croient pas qu'ils ne vont pas réaliser leurs intentions
- Dans certaines circonstances, les agents pensent qu'ils exécuteront leurs intentions. (on peut à tout moment reconsidérer ses intentions)
- Les agents n'ont pas besoin de prévoir tous les effets de bord de leurs intentions. (problème de la solution globale)

Cohen et Levesque s'appuyant sur ces critères, ils développent une logique de la rationalité qui spécifie ce que l'agent choisit et comment il est engagé par ce choix. Ce concept est composé des désirs choisis de l'agent que Cohen et Levesque nomment des « buts persistants», afin de souligner l'engagement «interne» de l'agent.

Un agent A a un but persistant envers ω si et seulement si : [11]

A a comme but que ω devienne vrai et croit que ce n'est pas actuellement le cas ;

- pour abandonner ce but, A doit croire que ω est vrai ou qu'il ne le sera jamais.

Dans le modèle de Cohen et Levesque, les intentions sont définies en terme de croyances et de désirs de l'agent

Opérateur	Signification
(Bel i θ)	L'agent i croit θ
(*Goal i* θ)	L'agent I a pour but θ
(Happens α)	L'action α va arriver prochainement
(Done **α**)	L'action **α** vient d'arriver

Tableau 4: Opérateurs utilisés dans la logique de l'action de Cohen et Levesque. [11]

3.1.3.2 La formalisation de Rao et Georgeff du modèle *BDI :* [12]

A la fin des années 80, Bratman a cherché à modéliser philosophiquement un agent humain rationnel. En particulier, il a insisté sur le rôle de l'intention dans le raisonnement et proposé une architecture d'agents basée sur les concepts de croyance, de désir et d'intention. Aujourd'hui, ces travaux sont considérés comme les fondements du modèle BDI. Parmi les nombreux travaux dans ce domaine, ceux de Rao et Georgeff sont particulièrement intéressants à considérer car ils proposent une formalisation logique du modèle BDI.

Dans les travaux de Rao et Georgeff, ils ont développé une logique modale temporelle basée sur trois primitives : croyances, désirs et intentions. L'ensemble des croyances constituent le modèle du monde qu'a l'agent. Les croyances peuvent donc porter sur des informations très divers (état des objets du monde, croyances des autres agents, . . .), être exactes/inexactes et complètes /incomplètes, mais forment toujours un ensemble consistant. Les désirs « buts » représentent les motivations « objectifs de l'agent » : tout ce que l'agent souhaite mettre en œuvre ou réaliser. Enfin, les Intentions désignent les plans que l'agent s'engage à réaliser. Les intentions sont le résultat du raisonnement de l'agent et sont maintenues tant qu'elles ne sont pas réalisées ou jugées comme impossibles. Elles forment un ensemble consistant et permettent à l'agent d'avoir un comportement sur le long terme. Notons que Cette approche diffère de celle de Cohen et Levesque parce que dans cette approche, l'intention est une attitude primitive et ne se réduit donc pas à une combinaison uniquement constituée de croyances et de buts.

Rao et Georgeff formalisent leur modèle à l'aide d'une logique multimodale. Plus particulièrement, ils étendent une logique propositionnelle de style CTL (Computation Tree

Logic) avec trois opérateurs modaux : BEL, GOAL, INTEND. Ces derniers représentent respectivement les croyances, les buts et les intentions de l'agent.

Pour un raisonnement adéquat, Rao et Georgeff ont proposé un ensemble d'axiomes pour la logique BDI, qui recouvre ceux proposés par le modèle de Bratman, afin de gérer les relations qui existent entre les croyances, les désirs et les intentions :

- les intentions sont des moteurs de l'action,
- les agents croient en leurs intentions,
- ils croient également en l'accomplissement de leurs buts,
- ils ont l'intention d'accomplir leurs buts,
- ils peuvent reconsidérer leurs intentions en fonction des évolutions de l'environnement.

Ces axiomes sont la base de la formalisation de la logique BDI de Rao et Georgeff [29]

- L'opérateur BEL est KD45 (il est clos sous l'implication logique, consistant et vérifie la propriété d'introspection positive et négative).
- Les opérateurs GOAL et INTEND sont KD (ils sont clos sous l'implication logique et consistants).
- Les trois opérateurs vérifient la règle de nécessitation : l'agent croie, désire et souhaite réaliser toutes les formules valides.
- INT END(p) ⇒ BEL(INT END(p)) et GOAL(p) ⇒ BEL(GOAL(p)) sont des axiomes : l'agent s'introspecte au sujet de ses buts et de ses intentions.
- GOAL(p) ⇒ BEL(p) est un axiome : l'agent croit que chacun de ses buts est réalisable.
- Si e est une action primitive alors done(e) ⇒ BEL(done(e)) est un axiome : `a chaque fois que l'action e est réalisée, l'agent pense que l'action e vient effectivement d'être faite (observation).
- Si e est une action primitive alors INTEND(does(e)) ⇒ does(e) est un axiome : Si un agent à l'intention de mettre en œuvre une action primitive alors elle est mise en œuvre.
- INTEND(p) ⇒ inevitable◊(¬INTEND(p)) est un axiome : un agent ne doit pas reporter sine die ses intentions.

La sémantique de leur logique est basée sur les mondes possibles. Un monde possible est modélisé par une structure temporelle appelée un arbre de temps, ayant un seul passé et plusieurs branches de futurs. chaque situation de l'univers est représentée par un monde possible. Ces derniers sont les nœuds d'arbre temporel. Dont les branches ont un sens et représentent les actions permettant de transformer l'univers d'une situation en une autre. Cette structure en arbre permet de représenter les options qui s'offrent à l'agent dans chaque situation : ce sont les branches au départ de chaque nœud.

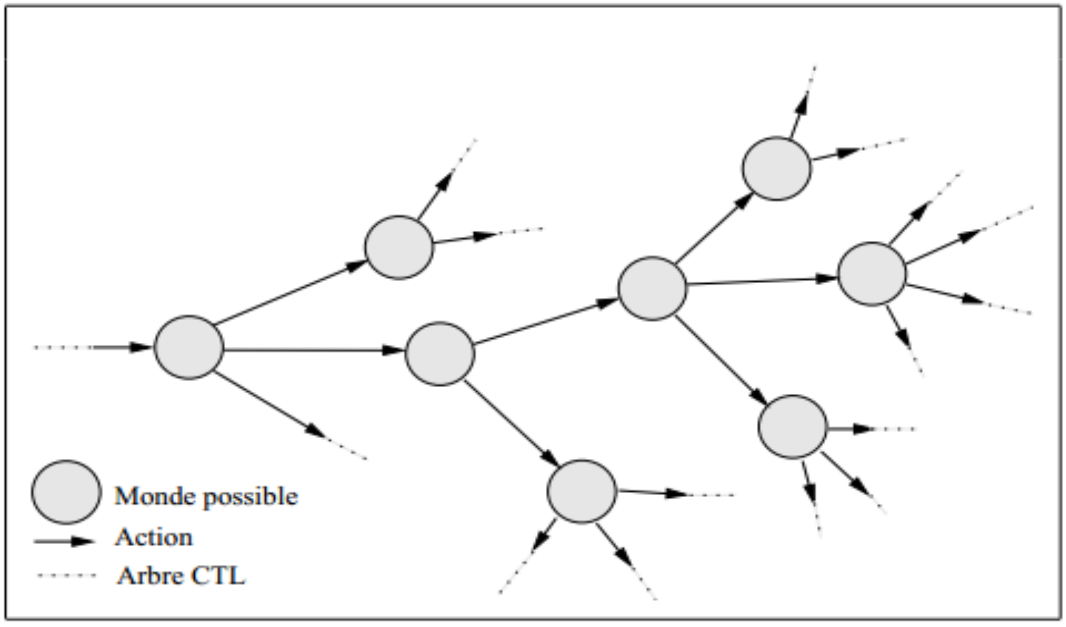


Figure 17: Schéma d'un arbre temporel CTL. [10**]**

3.1.4 Quelques architectures:

3.1.4.1 PRS (Procedural Reasoning System): [13]

Le système de raisonnement procédural PRS est la meilleure architecture disponible actuellement. Il a été déployé dans beaucoup d'applications industrielles importantes, développé à Stanford Research Center, a été ensuite utilisé dans le système du contrôle du trafic aérien OASIS à l'aéroport de Sydney en Australie, dans le système du management de business SPOC et il est devenu un produit commercial de la compagnie Agentis Solutions. Ce système est le plus connu au sein de la communauté de la recherche des agents intelligents, il est basé sur le concept de modèle de raisonnement BDI (belief, desire, intention).

L'architecture PRS est décrite initialement en 1987 par Georgeff et Lansky, réalisée en LISP. PRS a été réimplanté en C++ sous le nom de dMARS (distributed Multi-Agent Reasoning System)

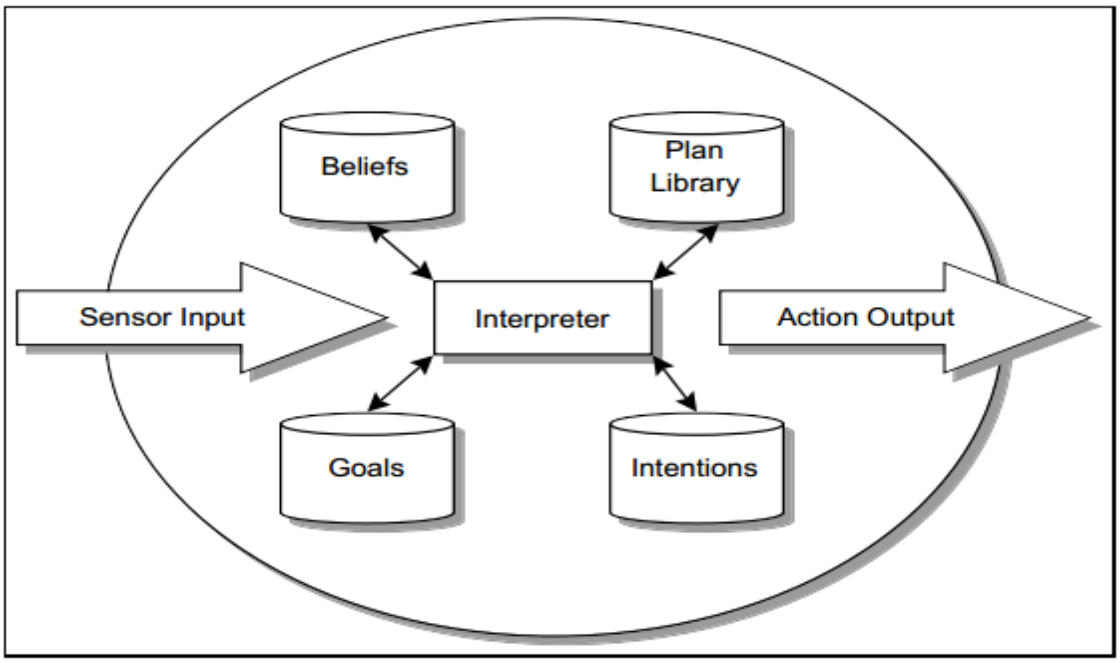


Figure 18: Architecture BDI de PRS. [13]

Elle comporte :

A. Croyances :

Une base de données contenant les croyances et les faits courants sur l'environnement. Les croyances d'un agent sont les informations que l'agent possède sur l'environnement et sur d'autres agents qui existent dans le même environnement. Les croyances peuvent être incorrectes, incomplètes ou incertaines. Les croyances peuvent changer au fur et à mesure que l'agent, par sa capacité de perception ou par l'interaction avec d'autres agents.

B. Désire (Objectifs) :

Les désire d'un agent représentent les états de l'environnement, et parfois de lui-même, que l'agent aimerait voir réalisés. Un agent peut avoir des désirs contradictoires. Dans ce cas, il doit choisir parmi ses désirs un sous-ensemble qui soit consistant. Ce sous-ensemble consistant de ses désirs est identifié avec les buts de l'agent. Ils sont représentés comme des plans non encore instanciés pour permettre à l'agent d'atteindre ses buts.

C. Intention :

Les intentions d'un agent sont les désirs que l'agent a décidé d'accomplir ou les actions qu'il a décidé de faire pour accomplir ses désirs. Même si tous les désirs d'un agent sont consistants, l'agent peut ne pas être capable d'accomplir tous ses désirs à la fois.

D. Bibliothèque de plans :

Bibliothèque de plans est un ensemble de plan, Cette composante représente le savoir-faire de l'agent sous la forme de plans décrivant les séquences d'actions à effectuer pour réaliser un but.

Chaque plan contient plusieurs composants:

- **Le déclencheur :** est le contexte ou les conditions préalables spécifiées dans lesquelles circonstances le plan prise en considération. Habituellement le plan est déclenché par des événements.
- **Condition de maintenance :** défini les circonstances qui doivent rester vraies quand le plan est en cours d'exécution.
- **Corps de plan :** défini une séquence d'action et des but (ou des subGloal (quelque chose qui doit être réalisée quand l'exécution de plan atteindre ce point dans le plan))

Exemple : le plan «préparée le thé » peut être déclenché par l'événement « assoiffé » et possède un contexte (ou pré-condition) qui défini la circonstance où le plan exécute par exemple « avoir des sacs de thé ».le corps de plan est obtenir l'eau bouillante, ajouter le sac de thé à la tasse, ajouter l'eau à la tasse. Ici, obtenir l'eau bouillante est un subGoal.

L'agent dMARS influe sur leur environnement, et leur état interne. Pour chaque événement reçu est placé dans la file d'attente d'événement.

E. L'interpréter :

Est le responsable sur la gestion des actions de l'agent. Il exécute continuellement le cycle suivant :

- actualiser la file d'événements et les croyances de l'agent.
- Il active alors de nouveaux désirs en sélectionnant les plans de la librairie de plans qui coïncident avec les événements.
- Un de ces plans est sélectionné pour exécution, et

- une fois instancié, il est placé dans la pile des intentions.
- la première action de la pile d'intentions est exécutée, et exécuté l'étape suivant de plan actuel si cette étape est une action alors exécuté autrement si est un subGoal alors placé ce subGoal dans la file d'événement. De cette façon, quand un plan commence à exécuter ses subgoals seront signalés sur la file d'événement qui à leur tour exécutera les plans qui réalisent ses subgoals pour devenir actifs, et ainsi de suite.

3.1.4.2 IRMA : Intelligent Resource-bounded Machine Architecture [14]

IRMA est une architecture qui a été développée par Bratman et collègues, en 1988, dans « Intelligent Resource-Bounded Machine Architecture ».

Cette architecture a quatre structures de données clé :

- une bibliothèque de plans,
- une représentation explicite des croyances,
- une représentation explicite des désirs,
- une représentation explicite des intentions.

Elle comporte également :

- un module de raisonnement, pour raisonner sur l'environnement,
- un analyseur moyens-fin, pour déterminer quels plans pourraient réaliser les intentions de l'agent,
- un analyseur d'opportunité, qui analyse l'environnement et en déduit des options possibles pour l'agent,
- un processus de filtrage, qui détermine quels plans sont consistants avec les intentions de l'agent,
- un processus de délibération, qui fait le choix entre les différentes options.

3.1.5 Extensions de l'architecture BDI

L'architecture BDI est ouverte aux extensions et de nombreux travaux se sont donc intéressés à l'extension des concepts de base pour supporter d'autres facteurs qui influencent le processus de raisonnement humain. Dignum et al [54] introduisent le concept de normes dans les systèmes BDI. De la même manière que le comportement humain est très influencé par les normes sociales, le comportement des agents peut lui aussi être influencé par les normes,

permettant aux agents de mieux comprendre ce que les autres agents sont susceptibles de faire. Pereira et al [55] présentent une extension de l'architecture BDI pour inclure des émotions artificielles, qui affectent le raisonnement de l'agent de la même manière que les émotions réelles affectent le raisonnement d'un être humain. Des travaux ont aussi intégré des planificateurs temps réel dans l'architecture pour supporter la création de plans lors de l'exécution de Silva et al [56]. Les émotions, les normes et la planification, [**15**] [**16**] [**17**]et la manière dont elles sont intégrées dans les simulations, seront discutées plus loin.

3.1.6 Application :

3.1.6.1 Le contrôle aérien : [18]

OASIS (Optimal Aircraft Sequencing using Intelligent Scheduling) est un système d'intelligence artificielle temps réel développé pour supporter le directeur de flux (contrôleur aérien). Il a été conçu en subdivisant la tâche de gestion de trafic aérien en ses parties majeurs et en concevant des agents séparés pour résoudre chacun de ces sous problèmes. Chaque agent résout sa partie de la tâche indépendamment et en coopérant avec les autres pour pouvoir produire le comportement général du système.

- Les agents communiquent entre eux et avec le milieu extérieur garce à des messages émis et reçus d'une manière asynchrone et supposé être sans perte. Ce pendant aucune garantie est donné concernant le traitement d'un message une fois qu'il a atteint le récipient.
- Les agents sont autonomes. Les faits, buts et intensions, qui font partie de la statue interne de l'agent, ne peuvent pas être manipulé de l'extérieur. La seule manière pour un agent de trouver la croyance, le but ou l'intention d'un autre agent est de lui envoyer un message demandant l'information.
- Comme l'environnement change, les agents doivent décider comment agir. Si la délibération prend du temps, l'agent peut trouver que les faits, buts et la situation du monde sur lesquelles se base la délibération ne sont plus valables. Donc chaque agent doit être capable de faire la correspondance entre le taux de changement du monde et les tâches qu'il doive faire pour utiliser efficacement ses ressources limitées.

Cette conception permet de faire correspondre chaque agent individuel au sous problème qu'il résout, elle permet une robustesse élevé et une réactivité variable et dynamique vis-à-vis les événements extérieurs. OASIS est conçu en utilisant deux clase d'agents. La première est ceux

qui assurent la coordination et le raisonnement inter avion appelé global agent et la deuxième ceux qui assure le calcul ou le raisonnement relatif à chaque avion individuellement appelé aircraft. Chaque avion a un agent associe à elle, incluant même les avions attendus prédis par le plan du vol et les messages de temps de dépars mais qui ne sont pas encore détecter par le radar.

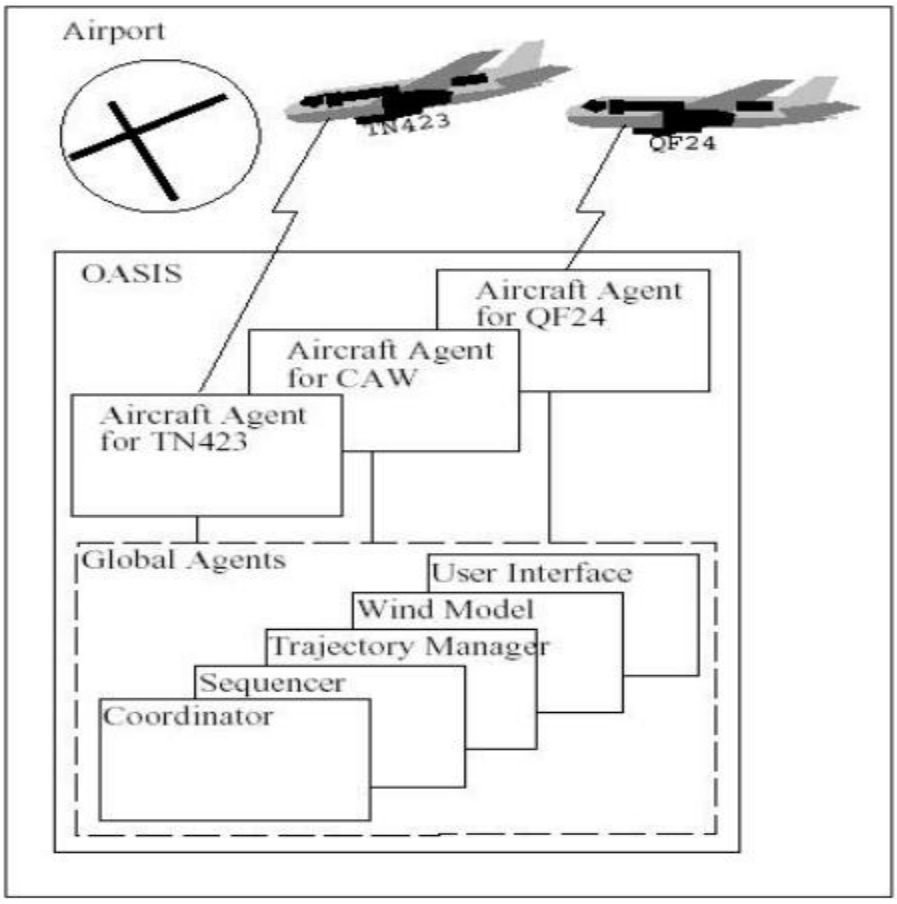


Figure 19: Architecture d'OASIS. [18**]**

Dans OASIS, il y a 5 agents globaux :

COORDINATOR : sert comme un gestionnaire de tâche, il coordonne les activités des autres agents globaux et les agents avion.

SEQUENCER : utilise les techniques de recherche pour arranger l'avion dans la plus petite séquence délais/coût.

TRAJECTORY CHECKER : vérifie que les instructions proposées par le système ne causent pas une violation des règles de sécurité de séparation.

WIND MODEL : utilise les observations individuelles prisent par les agents avion pour prévoir le champ de vent que l'avion peut rencontrer.

USER INTERFACE : sert comme le seul point de communication avec le directeur de flux et gère tous les interactions utilisateurs.

Le système associe un agent pour chaque avion approchant et essayant d'atterrir dans l'aéroport. L'agent AIRCRAFT contient toutes les données de l'avion demandé par le système. Cet agent intègre aussi la position, la vitesse et le rapport d'altitude dégagées des donné temps réel du radar. La tâche accomplie par l'agent avion comprend l'estimation du temps d'atterrissage, surveiller si l'avancement de l'avion est comme prévue et organiser la trajectoire de l'avion.

3.1.6.2 Quelques applications :

- SWARMM (Tidhar et co.), système de simulation de missions aériennes pour DSTO Australia
- IRNTMS (Rao et co.), système de gestion de réseaux pour Telstra (Telecom Australia)
- RCS (Georgeff et co.), système de diagnostic de panne pour la NASA
- BattleModel (Tidhar e co.), système de simulation de combats aériens pour Air Operations Division
- RoboCup
- Robots Footballeurs

3.2 Deuxième partie : Contexte

3.2.1 Définition du contexte : [19]

Le mot contexte est un ancien mot dans le domaine de l'informatique, en particulier dans les Systèmes Multi-Agents. Avant de présenter les définitions du contexte proposées dans le domaine de l'informatique, nous allons donner les définitions fournies par quelques dictionnaires pour le mettre dans un cadre général. Parmi les dictionnaires de référence, citons:

- Le Petit Robert : "ensemble des circonstances dans lesquelles s'insère un fait".

- L'encyclopédie Larousse : "ensemble des conditions naturelles, sociales, culturelles dans Les quelles se situe un énoncé, un discours"; ou encore : "ensemble des circonstances dans les quelles se produit un événement, se situe une action".
- Hachette Multimédia : "ensemble des éléments qui entourent un fait et permettent de le comprendre".

Ces définitions partagent l'idée d'un contexte est un ensemble d'éléments ou de circonstances considéré comme des informations qui entourent un fait. Nous abordons maintenant la notion de contexte dans le domaine de l'informatique

Schilit, Adams et Want [57] ont considéré que le contexte possède trois aspects importants qui consistent en des réponses aux questions suivantes : Où es-tu ? Avec qui es-tu ? De quelles ressources disposes-tu à proximité ?

Schilit et Theimer ont défini le contexte comme la localisation, la description de personnes et d'objets dans l'entourage et les changements à ces objets.

Brown, [58] a défini le contexte comme : « les éléments de l'environnement d'un utilisateur dont l'ordinateur à connaissance ».

Brown, Bovey et Chen [59] ont proposé un ensemble d'éléments extensibles pour caractériser le contexte dont les éléments de base sont : la localisation, l'ensemble des objets dont l'utilisateur a besoin, le temps et l'orientation spatiale (direction).

Ryan, Pascoe et Morse [60]: « les éléments du contexte sont : la localisation de l'utilisateur, l'environnement, l'identité et le temps ».

Wardet Hopper [61] : « les états des environnements possibles de l'application ». (Pascoe, 1998) : « ensemble d'états physiques et conceptuels ayant un intérêt pour une entité particulière ».

Schmidt et al [62] : « connaissances à propos de l'utilisateur et les états des équipements, l'entourage, la situation et la localisation ».

Brézillonet Pomerol [63] : « tout ce que n'intervient pas explicitement dans la résolution d'un problème mais le contraint ».

Chen et Kotz [64] : « ensemble des états environnementaux et paramètres qui déterminent le comportement d'une application ou dans lequel un événement de l'application se déroule et ayant un intérêt pour l'utilisateur ».

Dey [65] : « toute information qui peut être utilisée pour caractériser la situation d'une entité. Toute entité est une personne, ou un objet qui est considéré significatif à l'interaction entre l'utilisateur et l'application, incluant l'utilisateur et l'application eux-mêmes ».

Henricksen, Indulska et Rakotonirainy [66] : « la circonstance ou la situation dans laquelle une tâche informatique se déroule ».

Une analyse faite par Brezillon et al [67] concernant les définitions du terme contexte les a conduits à conclure que la plupart des définitions sont des réponses aux questions suivantes:

- **qui?** identité de l'utilisateur courant et d'autres personnes présentes dans l'environnement ;
- **quoi?** percevoir et interpréter l'activité de l'usager ;
- **où ?** localisation de l'usager, ou d'un événement du système ;
- **quand ?** repère temporel d'une activité, indexation temporelle d'un événement, temps écoulé de la présence d'un sujet à un point donné ;
- **pourquoi ?** il s'agit de comprendre la raison d'être de l'activité ;
- **comment ?** la manière de déroulement de l'activité.

Les réponses aux questions citées ci-dessus peuvent engendrer un grand ensemble d'informations dont une grande partie est inutile. Cela requiert également un plus grand effort pour la gestion de ces informations.

3.2.2 Catégories de contexte [20]

Etant donnée la diversité des informations composant le contexte, il est utile d'essayer de les classer par catégorie pour faciliter leur utilisation. Dans cette section, nous présentons une classification qui synthétise les informations contextuelles utilisées dans les solutions existantes. Dans ce domaine précis, plusieurs chercheurs ont proposé des catégorisations selon différentes approches. Schilit, Adams et Want [57] et Dey [65] ont catégorisé le contexte en deux classes : le contexte primaire qui contient les informations sur la localisation, l'identité, le temps et l'activité (statut) ; le contexte secondaire qui peut être déduit de ce dernier (exemple : de la localisation, on peut déduire les personnes à proximité). Chen et Kotz [64]

ont proposé deux catégories : le contexte actif qui influence le comportement d'une application et le contexte passif qui est nécessaire mais pas critique pour l'application. Petrelli et al [68] ont fait connaître le contexte matériel (localisation, machine, plateforme existante) et le contexte social (les aspects sociaux comme la relation entre les individus).Gwizdka, [70] a présenté deux catégories : le contexte interne contenant l'état de l'utilisateur et le contexte externe englobant l'état de l'environnement. Hofer et al [71] ont élaboré une catégorisation en deux classes : le contexte physique qui peut être mesuré par les capteurs physiques et le contexte logique qui contient les informations sur l'interaction (l'état émotionnel de l'utilisateur, ses buts, etc.).

3.2.3 Sources d'informations contextuelles [20]

Pour pouvoir assurer sa fonction principale qui consiste à l'adaptation selon le contexte, un agent doit faire la collection des informations contextuelles de différentes sources d'informations ayant des caractéristiques et des propriétés différentes. Selon (Henricksen, Indulska et Rakotonirainy [66], il y a trois méthodes pour l'acquisition d'information contextuelle. Ces trois méthodes sont :

- Capteurs physiques qui peuvent être intégrés ou non dans d'autres outils et permettant de capturer plusieurs types d'information physique telles que la température, la localisation géographique, niveau de bruit, lumière, etc. ;
- Les capteurs virtuels qui permettent d'extraireles informations contextuelles à partir des espaces virtuels tels que les programmes, systèmes d'exploitation, réseau, etc. (exemple : l'information sur la localisation peut être extraite à partir d'un calendrier électronique et d'un journal de connexion au réseau) ;
- Les capteurs logiques qui utilisent les informations des capteurs physiques et virtuels pour déduire d'autres informations.

3.2.4 Qualité de l'information contextuelle [20]

Les Systèmes Multi-Agents possèdent des environnements dynamiques et complexes**.** Cela conduit les informations contextuelles à contenir des erreurs au niveau de la collection, de l'interprétation et de la présentation de ces informations. Ainsi, pour assurer une adaptation fiable selon le contexte, une évaluation de la qualité de l'information contextuelle est requise.

Selon Krause et Hochstatter,[72], les informations contextuelles peuvent être erronées pour les raisons suivantes :

- l'information nécessaire (ou la source) n'est pas valable ;
- l'information n'est pas applicable à la situation actuelle ;
- les contraintes physiques et temporelles limitent la précision des capteurs ;
- les règles de raisonnement ne sont pas applicables dans toutes les situations (exemple : la détection d'une lumière dans une chambre ne veut pas dire toujours qu'il y a quelqu'un est présent) ;
- la possibilité d'acquisition d'informations de sources malicieuses.

3.2.5 Rôle du contexte [21]

3.2.5.1 Rôle du contexte au niveau des connaissances :

De nombreux faits et connaissances ne sont vrais que dans certains contextes ou changent de sens en changeant de contexte. Le contexte permet ainsi de définir le sens à donner à certains faits ou connaissances à chaque instant, en mettant de côté les connaissances qui ne sont pas vérifiées dans ce contexte ou les sens inadaptés. Le contexte permet, à chaque instant, de définir quelles connaissances peuvent être considérées, quelles sont leurs conditions d'activation, quelles sont leurs limites de validité et quand les utiliser.

3.2.5.2 Rôle du contexte au niveau des raisonnements :

Le contexte est également un point clé des raisonnements. Ceux-ci consistent en la comparaison d'une situation donnée à des situations déjà traitées, afin de déterminer leurs points communs et différences. Partant de ces résultats, ces raisonnements proposent des solutions par adaptation des solutions passées les plus proches. Le principe de ces raisonnements fait référence aux situations et à leurs similarités. Toutefois Le contexte n'intervient pas seulement au niveau de la description des situations, mais également dans la recherche de similarité entre deux situations.

3.2.6 Systèmes tenant compte du contexte :

3.2.6.1 Le système ORCA : [22] [23]

Le système ORCA de Turner [73] de contrôle de véhicules sous-marins autonomes. Le comportement de ce système est fondé sur le principe qu'un agent doit avoir des connaissances explicites sur les situations qu'il peut rencontrer, afin d'adapter son comportement à celles-ci (Context-Mediated behavior [74]). Pour atteindre ce but, Turner [75] propose l'utilisation de schémas contextuels (c-schemas) qui contiennent chacun des

connaissances descriptives sur un contexte précis et des connaissances prescriptives sur le comportement approprié de l'agent dans ce contexte. Les connaissances descriptives permettent de savoir dans quelles conditions le c-schema correspondant peut être appliqué et quelles sont les connaissances qui sont vraies dans ce contexte. Par exemple, si le système remarque que la profondeur est faible et que l'eau est très peu agitée, il peut conclure qu'il se trouve probablement dans un port et donc que le risque de croiser des bateaux est plus grande (et donc que le système doit anticiper d'avantage cette éventualité). Ces connaissances peuvent également lever l'ambiguïté sur certains éléments perçus. Les connaissances prescriptives modifient le comportement. Des actions sont autorisées ou interdites, d'autres sont réalisées différemment en fonction des contextes. La planification des actions peut également être dépendante du contexte. Ces connaissances orientent aussi la perception vers des éléments non-identifiés précédemment (car jugé peu important dans les contextes précédent). Enfin ces connaissances permet de gérer les événements de la situation rencontrée.

Les c-schemas sont organisés dans une hiérarchie de généralisation/spécialisation. Cette hiérarchie facilite grandement la recherche des schémas correspondants à la situation rencontrée. A chaque modification importante de l'environnement, le système de contrôle compare les caractéristiques de la situation et les descriptions des contextes des c-schemas, puis combine les c-schemas retenus pour obtenir un comportement la mieux adaptée à la situation. Ce mécanisme de combinaison de schémas est intéressant car il permet d'obtenir une ligne de comportement spécifique à une situation relevant de plusieurs situations connues. Ainsi quand le système se trouve dans un port et que ses batteries sont faibles, il peut adopter un comportement ad-hoc bien que le c-schema adéquat n'existe pas. La combinaison des c-schemas « dans un port » et « batterie faible » suffisent pour permettre à l'automate de conclure que la profondeur est faible et qu'il peut se poser sur le fond en attendant d'être récupéré. La combinaison de c-schemas peut faire apparaître des conflits quand des informations sont contradictoires. Plusieurs approches sont alors possibles : soit on ne fait rien et on laisse le raisonnement régler le conflit ; soit on conserve la meilleure information (celle avec le plus grand coefficient de certitude) ; soit on combine les valeurs. La sélection de l'attitude à avoir lors de conflit dépend de la nature des informations en conflit et s'opère donc dynamiquement chaque fois que ceci est nécessaire.

Une fois le comportement de système est adoptée, le système de contrôle ne s'inquiète plus du contexte tant que la situation ne change pas notablement. Les éléments contenus dans le

c-shema suffisent à adapter le comportement de l'automate. Les principaux avantages de cette approche sont rappelés par Turner [76] :

- Le système est efficace : la recherche du contexte ne se fait que lors de changements importants de la situation.
- Le comportement de contexte est adapté au contexte et peut même contenir des « réflexes » appropriés, ce qui rend le système particulièrement réactif en minimisant les processus de décisions.
- Les c-schemas peuvent contenir des informations permettant de mieux comprendre des éléments de l'environnement, de focaliser l'attention sur des éléments importants dans ce contexte, d'aider à la prédiction d'événements inattendus.
- La représentation explicite du contexte permet aux agents d'ajuster leurs connaissances contextuelles en fonction des expériences passées. Elle autorise aussi la manipulation de contexte obtenu par la combinaison des contextes proches.

Le modèle des c-schemas est très général et peut être appliqué dans de nombreuses applications concernant des agents autonomes. Le système ORCA est une application particulière des c-schemas pour des véhicules sous-marins autonomes. Il représente une implémentation effective des idées soutenues et est parfaitement opérationnel.

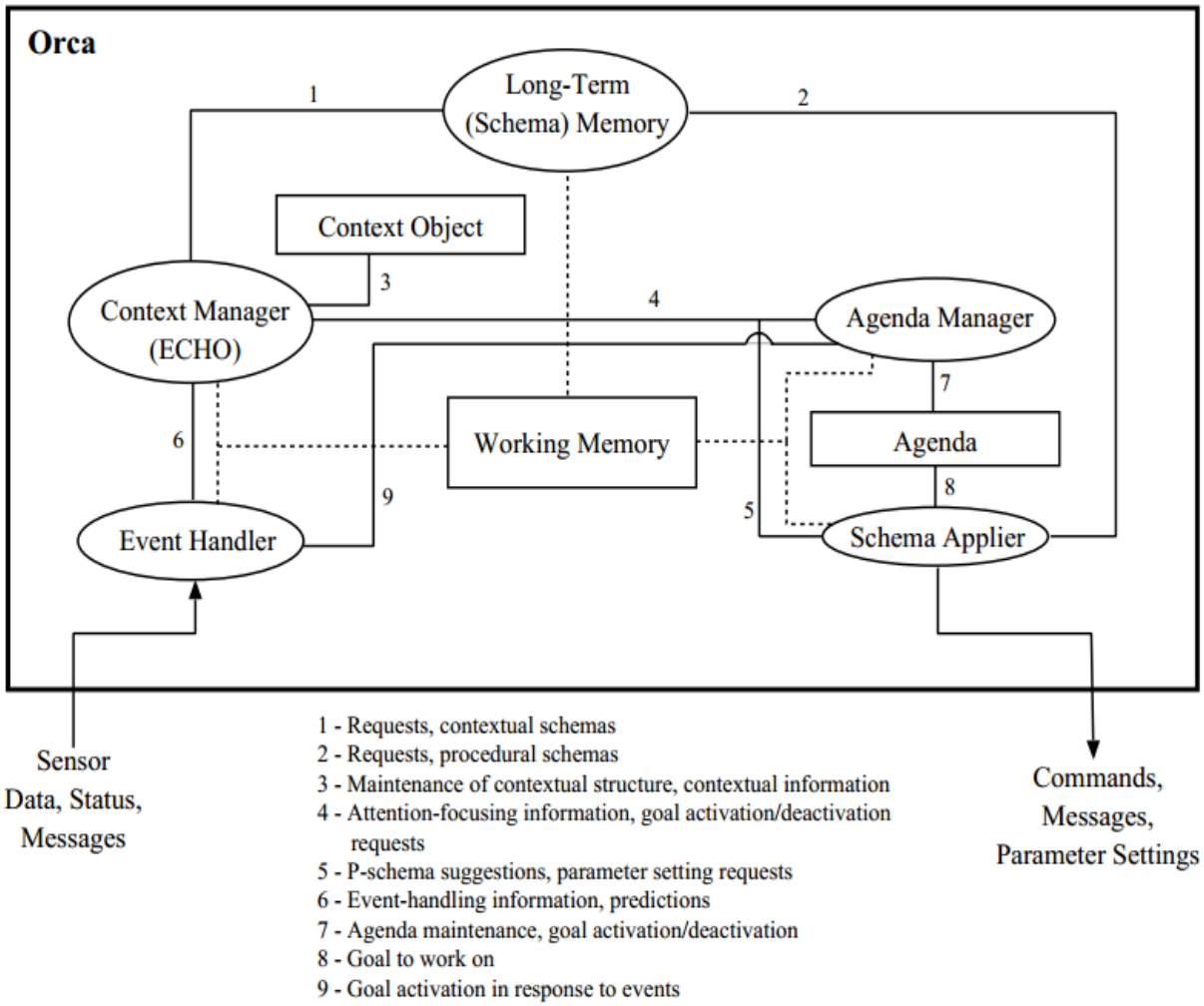


Figure 20: Structure interne d' ORCA. [23]

3.2.6.2 Context-Based Reasoning: [21]

Le Context-Based Reasoning (CxBR) est un modèle de raisonnement basé sur le contexte introduit par Gonzalez et Ahlers [77]. Les auteurs travaillent sur les structures de contrôle de plates-formes autonomes intelligentes (PAI) et sur l'adaptation de leur comportement au contexte. Sur ce point leurs travaux sont proches de ceux de Turner présentés ci-dessus.

Les structures de contrôle des PAI gèrent une séquence continue de contexte qui change quand la situation change. Le comportement d'un PAI est contrôlé par le contexte actif à cet instant, qui peut être différent d'un PAI à un autre dans une même situation (à cause des spécificités des missions, des capteurs et des capacités de chacun d'entre eux). A chaque instant au moins un contexte est actif. Quand plusieurs le sont, ils doivent être compatibles et l'un d'eux correspond plus précisément au centre d'intérêt courant. Les contextes sont définis

sur des intervalles de temps et peuvent définir des transitions vers des buts ou être des buts en soi. Un nombre limité d'événements est autorisé pour chaque contexte. Ceci permet de limiter l'espace de résolution d'un problème. L'avènement d'un nouveau contexte modifie plus ou moins le cours des actions ou les attentes de résultat de celles-ci. Le nouveau et l'ancien contexte peuvent soit être mutuellement exclusif (dans quel cas le nouveau contexte élimine le contexte précédent), soit non-mutuellement exclusifs (dans quel cas le nouveau contexte recouvre le contexte précédent puis rends la place à celui-ci quand il n'est plus actif).

Les auteurs utilisent des Scripts [78] pour représenter les contextes. Ainsi le modèle de chaque contexte contient des champs fixes, des champs variables, des procédures et des règles. Les champs fixes décrivent les connaissances statiques dans ce contexte (capacités, types et champs d'action des capteurs…). Les champs variables contiennent des variables du PAI (position, vitesse…). Les procédures décrivent des commandes de base du PAI. Les règles sont de trois types. Premièrement, les règles sentinelles : elles surveillent continuellement les données à l'affût de facteurs pouvant modifier la situation. Elles sont à la base de la situation-awarness. Deuxièmement, les règles de transition : elles réagissent à une situation identifiée par les règles sentinelles et déterminent quels contextes peuvent être activés. Troisièmement, les règles internes : elles servent à prendre des décisions dans le cadre d'un contexte.

Le CxBR [77] est fondé sur trois types de contextes : le contexte de mission, les contextes majeurs et les sous-contextes. Le contexte de mission définit les objectifs et les contraintes de la mission. Ainsi ce contexte est fixé une fois pour toute en début de mission et restera actif jusqu'à la fin de celle-ci. Les contextes majeurs encadrent le comportement stratégique des PAI. Ils contiennent des informations sur le contrôle du PAI et sur les règles de désactivation de ce contexte et sur les contextes pouvant les suivre. A chaque instant un et un seul contexte majeur est actif. Les sous-contextes encadrent le comportement tactique du PAI. Ces contextes ne sont pas mutuellement exclusifs et ne sont pas nécessaire à chaque instant. Ils délimitent plus précisément le comportement en fonction de détails de la situation. Cette approche distinguant trois niveaux de contexte est originale. Elle permet de décrire le contexte de manière opérationnelle : le contexte de la mission définit les objectifs, le contexte majeur dirige les raisonnements stratégiques et définit les buts à atteindre en fonction de la situation, et enfin les sous-contexte restreignent, si nécessaire, les actions du PAI pour les adapter à des situations particulières.

3.2.7 Architecture générale d'un système sensible au contexte [19]

Un système sensible au contexte est caractérisé par des fonctionnalités qui lui permettent d'interagir avec des capteurs, d'analyser les données collectées et d'agir en conséquence. Plusieurs travaux ont proposé des architectures sensibles au contexte. Un accord général, dans ces travaux, sur la séparation entre la capture des informations de contexte et l'utilisation de ces informations. La conception des systèmes sensibles au contexte se base sur quater couches principales : capture de contexte, interprétation de contexte, gestion de contexte et adaptation au contexte (figure 21).

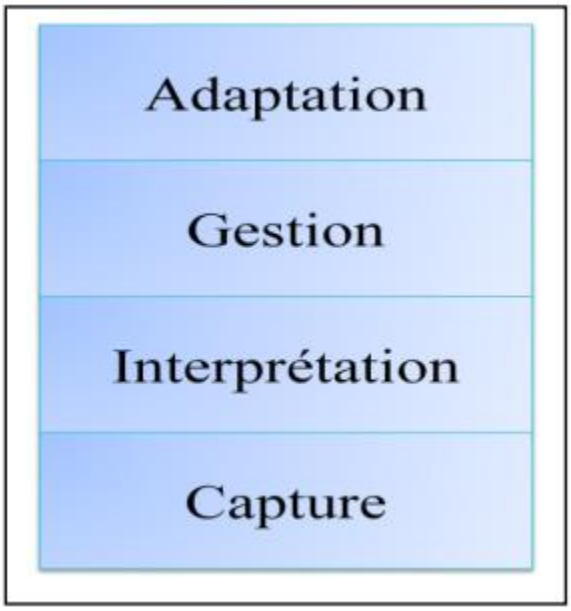


Figure 21: Architecture générale d'un système sensible au contexte [19]

3.2.7.1 Couche de capture du contexte :

La capture du contexte représente le mécanisme de collecte des données à partir de l'environnement qui entoure le système. Cette couche est composée d'une collection de capteurs. Un capteur est une source matérielle ou logicielle qui peut générer une information contextuelle. Nous distinguons trois types de capteurs : physiques, virtuels et logiques.

- **Capteurs physiques :** Les capteurs physiques sont des dispositifs matériels qui sont capables de fournir des données de contexte. Le tableau suivant donne quelques exemples de capteurs physiques selon le type d'information qu'ils fournissent.

Type d'information fournie	Capteurs disponibles
Lumière	Photodiodes, capteurs de couleurs, capteurs d'infrarouge et d'ultra violet…
Contexte visuel	Caméras numériques
Audio	Microphones
Localisation	GPS (Global Positionning System), GSM (Global System for Mobile Communications), Active Badge System [Want92]…
Mouvements et accélérations	Capteurs d'angles, accéléromètres, détecteurs de mouvement, champs magnétiques…
Température	Thermomètres numériques
Caractéristiques biologiques	Capteurs Biométriques (Tension, résistance de peau…)

Tableau 5: Exemples de capteurs de contexte [20**]**

- **Capteurs virtuels :** Les capteurs virtuels fournissent des informations contextuelles à partir d'applications ou services logiciels. Par exemple, il est possible de détecter l'emplacement d'un livreur de marchandise en consultant son carnet électronique de rendez-vous sans avoir recours à des capteurs physiques.

- **Capteurs logiques :** Ce type de capteurs utilise généralement plusieurs sources d'information contextuelles pour fournir une autre information de synthèse plus précise. Ces capteurs peuvent réutiliser des capteurs physiques et virtuels pour fournir un contexte de plus haut niveau. Par exemple, un capteur logique peut fournir l'emplacement d'un caissier dans un grand magasin en analysant les différentes sessions ouvertes sur les caisses du magasin.

3.2.7.2 Couche d'Interprétation :

Cette couche a pour but d'interpréter les données contextuelles fournies par les capteurs. Elle sert à l'analyse et à la transformation des données brutes, fournies par la couche de capture,

d'autres formats plus expressifs à l'application. Par exemple, les coordonnées GPS d'une personne peuvent être moins significatives qu'une adresse physique sous forme de numéro de rue et de ville.

3.2.7.3 Couche de gestion du contexte :

A ce niveau, le contexte capturé et interprété doit être bien géré pour faciliter l'utilisation. La gestion de contexte contient le stockage et la représentation formelle des informations de contexte.

3.2.7.4 Couche adaptation au contexte:

Cette couche représente l'endroit où les réactions aux changements du contexte sont mises en œuvre.

3.2.8 Modélisation du contexte [20]

Pour faciliter la tache d'adaptation du système à des situations contextuelle, il est essentiel d'utiliser un modèle efficace permettant la collecte, le stockage, l'interprétation et l'analyse des informations de contexte. Plusieurs approches de modélisation utilisée ont été proposées. Dans cette partie on peut constater que certaines modèles sont couramment utilisées dans la littérature :

3.2.8.1 Les modèles Attribut/Valeur :

La représentation attribut/valeur est la structure des données la plus simple pour la modélisation des informations contextuelles. Elle a été proposée au départ par Schilit, Adams et Want, [57] pour la gestion des informations contextuelle d'un environnement.

Plusieurs architectures présentent le contexte sous forme de paires (attribut, valeur). L'attribut représente un nom d'une information contextuelle. La valeur représente la valeur actuelle de cette information. Par exemple, (Name="context1", User="doctorEH102", Localisation="Edouard Herriot Hospital", Time="Mon Jul 09 16:51:20 CEST 2007"). Le contexte context1 est défini par "l'utilisateur x est localisé dans un emplacement y à un temps t". Cette méthode présente l'avantage de la facilité d'implantation. En effet, la gestion du contexte revient à parcourir la liste des contextes disponibles. Cependant, ce modèle manque d'expression et de complétude.

En effet, sa structure trop « plate » ne permet pas de définir tous les aspects contextuels de l'application. Ce genre de modèle est aussi une source de conflits. Par exemple, si nous

définissons un nouveau contexte (Name="context2", User="x", Localisation="z", Time="t") avec l'emplacement z est « dans l'hôpital Edouard Herriot » (par exemple la chambre 220 de l'hôpital), nous ne pouvons pas dire que context2 est un sous contexte de context1 et que toutes les fonctionnalités offertes par l'application dans context1 doivent aussi exister dans context2. Mais cette approche est très utilisée dans les services distribués (les services sont décrits en général avec une liste d'attributs sous forme d'attribut/valeur et la découverte de service est ensuite appliquée par utilisation d'un algorithme qui utilise ces paires attribut/valeur)

3.2.8.2 Les modèles de représentation par balises : [20]

Cette représentation est formée d'une structure de donnée hiérarchique constituée de balises avec des attributs et des contenus. Ces derniers peuvent être définis récursivement par d'autres balises. Comme elle est formée de balises, elle utilise des langages dérivés du SGML (Standard Generic Markup Language) en particulier, le XML (eXtended Markup Language). Par exemple nous pouvons citer quelque exemple des langages qui utilise cette approche :

- **ConteXtML :** Le ConteXtML (Contexte Markup Language) est un protocole simple basé sur XML pour échanger des informations contextuelles entre un client mobile et un serveur. Les messages ConteXtML sont regroupés dans des balises <context> ou des éléments. Ci-dessous un exemple (Figure22) proposé par Ryan (Ryan, 2007) qui présente un message ConteXtML qui peut être envoyé par un client pour indiquer sa localisation courante (l'élément <spatial>) et qui nécessite une note contenant un élément de donnée nommé « landuse » et ayant une valeur « pasture » (l'élément <require>).

```
<context session="123" action="update">
<spatial proj="UTM" zone="33" datum="Euro 1950 (mean)">
<point x="281993" y="4686790" z="205" />>
</spatial>
<require>
<note>
<data name="landuse" value="pasture" />
</note>
</require>
</context>
```

Figure 22: exemple d'un message ConteXtML. [20**]**

- **CC/PP :** Le langage CC/PP (Composite Capabilities / Preferences Profiles), est une recommandation de W3C (World Wide Web Consortium) dans le cadre des travaux sur « device dependance » pour supporter la négociation de contenu entre un navigateur web et un serveur. Il est basé sur RDF (Resource Description Framework) qui est utilisé afin de créer des profils décrivant les capacités des terminaux et les préférences d'un agent utilisateur. CC/PP est utilisé pour personnaliser le contenu sur la base de ses capacités et ses préférences. Nous donnons ci-dessous (Figure 23) l'exemple d'un profil CC/PP d'un ordinateur portable.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.etsmtl.ca/2007/04/01-rdf-syntax-ns#"
xmlns:prf="http://www.etsmtl.ca/TR/WD-profile-vocabulary#">
<rdf:Description about="HardwarePlatform">
<prf:Defaults
Vendor="Acer"
Model="TravelMate2428AWXMi"
Type="LapTop"
ScreenSize="14.1"
CPU="IntelPentiumM"
Speed="1.7GHz"
Memory="512mB"
HardDisk="60Gb"
Dvd="Yes"
Wireless="Yes"
Bluetooth="YES"
Speaker="Yes" />
<prf:Modifications
Memory="1gB" />
```

Figure 23: exemple d'un profil CC/PP d'un ordinateur portable. [20**]**

3.2.8.3 Les modèles graphiques :[20]

Cette approche consiste à modéliser les informations contextuelles selon un graphe conceptuel. Bauer [84] a utilisé une représentation graphique basé sur UML (Unified Modeling Language) pour la modélisation des informations contextuelles pour le susteme de contrôle du trafique aérien.

(Henricksen, Indulska et Rakotonirainy, [66]) ont proposé un modèle graphique qui se base sur le formalisme « entité/association ». Les éléments du contexte (utilisateur, machine, réseau, etc.) sont représentés par des entités avec des attributs et en associations entre eux (une association peut aussi avoir des attributs). Ils ont fourni aussi une notation graphique pour leur concept de modélisation.

(Henricksen et Indulska [79]; Henricksen, Indulska et McFadden [80]) ont développé une approche de modélisation graphique basée sur la méthode ORM (Object Role Modeling). C'est une méthode orientée « fait » pour l'analyse de l'information au niveau conceptuel, en particulier pour les bases de données relationnelles. La modélisation dans ORM consiste à identifier les types des faits appropriés et les rôles des types d'entités. Selon les auteurs, cette extension est plus formelle et plus expressive pour capter différents types d'informations contextuelles ; elle appuie le raisonnement sur le contexte, décrit bien les informations imparfaites et résout l'ambiguïté dans l'information contextuelle. Cette méthode qui a subi des améliorations est devenue CML (Context Modeling Language)[81] et apparaît comme une extension vers une représentation basée sur XML, XCML

3.2.8.4 Les modèles orientés objets :

Le but de la modélisation du contexte par l'approche orientée objet est de profiter de la puissance offerte par les mécanismes de cette technique (encapsulation, réutilisation, héritage,…). Parmi les travaux qui ont abordé cette méthode, le travail de Hofer et al. [22] qui ont introduit l'approche "HYDROGEN ". Ils ont proposé une architecture à 3 niveaux pour la modélisation du contexte dans le domaine de l'informatique mobile. Le contexte est modélisé sous forme des diagrammes de classes UML. Ils ont décomposé le contexte en contexte local et en contexte distant (les machines distantes en communication avec la machine locale). Chaque type de contexte est composé de plusieurs objets contexte qui constitue la superclasse de plusieurs éléments du contexte, notamment : le temps, le réseau, la localisation, l'utilisateur, la machine et autres qui peuvent être ajoutés par héritage de la super classe (Figure24).

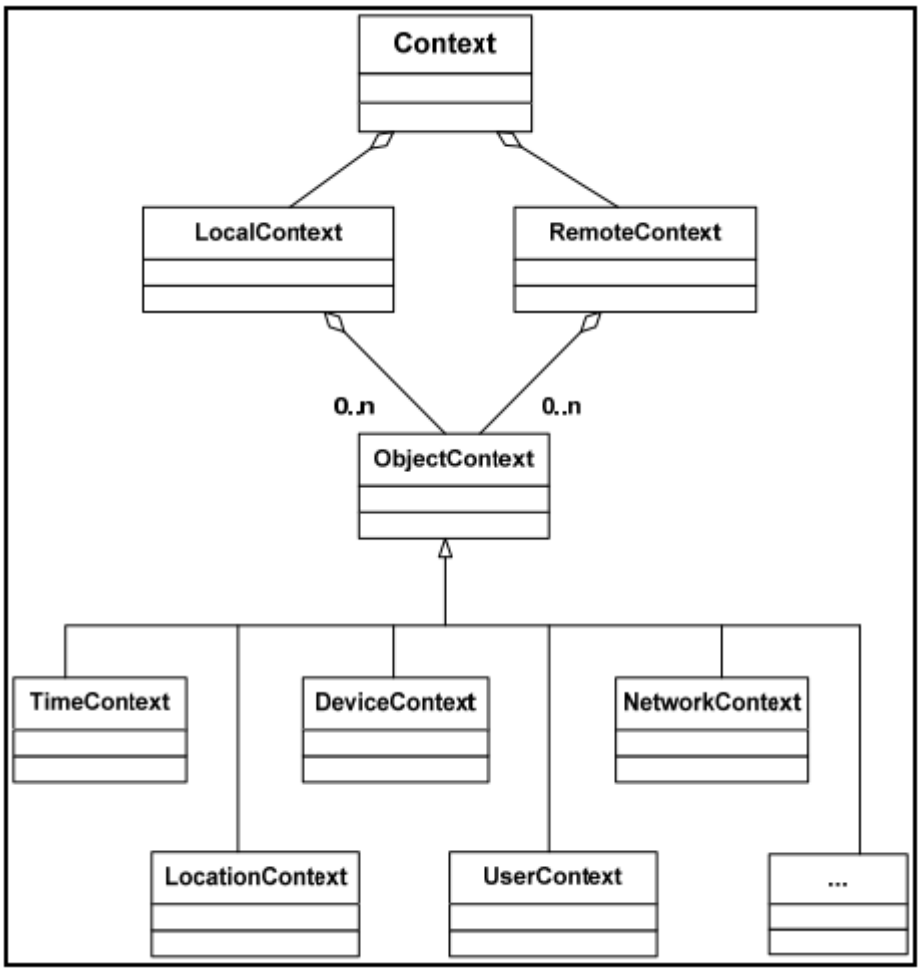


Figure 24: Le modèle UML de l'approche Hydrogen. [20**]**

3.2.8.5 Les modèles logiques :

Dans un modèle logique, les informations contextuelles doivent être représentées d'une façon formelle comme des faits. Un processus de raisonnement ou d'inférence est ensuite utilisé pour déduire des nouveaux faits en se basant sur des règles existant dans le système. Ces méthodes permettent une représentation formelle du contexte et offrent un mécanisme d'inférence, offrant la possibilité d'abstraire les données.

Parmi les premiers travaux de modélisation du contexte utilisant cette approche logique et celui de Carthy et Buvac, [83] qui ont introduit le contexte comme objet formel. La relation de base dans cette approche est ist(c,p) qui signifie que la proposition p est vraie dans le contexte c.

3.2.8.6 Les modèles ontologiques :

Une ontologie est un ensemble structuré de concepts. Les concepts sont organisés dans un graphe dont les relations peuvent être des relations sémantiques ou des relations de composition et d'héritage. L'objectif d'une ontologie est de modéliser un ensemble de connaissances dans un domaine donné. Les ontologies représentent actuellement une solution favorisée pour modéliser les informations du contexte.

Parmi les travaux s'intéressant à une modélisation par les ontologies, Chen, Finin et Joshi [82] ont proposé une approche fondée sur l'idée d'un courtier de contexte (Context Broker Architecture ou CoBrA). Il définit une collection d'ontologies appelée COBRA-ONT pour la modélisation de contexte dans un environnement d'une salle de rencontre intelligente. COBRA-ONT des concepts typiques associés avec des places, des agents et des événements. Les ontologies jouent un rôle important dans CoBrA. Elles aident le courtier de contexte à partager les informations contextuelles avec d'autres agents et lui permet de raisonner sur le contexte.

Chapitre 3 : l'architecture BDI proposée

4.1 Introduction

Les architectures d'agents ont été conçues et implémentées pour fonctionner dans les environnements sensibles au contexte. Ces architectures d'agents ont réussi à fournir des systèmes intelligents à usage général. Mais la complexité des environnements nécessite le développement de quelques points dans ces architectures tel que la possibilité de représenter le contexte, de mettre à jour la structure de données de contexte, et de sensibilité au contexte.

La représentation explicite de contexte et de connaissance contextuelle est importante pour les agents intelligents. Dans cette section nous discutons notre vue sur le contexte et le raisonnement sensible au contexte.

4.2 Architecture d'agent BDI et le contexte :

L'architecture d'un agent est une description de son organisation interne : les données, les connaissances, et les processus internes d'un agent lui permettant de prendre une décision consistant à choisir une action. La communauté multi-agents distingue deux grandes catégories d'architectures d'agents : les architectures cognitives et les architectures réactives. Cependant, il est possible de combiner les deux architectures d'agents pour obtenir des architectures hybrides.

4.2.1 L'architecture d'un agent BDI :

Dans le domaine des modèles cognitifs, l'architecture BDI est basée sur un courant philosophique (par exemple les travaux de Bratman [28] et formalisée principalement par les logiques modales (Rao et Al [29] ou Cohen et Al [11]). Ces logiques décrivent les trois principaux états intentionnels (Croyances, Désirs, Intentions) qui régissent le comportement d'un agent :

- **Croyance (**Belief) : Les croyances d'un agent sont les informations que l'agent possède sur l'environnement et sur d'autres agents qui existent dans le même environnement. Les croyances peuvent être incorrectes, incomplètes ou incertaines et, à cause de cela, elles sont différentes des connaissances de l'agent, qui sont des informations toujours vraies. Les croyances peuvent changer au fur et à mesure que l'agent, par sa capacité de perception ou par l'interaction avec d'autres agents, recueille plus d'informations.
- **Désir** (Desire) : Les désirs d'un agent représentent les états de l'environnement, et parfois de lui-même, que l'agent aimerait voir réalisés. Un agent peut avoir des désirs

contradictoires ; dans ce cas, il doit choisir parmi ses désirs un sous-ensemble qui soit consistant. Ce sous-ensemble consistant de ses désirs est parfois identifié avec les buts de l'agent.

- **Intention** (Intention) : Les intentions d'un agent sont les désirs que l'agent a décidés d'accomplir ou les actions qu'il a décidées de faire pour accomplir ses désirs. Même si tous les désirs d'un agent sont consistants, l'agent peut ne pas être capable d'accomplir tous ses désirs à la fois.

L'architecture BDI est largement adoptée dans la communauté SMA et ces trois concepts centraux sont une constante dans le nombre considérable de formalisations et d'implémentations existantes de cette architecture. Elle peut aussi être étendue avec d'autres concepts déontiques ou sociaux tels que les émotions ou les normes.

Le modèle BDI a inspiré beaucoup d'architectures d'agents cognitifs. Nous pouvons citer : Le système de raisonnement procédural PRS (Procedural Reasoning System) qui est un exemple d'architecture d'agent basée sur le modèle BDI. Le PRS a été développé, initialement, en 1987 par Georgeff et Lansky. Cette architecture est la base pour presque tous les systèmes BDI.

4.2.2 Contexte

Dans cette section, nous devons expliquer ce que nous entendons par le contexte dans les agents BDI. Ainsi que de fournir certaines définitions connexes.

Définition 1: Dans ce travail, nous utilisons l'expression **état de l'environnement** pour désigner l'état de l'environnement de l'agent à un instant t, c'est-à-dire les caractéristiques de l'environnement et tous les objets existants, les caractéristiques de l'état interne de l'agent et la relation entre eux.

Définition 2: La situation est une partie de l'état de l'environnement. La situation observée est l'ensemble des caractéristiques que l'agent croit appartenir à sa situation actuelle.

Selon cette définition, la situation actuelle comprend tous les caractéristiques de l'environnement qui affectent sur l'agent. La connaissance de l'agent d'une situation particulière est plus importante que la somme des caractéristiques observables par l'agent. Un agent peut prévoir dans une situation des caractéristiques qu'il n'a pas encore vus.

Définition 3: Un contexte est une collection de caractéristiques possibles de l'environnement qui a une valeur à l'agent.

Chez n'importe quel agent intelligent, le contexte affecte sur son comportement. Cette caractéristique est étudiée dans la psychologie et la sociologie. Dans ce travail, nous voulons ajouter cette caractéristique aux agents intelligents comme les agents BDI afin d'améliorer leurs comportements dans les différents environnements.

4.3 L'architecture BDI proposée :

Pour intégrer le contexte dans les agents BDI, nous devons résoudre trois problèmes:

1. Comment représenter le contexte de l'agent?
2. Comment mettre à jour la structure de données de contexte?
3. Comment le contexte affecte dans le processus de prise de décision?

La figure25 montre l'architecture de notre modèle. Notre approche diffère des modèles BDI classiques par la prise en compte de contexte. Pour cela, nous avons besoin d'une base de connaissances pour représenter le contexte de l'agent. Dans cette nouvelle structure, l'agent peut gérer toutes les informations de sa situation.

Le processus de raisonnement pratique d'un agent BDI est résumé dans la figure 25. Comme cette figure l'illustre, on trouve dans un agent BDI, les composantes principales suivantes:

- Un ensemble de contextes, représentant des connaissances contextuelles que l'agent possède sur les différentes situations possibles dans l'environnement, et les objectifs à atteindre dans chaque situation ;
- Un module de gestion de contexte (gestionnaire de contexte) qui prend en entrée les nouvelles perceptions pour déterminer les meilleures situations de contexte qui caractérisent la situation actuelle de l'environnement et construisent une structure de connaissances contextuelles qui représentent une situation plus claire de l'environnement et déterminent les objectifs à atteindre dans cette situation. La connaissance contextuelle fournie par ce module est répartie sur d'autres modules de raisonnement de l'agent BDI;
- Un ensemble de croyances actuelles qui représente des informations actuelles que l'agent possède sur l'environnement;
- Une fonction de révision de croyance (BRF) basée sur les informations de la situation actuelle du contexte fournie par le gestionnaire de contextes pour la mise à jour de l'ensemble des croyances et d'assurer la cohérence entre les différentes informations dans la base de croyance ;

- Une fonction de génération d'options (options) qui détermine les options disponibles pour l'agent (ses désirs), en se basant sur les croyances courantes sur son environnement, les objectifs de la situation actuelle de contexte fournie par le gestionnaire de contexte et de ses intentions actuelles;
- Un ensemble d'options en cours, ce qui représente des options disponibles à l'agent;
- Une fonction de filtre (filtre) représente le processus de délibération de l'agent, et détermine les intentions de l'agent à base de ses croyances, ses désirs et ses intentions;
- Un ensemble d'intentions actuelles;
- Une fonction de sélection d'action (exécuter), détermine une action à effectuer en se basant sur les intentions actuelles.

Dans le reste de cette section, nous expliquons en détail le module de gestionnaire de contexte, et son rôle pour améliorer le comportement de l'agent BDI dans les différentes situations d'environnent. Nous discutons aussi la représentation de connaissance contextuelle dans la base de contexte.

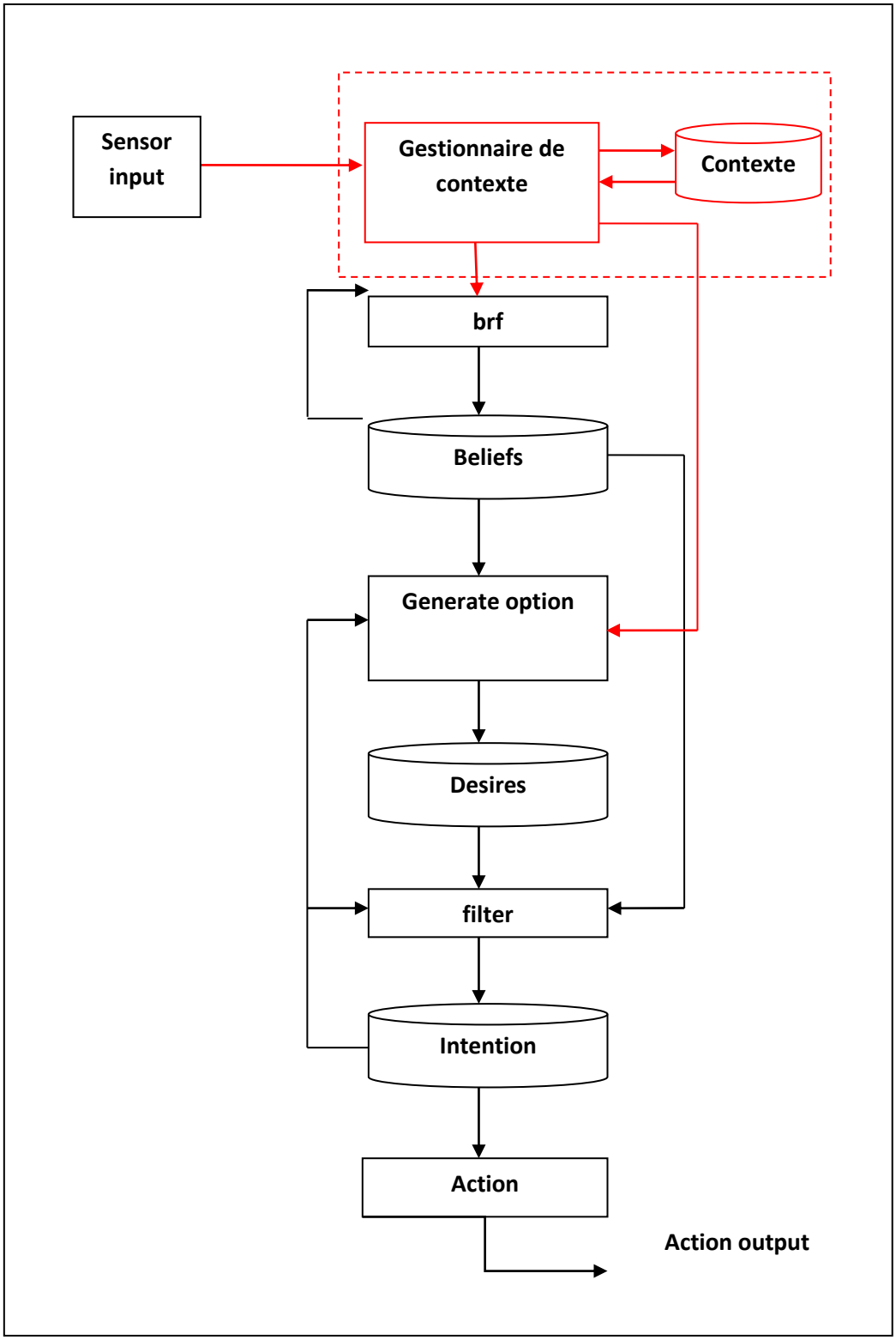


Figure 25: Notre architecture BDI

4.3.1 La représentation de connaissance contextuelle

Dans cette approche le contexte est représenté par des situations contextuelles. Chaque situation contextuelle contient une connaissance descriptive et prescriptive sur le contexte représenté :

- **La connaissance descriptive se compose de :**
  - Caractéristique de la situation qui doit être présente.
  - Caractéristique de la situation qui doit être non présente.
  - Caractéristique de la situation qui peut être encore invisible, et devrait être présente dans ce contexte et permettre à l'agent de reconnaitre des évènements prévus.
- **La connaissance prescriptive se compose de :**
  - La gestion des évènements imprévus : Comment détecter, comment diagnostiquer, leur importance au contexte et comment les gérer ?
  - Attirer l'attention : Quels objectifs (doivent /ne doivent pas) être atteints et l'importance des objectifs particuliers dans ce contexte ?

Pour construire la base du contexte par les situations contextuelles possibles de l'environnement nous proposons le processus suivant:

4.3.1.1 Détermination des éléments du contexte :

Une situation du contexte est une conjonction de valeurs d'éléments qui la compose :

$Valeur_{i1}(E_1)$, $Valeur_{i2}(E_2)$, $Valeur_{i3}(E_3)$,……., $Valeur_{in}(E_n)$.

Pour chaque élément du contexte, nous spécifierons l'ensemble des valeurs possibles. (par exemple niveau de lumière (haut,bas), localisation (à la maison, à l'université, etc.). Selon le nombre d'éléments du contexte (qu'on suppose égale à N), nous obtiendrons N vecteurs de différentes tailles.

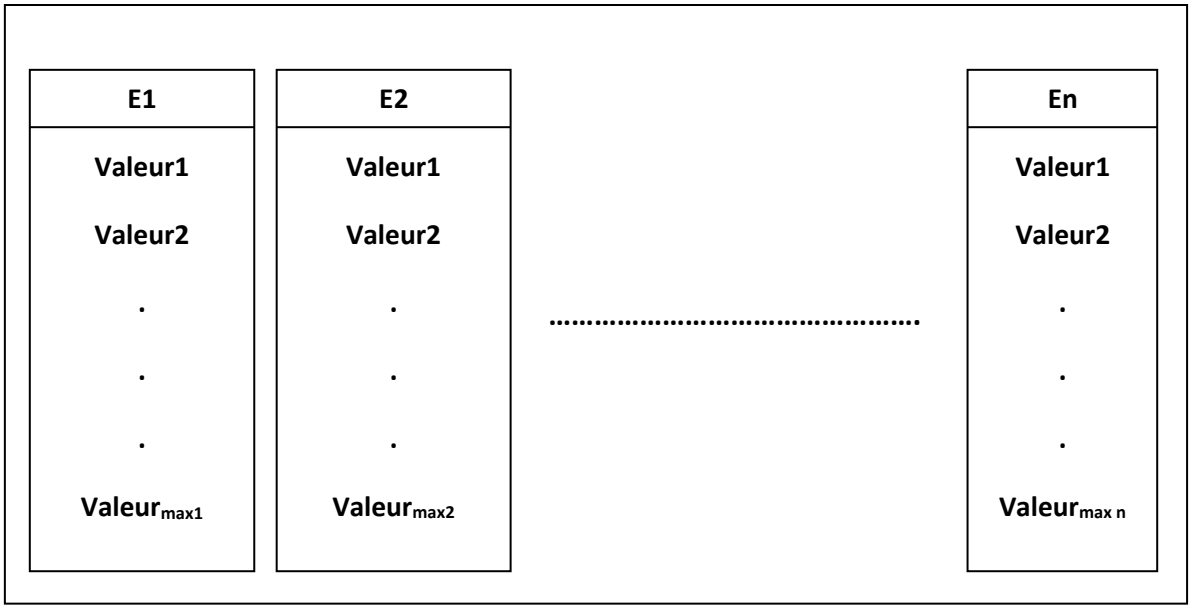


Figure 26: Éléments du contexte contenant différentes valeurs.

4.3.1.2 Enumération de toutes les situations possibles du contexte :

Cette tâche consiste à énumérer toutes les situations possibles du contexte. Le nombre total de situation possibles pour les éléments du contexte de la **Figure26** peut être calculé selon cette formule :

$$\boldsymbol{Max} = \prod_{\boldsymbol{i=1..n}} \boldsymbol{max_i}$$

Alors la situation du contexte est définie par :

$$\boldsymbol{SC_i(valeur_{i1}, valeur_{i2}, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, valeur_{in})}$$

$$\mathrm{i_1} = 1..\mathrm{max_1}, \mathrm{i_2} = 1..\mathrm{max_2}, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, \mathrm{i_n} = 1..\mathrm{max_n},$$

Chaque situation du contexte peut être interprétée comme conjonction de différentes valeurs des éléments de contexte.

L'ensemble des situations de contexte peut inclure quelques situations de contexte sans signification. Nous avons besoin de certaines procédures pour éliminer les situations non significatives. Et pour éliminer une situation du contexte, nous allons utiliser certaines règles contradictoires composées d'au moins deux valeurs contradictoires de deux éléments du contexte. La procédure prendra ces règles une par une et passera en revue toutes les situations du contexte et éliminera celles qui contiennent les éléments de la règle contradictoire.

4.3.1.3 Association

Cette étape consiste à associer à chaque situation du contexte les objectifs et les évènements appropriés dans ce contexte, c'est-à-dire selon les valeurs des éléments du contexte l'agent sera fournie les objectifs et les événements pour déclencher les plan les plus approprie à ce contexte.

4.3.2 Le gestionnaire de contexte

Le gestionnaire de contexte est un mécanisme qui nous permet d'assurer que l'agent se comporte de façon appropriée à son contexte. Il représente explicitement la connaissance contextuelle d'un agent et s'assure que la connaissance est disponible au moment où elle est nécessaire.

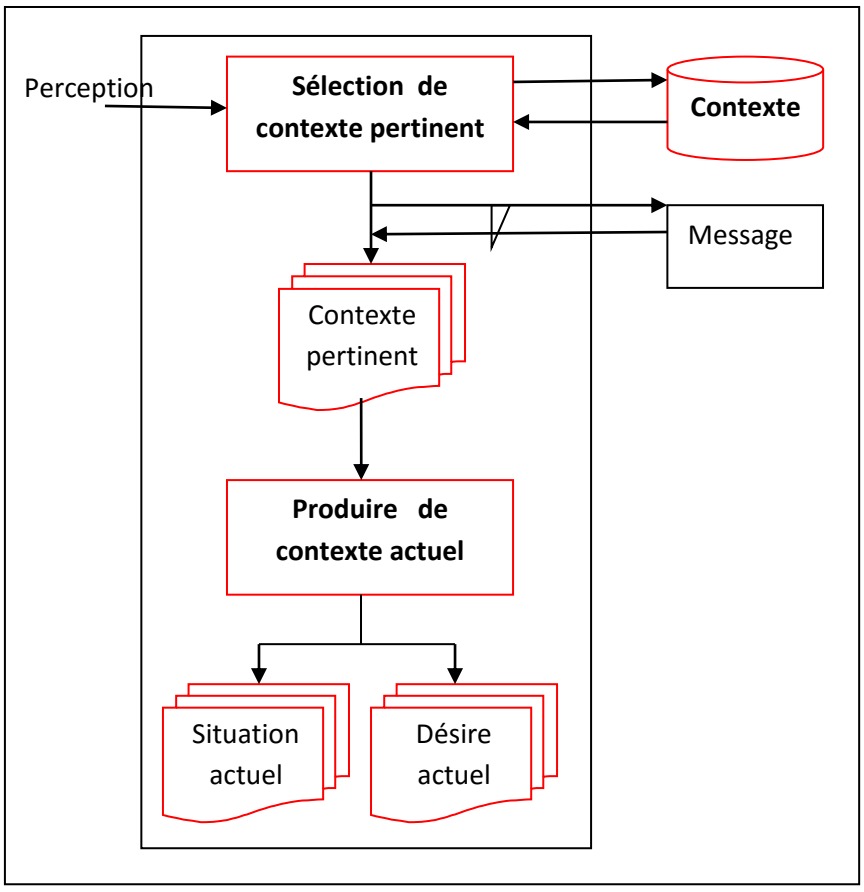


Figure 27: Le gestionnaire de contexte.

Le processus global de gestionnaire de contexte est représenté dans **la Figure27**. Lorsque les capteurs de l'agent perçoit la situation actuelle de l'environnement, et à chaque fois où il y a un changement important dans la situation actuelle, le gestionnaire de contexte fait des recherches dans la base de contexte pour déterminer les différentes situations pertinentes à la

situation actuelle en se basant sur la compatibilité entre les caractéristiques de la situation actuelle et les caractéristiques de chaque situation dans la base de contexte. Ceci est fait par la fonction (**Sélection de contexte pertinent**).Dans le cas où l'agent ne trouve aucune situation pertinente, il envoie un message de demande d'aide à différentes sources de contextes (les autres agents du même environnement, l'utilisateur du système,…).

Une fois la liste des situations pertinentes a été obtenue, la fonction **Produire de contexte actuel** commence à traiter et combiner les différentes caractéristiques des situations pertinentes pour construire une situation globale qui représente mieux la situation actuelle de l'environnement. Aussi cette fonction permet à l'agent BDI de désigner un ensemble consistant des désirs. Cela permet à notre agent de se comporter d'une façon plus approprie à la situation actuelle de l'environnement.

4.4 Architecture proposée en détail :

Dans notre architecture on peut considérer un littéral comme une formule de prédicat L ou sa négation ┐L ou sa forte négation ~L.

Les croyances : Les croyances sont représentées par des littéraux :

- L : l'agent croit que L est vrai ;
- ┐L : l'agent ne croit pas que L est vrai ;
- ~L : l'agent croit que L est faux ;
- ┐(~L) : l'agent ne croit pas que L est faux ;

Les désirs : Ils existent deux types de désirs chez l'agent BDI

- Désir de test : Utilisé pour vérifier la validité du désir dans la base de croyance, et il est représenté par « ? » suivi d'un littéral.
- Désir à atteindre: Utilisé pour atteindre un but, et il est représenté par « ! » suivi d'un littéral.

Les intentions : Elles peuvent être :

- Une croyance à ajouter à la base des croyances ;
- Un sous but à atteindre ou tester ;
- Une action à exécuter sur l'environnement ;

Les plans :

Un plan dans notre architecture se compose de trois parties :

- Un évènement : Il est utilisé pour déclencher le plan dans le cas où cet évènement est généré ;
- Un contexte : Il est utilisé pour déterminer dans quel contexte on peut exécuter ce plan ;
- Corps de plan : est une séquence d'intention ;

Evènement : Contexte -> $\underbrace{\text{formule}_1, \text{formule}_2, \ldots\ldots, \text{formule}_n}$

Corps de plan

Algorithme BDI

B :=B_0 ; //initialiser la base des croyances

D :=D_0 ; //initialiser la base des désirs

C :=C_0 ; //initialiser la base de contextes

Pl :=Pl_0 ; //initialiser la bibliothèque des plans

Tan Que (vrai) faire

Perception(env, P, S_change) ;//env :environnement , P :ensemble des perception

Si (S_change) Alors

Sel_cont_pert(P, En_S) ;

Si (En_S≠ ∅) Alors

Envoyer (ag, traiter, P) ;

Attend(t) ;

Si aucune réponse alors Envoyer (utilisateur, traiter, P) ;

Reçu(En_S) ;

C :=C+En_S ; //mettre à jours la base de contextes

Fin_Si

Cont_act (En_S, B_a,D_a) ;// B_a : *les croyance de la situation actuelle ;D_a :les désirs de la situation actuelle*

Brf(B, B_a) ;//Brf est une fonction de révision de croyances

Drf(D, D_a) ; //Drf est une fonction de révision de désirs

Fin_Si

Si (D≠ ∅) Alors

Sel_D (D, d) ;

Sel_Pl_p (Pl, d, Pl_p) ;

Si (Pl_p≠ ∅) Alors

Envoyer (dest, comment, d) ;

Reçu (Pl_p) ;

Pl :=Pl+Pl_p ;
Fin_Si
Sel_Pl$_a$(Pl$_p$,Pl$_a$) ;
I :=I+instancie(Pl$_a$) ;
Si (D≠ ∅) Alors
Sel_Int (I, i) ;
Exécuter (i)
Fin_Si
Fin_Si
Fin_TQ
Fin

1. **Etape 0 : initialisation:** Dans cette étape l'agent BDI charge son état mental par les informations nécessaires pour sa mise en ouvre dans l'environnement :

   - Remplir la base de contextes par les situations possibles de l'agent dans son environnement ;
   - Initialiser la base des croyances ;
   - Initialiser la base des désirs ;
   - Créer la bibliothèque des plans ;

2. **Etape 1 : Perception de l'environnement :** L'architecture globale de l'agent doit avoir un composant capable de percevoir son environnement. Pour cela l'agent utilise ses différents capteurs et les messages des autres agents avec qui il partage le même environnement pour collecter toutes les informations nécessaires. Cela permet à l'agent de comprendre sa situation dans l'environnement exploré.

   Dans cette étape l'agent perçoit son environnement sous forme symbolique et retourne une liste des littéraux. Chaque littéral est un percept d'un élément particulier de contexte. Dans notre architecture, l'agent utilise deux listes de perception P et P1 pour détecter chaque changement dans la situation de l'agent, la liste P1 représente l'ensemble de perception actuelle, et la liste P représente l'ensemble des littéraux obtenus à partir de la dernière perception de l'environnement.

Procédure perception (env : environnement ; P : ensemble de perceptions ; S_change :booléen)
Var P_1 : ensemble de perceptions
Début
P_1 :=percept(env) ;

S_change :=faux ;

Pout (toute l_i dans P_1) faire // *Chaque littéral l dans **P1** n'existe pas dans **P** est //ajouté à P*

Début

Si (l_i n'existe pas dans P) alors

Début

Ajouter l_i dans p;

S_change :=vrai;

Fin

Fin

Pout (toute l_i dans P) faire // Chaque littéral l dans **B** n'existe pas dans **P** est supprimé sur **B**.

Début

Si (l_i n'existe pas dans P_1) alors

Début

Supprimer l_i sur p;

S_change :=vrai;

Fin

Fin

Fin

3. **Etape 2 : Sélectionner les situations pertinentes :** Une fois la liste P a été obtenue, et s'il n'ya pas un changement dans la situation actuelle de l'environnement on saute à l'étape 5 sinon la fonction **Sél_cont_pert (sélection de contexte pertinent)**. On effectue des recherches dans la base de contextes pour déterminer les différentes situations pertinentes à la liste des perceptions actuelles en se basant sur la compatibilité entre les caractéristiques de la liste P_2 et les caractéristiques de chaque situation dans la base de contextes.

Procédure Sél_cont_pert (P : ensemble de perceptions ; EnSC : ensemble de situations)

Début

EnSC ⟵ ∅ ;

Pour (toute SC dans la Base de contextes) faire

Début

Si (compatible(SC,P_2) Alors

EnSC ⟵ EnSC+SC ;

Fin

Fin

Dans le cas où il n'y a pas de situation pertinente, l'agent BDI envoie un message à d'autres agents du même environnement, ensuite l'agent met à jours la base de contextes par les nouvelles situations reçues. Mais dans le cas où il n'y a aucune réponse l'agent envoie un message à l'utilisateur pour l'aider.

4. **Etape 3 : Produire le contexte actuel :** Une fois l'ensemble des situations pertinentes a été obtenu, la fonction **cont_act** (**contexte actuel)** commence à traiter et combiner les différentes caractéristiques des situations pertinentes pour construire une situation globale qui représente mieux la situation actuelle de l'environnement.

   Procédure cont_act (EnSC : ensemble de situations; B_a : ensemble de croyances ; D_a : ensemble de Désirs)

   Début

   SA ⟵ combiner (EnSC) ;

   B_a ⟵ extraire_croyance(SA) ;

   D_a ⟵ extraire_désire(SA) ;

   Fin

5. **Etape 4 : Mise à jour de la base de croyances et la base de désirs :** Une fois la situation actuelle a été obtenue, la base de croyances et la base de désirs doivent être mises à jour mais sans oublier d'assurer la cohérence entre les différentes informations dans les deux bases. Pour la mise à jour de la base des croyances, elle est faite par la fonction BRF (Belief Revision Functions). Quant à celle de la base des désirs, elle est faite par la fonction DRF (Desire Revision Functions).
6. **Etape 5 : Déterminer l'objectif à atteindre :** Dans chaque cycle de raisonnement, l'agent BDI choisit un seul but à atteindre. Ce but est sélectionné par la fonction Sel_D. Cette fonction choisit toujours le premier élément de la base des désirs. Si la base de désirs est vide alors on passe à l'étape 9.
7. **Etape 6 : Sélectionner les plans pertinents :** Une fois un but est choisi, l'agent BDI sélectionne tous les plans pertinents à ce but.
8. **Etape 7 : Sélectionner les plans applicables :** Une fois que les plans pertinents sont sélectionnés, l'agent BDI détermine, parmi ces derniers ceux qui sont applicables dans le contexte.

9. **Etape 8 : Déterminer le plan à exécuter :** Dans cette étape l'agent BDI choisit un seul plan sur l'ensemble des plans applicables pour l'exécuter, et instancie ce plans dans la base des intentions.
10. **Etape 9 : Sélectionner l'intention à exécuter :** Après l'instanciation du plan choisi, l'agent sélectionne une intention et l'exécute.

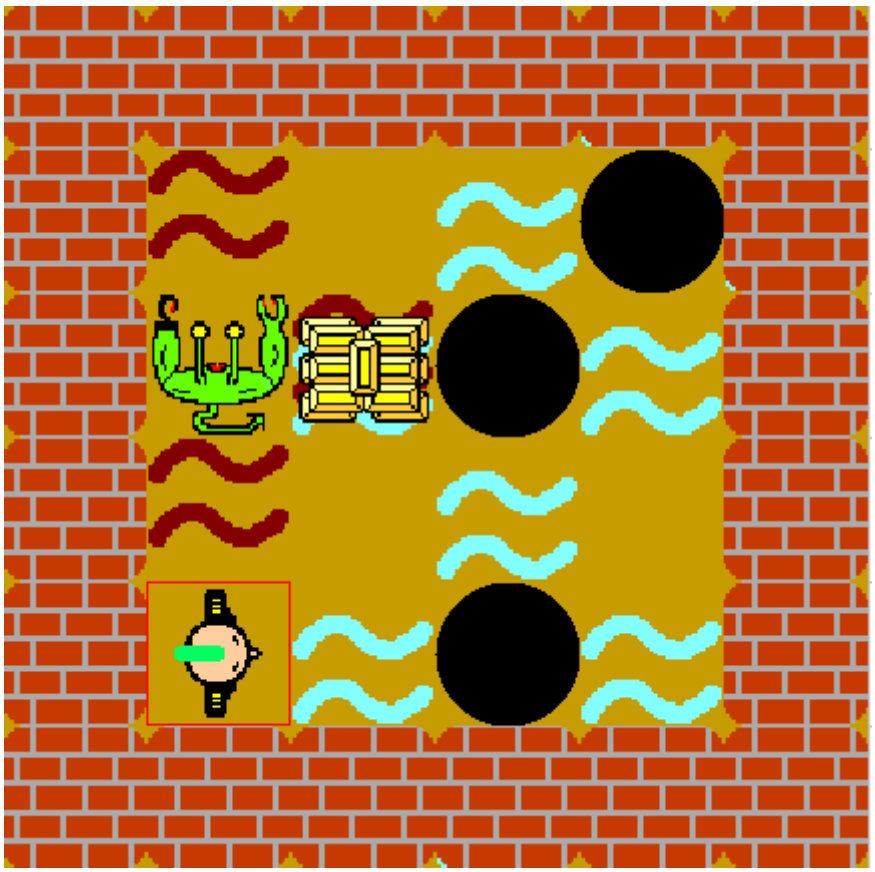
4.5 Conclusion

Dans ce travail, nous avons présenté un modèle d'agent qui prend en considération son contexte, et permet de le représenter explicitement. Cette architecture est basée sur l'architecture la plus populaire qui est l'architecture BDI et ses trois concepts (Croyance (Belief), Désir (Desire), Intention (Intention)). Nous proposons d'étendre le modèle d'agent BDI avec un quatrième concept qui permet d'améliorer son fonctionnement et le concept de contexte.

Chapitre 4 : Etude de cas

5.1 Etude de cas

Pour évaluer notre architecture proposée, nous avons essayé de résoudre le problème du Wumpus. Dans ce problème, il y a un joueur qui évolue sur une grille rectangulaire de taille variable. Il a comme but d'explorer la grille une cellule à la fois pour retrouver l'or. Le joueur peut se déplacer à gauche, à droite, en haut et en bas, mais pas en diagonale. La grille du jeu est composée de plusieurs cellules qui peuvent être libres, contenir un trou mortel, abriter le Wumpus ou contenir un lingot d'or. Il ne peut y avoir qu'un seul lingot d'or et un seul Wumpus par grille, mais il peut y avoir plusieurs trous mortels. Le joueur doit trouver le lingot d'or en évitant d'être dévoré par le Wumpus et de tomber dans un trou mortel. Afin d'éviter les trous mortels, ainsi que le Wumpus, le joueur se trouvant dans une pièce adjacente au Wumpus perçoit une odeur. Lorsque le joueur se trouve sur une pièce adjacente à un trou mortel, il perçoit un courant d'air. Le joueur a aussi la possibilité de tirer une flèche en direction du Wumpus pour le tuer.



5.1.1 Les formules utilisées

But	Croyance	action
!explorer(cellule(x,y))	visité(cellule(x,y))	tourne_G
!aug_RP(cellule(x,y)	adjacent(cellule(x,y),cellule(x+1,y),N)	tourne_D
!aug_RW(cellule(x,y))	est_cellule(cellule(x+1,y)))	prend_or
!démunie_RW	risque_w(cellule(x+1,y),z))	Shoot
!diminue_RP	risque_P(cellule(x+1,y),z))	
!selection	Sentir(brise)	
!aller(cellule(x,y))	sent(odeur)	
!tourne(d)	direction(d)	
!shoot(w,cellule(x,y))	risque(cellule(x,y),-5)	
!prend(or)	Do(direct)	
	position(cellule(x_1,y_1))	
	trace(cellule(x_1,y_1),e)	
	posséde(or)	

5.1.2 La base de contexte initiale :

A. Détermination des éléments du contexte :

possède(or)	Visité(cellule(x,y)	Sentir(brise)	sent(odeur)	Voir (brillant)
possède(or)	visité(cellule(x,y)	sentir(brise)	sent(odeur)	voir (brillant)
~possède(or)	┐visité(cellule(x,y)	~sentir(brise)	~sent(odeur)	~voir (brillant)

B. Enumération de toutes les situations possibles du contexte :

Le nombre total de situations possibles pour les éléments du contexte est $\boldsymbol{Max = \prod_{i=1..n} max_i}$ **=32 situation**

possède(or)	**visité(cellule(x,y))**	sentir(brise)	sent(odeur)	voir(brillant)
possède(or)	**visité(cellule(x,y))**	sentir(brise)	sent(odeur)	~voir(brillant)
possède(or)	**visité(cellule(x,y))**	sentir(brise)	~sent(odeur)	voir(brillant)
possède(or)	**visité(cellule(x,y))**	sentir(brise)	~sent(odeur)	~voir(brillant)
possède(or)	**visité(cellule(x,y))**	~sentir(brise)	sent(odeur)	voir(brillant)
possède(or)	**visité(cellule(x,y))**	~sentir(brise)	sent(odeur)	~voir(brillant)
possède(or)	**visité(cellule(x,y))**	~sentir(brise)	~sent(odeur)	voir(brillant)
possède(or)	**visité(cellule(x,y))**	~sentir(brise)	~sent(odeur)	~voir(brillant)
possède(or)	┐ **visité(cellule(x,y))**	sentir(brise)	sent(odeur)	voir(brillant)
possède(or)	┐ **visité(cellule(x,y))**	sentir(brise)	sent(odeur)	~voir(brillant)
possède(or)	┐ **visité(cellule(x,y))**	sentir(brise)	~sent(odeur)	voir(brillant)
possède(or)	┐ **visité(cellule(x,y))**	sentir(brise)	~sent(odeur)	~voir(brillant)
possède(or)	┐ **visité(cellule(x,y))**	~sentir(brise)	sent(odeur)	voir(brillant)
possède(or)	┐ **visité(cellule(x,y))**	~sentir(brise)	sent(odeur)	~voir(brillant)
possède(or)	┐ **visité(cellule(x,y))**	~sentir(brise)	~sent(odeur)	voir(brillant)
possède(or)	┐ **visité(cellule(x,y))**	~sentir(brise)	~sent(odeur)	~voir(brillant)
~possède(or)	**visité(cellule(x,y))**	sentir(brise)	sent(odeur)	voir(brillant)
~possède(or)	**visité(cellule(x,y))**	sentir(brise)	sent(odeur)	~voir(brillant)
~possède(or)	**visité(cellule(x,y))**	sentir(brise)	~sent(odeur)	voir(brillant)
~possède(or)	**visité(cellule(x,y))**	sentir(brise)	~sent(odeur)	~voir(brillant)
~possède(or)	**visité(cellule(x,y))**	~sentir(brise)	sent(odeur)	voir(brillant)
~possède(or)	**visité(cellule(x,y))**	~sentir(brise)	sent(odeur)	~voir(brillant)

~possède(or)	**visité(cellule(x,y))**	~sentir(brise)	~sent(odeur)	voir(brillant)
~possède(or)	**visité(cellule(x,y))**	~sentir(brise)	~sent(odeur)	~voir(brillant)
~possède(or)	┐ **visité(cellule(x,y))**	sentir(brise)	sent(odeur)	voir(brillant)
~possède(or)	┐ **visité(cellule(x,y))**	sentir(brise)	sent(odeur)	~voir(brillant)
~possède(or)	┐ **visité(cellule(x,y))**	sentir(brise)	~sent(odeur)	voir(brillant)
~possède(or)	┐ **visité(cellule(x,y))**	sentir(brise)	~sent(odeur)	~voir(brillant)
~possède(or)	┐ **visité(cellule(x,y))**	~sentir(brise)	sent(odeur)	voir(brillant)
~possède(or)	┐ **visité(cellule(x,y))**	~sentir(brise)	sent(odeur)	~voir(brillant)
~possède(or)	┐ **visité(cellule(x,y))**	~sentir(brise)	~sent(odeur)	voir(brillant)
~possède(or)	┐ **visité(cellule(x,y))**	~sentir(brise)	~sent(odeur)	~voir(brillant)

Eliminer toute les situations en gris parce qu'on a la règle contradictoire suivante : possède(or) et voir(brillant) ;

C. Association

Situation	Les éléments de contexte					but
1	possède(or)	?	?	?	~voir(brillant)	!aller(cellule(0,0))
2	~possède(or)	?	?	?	voir(brillant)	!prend(or)
3	~possède(or)	visité(cellule(x,y))	?	?	~voir(brillant)	Rien
4	~possède(or)	┐ visité(cellule(x,y))	sentir(brise)	sent(odeur)	~voir(brillant)	!explorer(cellule(x,y)) !aug_RP(cellule(x,y)) !aug_RW(cellule(x,y)) !selection

5	~possède(or)	┐visité(cellule(x,y))	sentir(brise)	~sent(odeur)	~voir(brillant)	!explorer(cellule(x,y)) !aug_RP(cellule(x,y)) !démunie_RW(cellule(x,y)) !selection
6	~possède(or)	┐visité(cellule(x,y))	~sentir(brise)	sent(odeur)	~voir(brillant)	!explorer(cellule(x,y)) !démunie_RP(cellule(x,y)) !aug_RW(cellule(x,y)) !selection
7	~possède(or)	┐visité(cellule(x,y))	~sentir(brise)	~sent(odeur)	~voir(brillant)	!explorer(cellule(x,y)) !démunie_RP(cellule(x,y)) !démunie_RW(cellule(x,y)) !selection

5.1.3 Les croyances initiales :

Direction(E) ; position (cellule(0,0)) ;etape(0) ;~posséde(or) ;

~Est_cellule(cellule(-1,y) ; ~Est_cellule(cellule(x,-1) ;

~Est_cellule(cellule(max_x,y) ; ~Est_cellule(cellule(x,max_y) ;

- **Règle :** Risque(z1+z2) :risque_ P(cellule(x1,y1),z1) ;risque_W(cellule(x,y),z2)

5.1.4 Les plans initiaux :

Plan : P1

Evènement : + !explorer(cellule(x,y))

Contexte : ~visité(cellule(x,y))

Corps : +adjacent(cellule(x,y),cellule(x+1,y),N) ;

Si est_cellule(cellule(x+1,y))) alors

Si (?¬ risque_w(cellule(x+1,y),z)) alors +risque_w(cellule(x+1,y),0) ;

Si (?¬ risque_P(cellule(x+1,y),z)) alors +risque_P(cellule(x+1,y),0) ;

Fin_si ;

+adjacent(cellule(x,y),cellule(x-1,y),S) ;

Si est_cellule(cellule(x-1,y))) alors

Si (?¬ risque_w(cellule(x-1,y),z)) alors +risque_w(cellule(x-1,y),0) ;

Si (?¬ risque_P(cellule(x-1,y),z)) alors +risque_P(cellule(x-1,y),0) ;

Fin_si ;

+adjacent(cellule(x,y),cellule(x,y-1),O) ;

Si est_cellule(cellule(x,y-1))) alors

Si (?¬ risque_w(cellule(x,y-1),z)) alors +risque_w(cellule(x,y-1),0) ;

Si (?¬ risque_P(cellule(x,y-1),z)) alors +risque_P(cellule(x,y-1),0) ;

Fin_si ;

+adjacent(cellule(x,y),cellule(x,y+1),E) ;

Si est_cellule(cellule(x,y+1))) alors

Si (?¬ risque_w(cellule(x,y+1),z)) alors +risque_w(cellule(x,y+1),0) ;

Si (?¬ risque_P(cellule(x,y+1),z)) alors +risque_P(cellule(x,y+1),0) ;

Fin_si ;

Dans ce plan, l'agent détermine toutes les cellules adjacentes à sa position et initialise la valeur de risque de chaque cellule.

Plan : P2

Evènement : +!aug_RP(cellule(x,y)

Contexte : ~visité(cellule(x,y)) et Sentir(brise))

Corps :

si (adjacent(cellule(x,y),cellule(x1,y1),E) et est_cellule(cellule(x1,y1)))alors

?risque_p(cellule(x1,y1),z)

Si (z≥0) alors ±risque_p(cellule(x1,y1),z+1) ;

Fin si

si (adjacent(cellule(x,y),cellule(x2,y2),N) et est_cellule(cellule(x2,y2)))alors

?risque_P(cellule(x2,y2),z)

Si (z≥0) alors ±risque_p(cellule(x2,y2),z+1) ;

Fin si

si (adjacent(cellule(x,y),cellule(x3,y3),O) et est_cellule(cellule(x3,y3)))alors

?risque_P(cellule(x3,y3),z)

Si (z≥0) alors ±risque_p(cellule(x3,y3),z+1) ;

Fin si

si (adjacent(cellule(x,y),cellule(x4,y4),S) et est_cellule(cellule(x4,y4)))alors

?risque_P(cellule(x4,y4),z)

Si (z≥0) alors ±risque_p(cellule(x4,y4),z+1) ;

Fin si

Dans ce plan, l'agent augmente le risque de trouver un trou dans les cellules adjacentes.

Plan : P3

Evènement : +!aug_RW(cellule(x,y))

Contexte : ~visité(cellule(x,y)) et Sent(odeur)

Corps :

si (adjacent(cellule(x,y),cellule(x1,y1),E) et est_cellule(cellule(x1,y1)))alors

?risque_P(cellule(x1,y1),z)

Si (z≥0) alors

±risque_p(cellule(x1,y1),z+1) ;

Si (z=1) alors +!shoot(wumpus,cellule(x1,y1))

Fin si

Fin si

si (adjacent(cellule(x,y),cellule(x2,y2),N) et est_cellule(cellule(x2,y2)))alors

?risque_W(cellule(x2,y2),z)

Si (z≥0) alors

±risque_p(cellule(x2,y2),z+1) ;

Si (z=1) alors +!shoot(wumpus,cellule(x2,y2))

Fin si

Fin si

si (adjacent(cellule(x,y),cellule(x3,y3),O) et est_cellule(cellule(x3,y3)))alors

?risque_P(cellule(x3,y3),z)

Si (z≥0) alors

±risque_p(cellule(x3,y3),z+1) ;

Si (z=1) alors +!shoot(wumpus,cellule(x3,y3))

Fin si

Fin si

si (adjacent(cellule(x,y),cellule(x4,y4),S) et est_cellule(cellule(x4,y4)))alors

?risque_P(cellule(x4,y4),z)

Si (z≥0) alors

±risque_p(cellule(x4,y4),z+1) ;

Si (z=1) alors +!shoot(wumpus,cellule(x4,y4))

Fin si

Fin si

Dans ce plan, l'agent augmente le risque de trouver un wumpus dans les cellules adjacentes.

Plan : P4

Evènement : +!démunie_RW

Contexte : ~vésité(cellule(x,y)) et ~sent(odeur)

Corps : ?adjacent(cellule(x,y),cellule(x1,y1),E)

±risque_W(cellule(x1,y1),-5) ;

?adjacent(cellule(x,y),cellule(x2,y2),N) ;

±risque_W(cellule(x2,y2) ,-5) ;

?adjacent(cellule(x,y),cellule(x3,y3),O) ;

±risque_W(cellule(x3,y3) ,-5) ;

?adjacent(cellule(x,y),cellule(x4,y4),S) ;

±risque_W(cellule(x4,y4) ,-5) ;

Dans ce plan l'agent diminue le risque de trouver un wumpus dans les cellules adjacent.

Plan :P5

Evènement : +!diminue_RP

Contexte : ~vésité(cellule(x,y)) et ~ Sentir(brise))

Corps : ?adjacent(cellule(x,y),cellule(x1,y1),E) ;

Si est_cellule(cellule(x1,y1)) alors ±risque_P(cellule(x1,y1),-5) ;

?adjacent(cellule(x,y),cellule(x2,y2),N) ;

Si est_cellule(cellule(x1,y1)) alors ±risque_P(cellule(x2,y2) ,-5) ;

?adjacent(cellule(x,y),cellule(x3,y3),O) ;

Si est_cellule(cellule(x1,y1)) alors ±risque_P(cellule(x3,y3) ,-5) ;

?adjacent(cellule(x,y),cellule(x4,y4),S) ;

Si est_cellule(cellule(x1,y1)) alors ±risque_P(cellule(x4,y4) ,-5) ;

Dans ce plan, l'agent diminue le risque de trouver un trou dans les cellules adjacentes.

Plan : P6

Evènement :+ !selection

Contexte :

Corps : ?pos(cellule(x1,y1)) ;

+visité(cellule(x1,y1)) ;

+ risque_w(cellule(x1,y1),-5) ;

+ risque_p(cellule(x1,y1),-5) ;

?etape(e) ;

±etape(e+1) ;

+trace(cellule(x1,y1),e+1) ;

Si risque(cellule(x,y),-5) et ┐ visité(cellule(x,y)) alors + !aller(cellule(x,y)) ;

Sinon Si risque(cellule(x,y),0) et ┐ visité(cellule(x,y)) alors + !aller(cellule(x,y)) ;

Sinon Si risque(cellule(x,y),1) et ┐ visité(cellule(x,y)) alors + !aller(cellule(x,y)) ;

Sinon Si risque(cellule(x,y),2) et ┐ visité(cellule(x,y)) alors + !aller(cellule(x,y)) ;

Sinon Si risque(cellule(x,y),3) et ┐ visité(cellule(x,y)) alors + !aller(cellule(x,y)) ;

Sinon Si risque(cellule(x,y),4) et ┐ visité(cellule(x,y)) alors + !aller(cellule(x,y)) ;

Ce plan permet à l'agent de sélectionner la cellule de moins de risque.

Plan : P7

Evènement : +!aller(cellule(x,y)) ;

Contexte : (position(cellule(x_1,y_1)) et adjacent(cellule(x_1,y_1),cellule(x,y),d)) ;

Corps: +!tourne(d) ;

Do(direct) .

Plan : P8

Evènement : +!aller(cellule(x,y)) ;

Contexte : (position(cellule(x_1,y_1)) et ┐ (adjacent(cellule(x_1,y_1),cellule(x,y),d))) ;

Corps: ?trace(cellule(x_1,y_1),e) ;

?trace(cellule(x_2,y_2),e-1) ;

+!aller(cellule(x_2,y_2));

+!aller(cellule(x,y));

Les deux plan P7 et P8 permettent à l'agent d'aller à la cellule qu'il désigne.

Plan : P9

Evènement : +!tourne(d)

Contexte : direction(d).

Plan : P10

Evènement : +!tourne(d)

Contexte : ┐ direction(d)

Corps : ?direction(d_1)

Case d_1 of

E : case d of

N: tourne_G;±direction(N) ;

O: tourne_G; tourne_G; ±direction(O) ;

S: tourne_D; ±direction(S) ;

N : case d_1 of

E: tourne_D; ±direction(E) ;

O: tourne_G; ±direction(O) ;

S: tourne_G; tourne_G; ±direction(S) ;

O : case d_1 of

E: tourne_G; tourne_G; ±direction(E) ;

N: tourne_D; ±direction(N) ;

S: tourne_G; ±direction(S) ;

S : case d_1 of

E: tourne_G; ±direction(E) ;

N: tourne_G; tourne_G; ±direction(N) ;

O: tourne_D; ±direction(O) ;

Fin.

Ce plan permet à l'agent de tourner vers l'est, l'ouest,le nord ou le sud.

Plan P11

Evènement : + !prend(or) ;

Contexte : Voir (brillant)

Corps : prend_or ; +possède(or) ;

Dans ce plan, l'agent porte l'or trouvé dans la cellule .

Plan : P12

Evènement :+ !shoot(w,cellule(x,y))

Contexte :risque_W(cellule(x,y),2)

Corps : ?pos(cellule(x1,y2)) ;

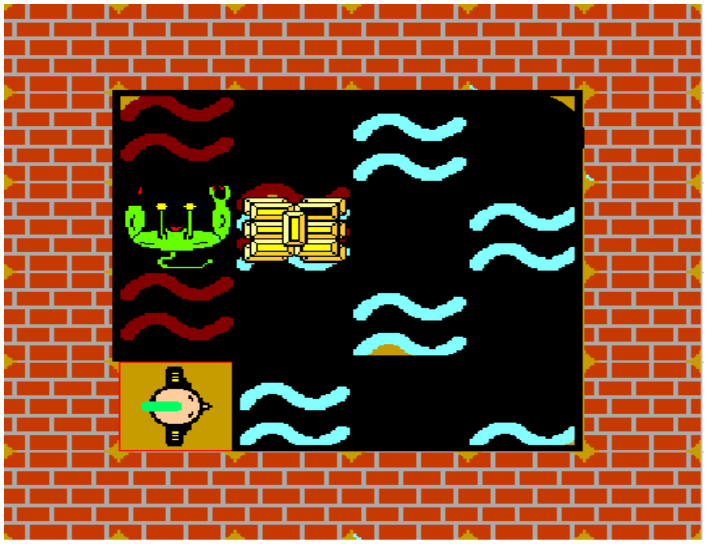
?adjacent(cellule(x1,y1),cellule(x,y),d) ;

Tourne(d) ;

Shoot .

5.1.5 Le scénario :

Dans cette étape, l'agent charge son état mental par les informations de sa situation dans l'environnement. Dans ce cas, l'agent croit que sa position est dans la cellule (0,0). Il ne possède pas l'or, et ne trouve aucune brise, aucune odeur, aucune brillance.



1er cycle de raisonnement	
Situation	**Sit 7**
Croyance	*Direction(E) ; position (cellule(0,0)) ;etape(0) ;~posséde(or) ;* *~Est_cellule(cellule(-1,y)) ; ~Est_cellule(cellule(x,-1)) ;* *~Est_cellule(cellule(max$_x$,y)) ; ~Est_cellule(cellule(x,max$_y$)) ;* *~possède(or) ; ┐visité(cellule(0,0)) ; ~sentir(brise) ; ~sent(odeur) ; ~voir(brillant)*
Evènement	**!explorer(cellule(x,y))** *;!démunie_RP(cellule(x,y)) ;!démunie_RW(cellule(x,y)) ;!selection*
Plan	**P1**
Intention	**+adjacent(cellule(0,0),cellule(1,0),N)** ; *+risque_w(cellule(1,0),0) ;* *+adjacent(cellule(0,0),cellule(0,1),N) ; +risque_w(cellule(0,1),0) ;*

2ème cycle de raisonnement	
Situation	
Croyance	*Direction(E) ; position (cellule(0,0)) ;etape(0) ; ~posséde(or) ;* *~Est_cellule(cellule(-1,y)) ; ~Est_cellule(cellule(x,-1)) ;* *~Est_cellule(cellule(max$_x$,y)) ; ~Est_cellule(cellule(x,max$_y$)) ;* *~possède(or) ; ┐visité(cellule(0,0)) ; ~sentir(brise) ; ~sent(odeur) ; ~voir(brillant)* **adjacent(cellule(0,0),cellule(1,0),N) ;**
Evènement	**!démunie_RP(cellule(x,y))** *;!démunie_RW(cellule(x,y)) ;!selection*
Plan	**P5**
Intention	**+risque_w(cellule(1,0),0)** *;+adjacent(cellule(0,0),cellule(0,1),N) ; +risque_w(cellule(0,1),0) ;*
	?adjacent(cellule(x,y),cellule(x1,y1),E) ; ±risque_P(cellule(x1,y1),-5) ; ….

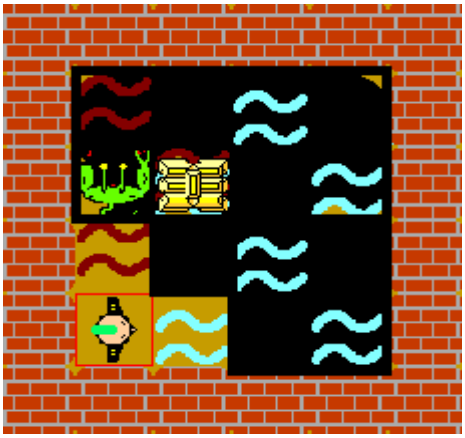
.

.

.

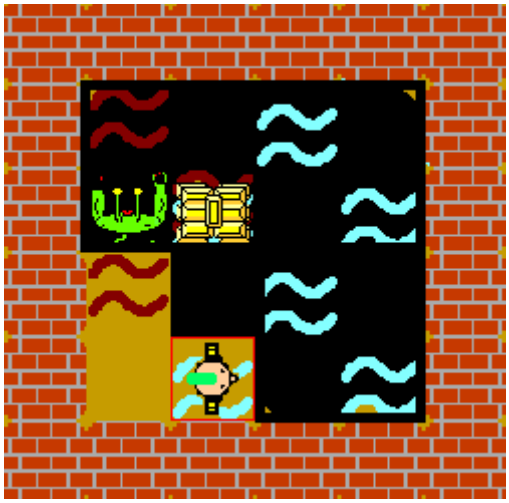
.

$(N_2)^{ème}$ cycle de raisonnement	
Situation	
Croyance	*Direction(E) ; position (cellule(0,0)); ~posséde(or) ; ~Est_cellule(cellule(-1,y)) ; ~Est_cellule(cellule(x,-1)) ; ~Est_cellule(cellule(max_x,y)) ; ~Est_cellule(cellule(x,max_y)) ; ~possède(or) ; ┐visité(cellule(0,0)) ; ~sentir(brise) ; ~sent(odeur) ; ~voir(brillant) ; adjacent(cellule(0,0),cellule(1,0),N) ; risque_w(cellule(1,0),-5) ; risque_P(cellule(1,0),-5) ; adjacent(cellule(0,0),cellule(0,1),E) ; risque_w(cellule(0,1),-5) ;risque_P(cellule(0,1),-5) ; risque (cellule(0,1),-10) ; risque (cellule(1,0),-10)*
Evènement	**+ !selection**
Plan	**P6**
Intention	**?pos(cellule(x1,y1)) ;** *+visité(cellule(x1,y1)) ; ?etape(e) ; ±etape(e+1) ; +trace(cellule(x1,y1),e+1) ;…*
$(N_3)^{ème}$ cycle de raisonnement	
Situation	
Croyance	*Direction(E) ; position (cellule(0,0)) ; ~posséde(or) ; ~Est_cellule(cellule(-1,y)) ; ~Est_cellule(cellule(x,-1)) ; ~Est_cellule(cellule(max_x,y)) ; ~Est_cellule(cellule(x,max_y)) ; ~possède(or) ; ┐visité(cellule(0,0)) ; ~sentir(brise) ; ~sent(odeur) ; ~voir(brillant) ; adjacent(cellule(0,0),cellule(1,0),N) ; risque_w(cellule(1,0),-5) ; risque_P(cellule(1,0),-5) ; adjacent(cellule(0,0),cellule(0,1),E) ; risque_w(cellule(0,1),-5) ;risque_P(cellule(0,1),-5) ; visité(cellule(0,0)) ; etape(1) ; trace(cellule(0,0), 1) ; risque (cellule(0,1),-10) ; risque (cellule(1,0),-10)*
Evènement	**!aller(cellule(0,1))**
Plan	**P7, P8**
Intention	**+!tourne(d)** *; Do(direct)* **.**



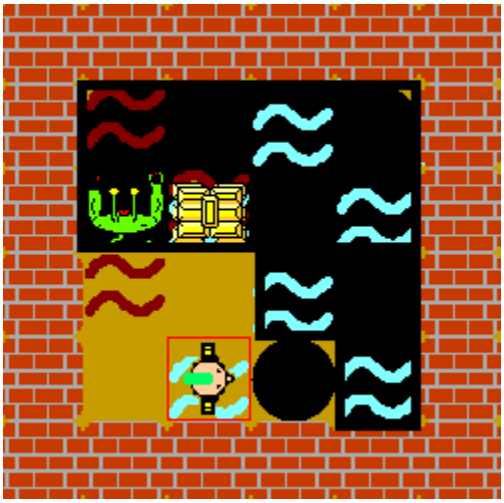
L'agent, après avoir effectué certains cycles de raisonnement, il déduit le risque de trouver un Wumpus ou un trou dans les cellules adjacentes et choisit la cellule la plus sécurisée.

(N_4)ème cycle de raisonnement	
Situation	**Sit 5**
Croyance	*Direction(E) ;* **position (cellule(0,1))** *; ~posséde(or) ; ~Est_cellule(cellule(-1,y)) ;* *~Est_cellule(cellule(x,-1)) ; ~Est_cellule(cellule(max$_x$,y)) ; ~Est_cellule(cellule(x,max$_y$)) ;* **~possède(or) ; ┐visité(cellule(0,0)) ; sentir(brise) ; ~sent(odeur) ; ~voir(brillant) ;** *adjacent(cellule(0,0),cellule(1,0),N) ; risque_w(cellule(1,0),-5) ; risque_P(cellule(1,0),-5) ;* *adjacent(cellule(0,0),cellule(0,1),E) ; risque_w(cellule(0,1),-5) ;risque_P(cellule(0,1),-5) ;* *visité(cellule(0,0)) ; etape(1) ; trace(cellule(0,0), 1) ;* *risque (cellule(0,1),-10) ; risque (cellule(1,0),-10)*
Evènement	**!explorer(cellule(x,y))** *;!aug_RP(cellule(x,y)) ;!démunie_RW(cellule(x,y)) ;!selection*
Plan	**P1**
Intention	**….**



Dans cette étape, l'agent va à la cellule choisie.

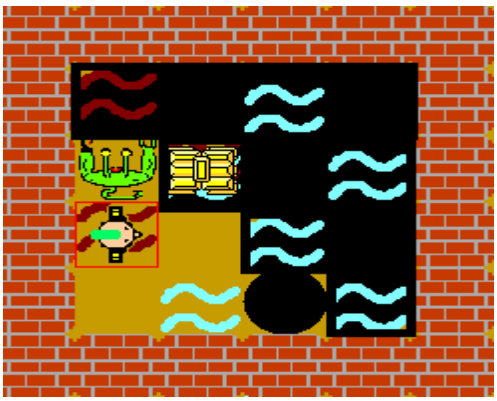
(N_5)^{ème} cycle de raisonnement	
Situation	
Croyance	*Direction(E) ; position (cellule(0,1)) ; ~posséde(or) ; ~Est_cellule(cellule(-1,y)) ;* *~Est_cellule(cellule(x,-1)) ; ~Est_cellule(cellule(max$_x$,y)) ; ~Est_cellule(cellule(x,max$_y$)) ;* *~possède(or) ; ┐visité(cellule(0,1)) ; sentir(brise) ; ~sent(odeur) ; ~voir(brillant) ;* *adjacent(cellule(0,0),cellule(1,0),N) ; risque_w(cellule(1,0),-5) ; risque_P(cellule(1,0),-5) ;* *adjacent(cellule(0,0),cellule(0,1),E) ; risque_w(cellule(0,1),-5) ;risque_P(cellule(0,1),-5) ;* *visité(cellule(0,0)) ; etape(1) ; trace(cellule(0,0), 1) ; risque_w(cellule(0,0),-5) ;* *risque_P(cellule(0,0),-5) ;* *risque (cellule(0,1),-10) ; risque (cellule(1,0),-10) ; risque (cellule(0,0),-10)* *adjacent(cellule(0,1),cellule(1,1),N)) ; risque_w(cellule(1,1),-5) ; risque_P(cellule(1,0),1) ;* *adjacent(cellule(0,1),cellule(0,2),E)) ; risque_w(cellule(0,2),-5) ;risque_P(cellule(0,1),1) ;* *adjacent(cellule(0,1),cellule(0,0),O) ;* *visité(cellule(0,1)) ; etape(2) ; trace(cellule(0,1), 2) ;* *risque (cellule(1,1),-4) ; risque (cellule(0,2),-4) ;*
Evènement	
Plan	
Intention	**….**



Dans cette étape, l'agent trouve une brise dans la cellule. Pour cela, l'agent augmente le risque de trouver un trou dans les cellules adjacentes. Ensuite il choisit la cellule la plus sécurisée.

(N_6)ème cycle de raisonnement	
Situation	**Sit 6**
Croyance	*Direction(E) ;* **position (cellule(1,0))***; ~posséde(or) ; ~Est_cellule(cellule(-1,y)) ;* ~Est_cellule(cellule(x,-1)) ; ~Est_cellule(cellule(max_x,y)) ; ~Est_cellule(cellule(x,max_y)) ; **~possède(or) ; ┐visité(cellule(0,1)) ; ~sentir(brise) ; sent(odeur) ; ~voir(brillant) ;** *adjacent(cellule(0,0),cellule(1,0),N) ; risque_w(cellule(1,0),-5) ; risque_P(cellule(1,0),-5) ;* *adjacent(cellule(0,0),cellule(0,1),E) ; risque_w(cellule(0,1),-5) ;risque_P(cellule(0,1),-5) ;* *visité(cellule(0,0)) ; trace(cellule(0,0), 1) ; risque_w(cellule(0,0),-5) ;* *risque_P(cellule(0,0),-5) ;* *risque (cellule(0,1),-10) ; risque (cellule(1,0),-10) ; risque (cellule(0,0),-10)* *adjacent(cellule(1,0),cellule(1,1),N)) ; risque_w(cellule(1,1),-5) ;* **risque_P(cellule(1,1),-5) ;** *adjacent(cellule(1,0),cellule(0,2),E)) ; risque_w(cellule(0,2),-5) ;risque_P(cellule(0,2),1) ;* *adjacent(cellule(1,0),cellule(0,0),O) ;* *visité(cellule(1,0)) ; trace(cellule(1,0), 2) ;* **risque (cellule(1,1),-10)** *; risque (cellule(0,2),-4) ;* *adjacent(cellule(1,0),cellule(1,1),E)) ; risque_w(cellule(1,1),1) ;* *adjacent(cellule(1,0),cellule(0,2),N)) ; risque_w(cellule(0,2),-5) ;risque_P(cellule(0,2),1) ;* *adjacent(cellule(1,0),cellule(0,0),S)) ;* *visité(cellule(1,0)) ; etape(3) ; trace(cellule(1,0), 3) ;*

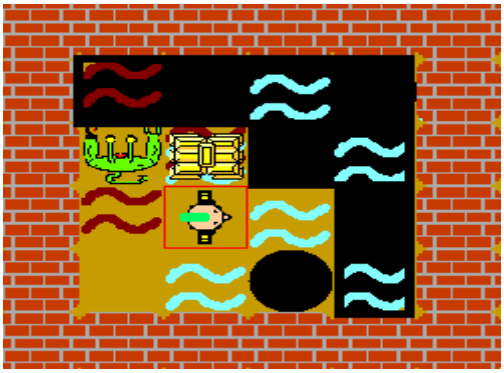
	risque (cellule(0,2),-4)
Evènement	!explorer(cellule(x,y)) ;!démunie_RP(cellule(x,y)) ;!aug_RW(cellule(x,y)) ;!selection **!aller(cellule(1,1))**
Plan	**P7 ,P8**
Intention	**….**



Dans cette étape, l'agent va à la cellule choisie.

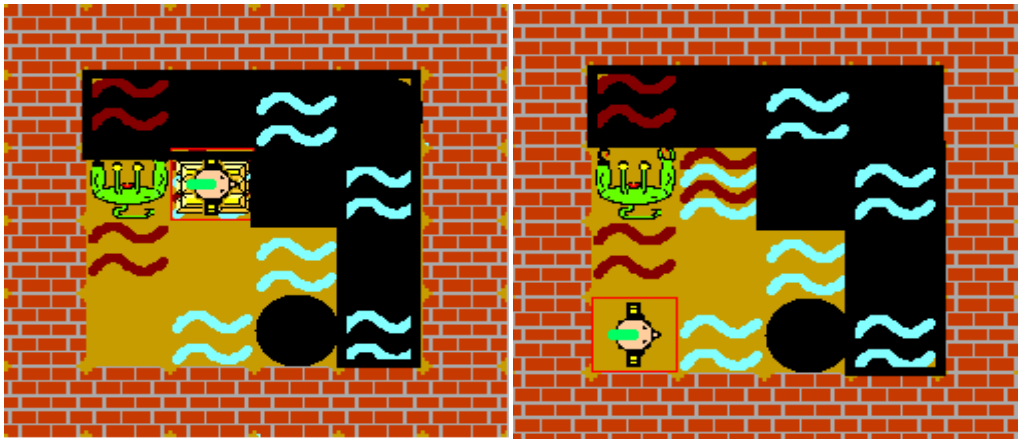
(N_7)^{ème} cycle de raisonnement	
Situation	**Sit 7**
Croyance	*Direction(E) ;* **position (cellule(1,1))** *; ~posséde(or) ; ~Est_cellule(cellule(-1,y)) ;* *~Est_cellule(cellule(x,-1)) ; ~Est_cellule(cellule(max_x,y)) ; ~Est_cellule(cellule(x,max_y)) ;* **~possède(or) ; ┐visité(cellule(0,1)) ; ~sentir(brise) ; ~sent(odeur) ; ~voir(brillant) ;** *adjacent(cellule(0,0),cellule(1,0),N) ; risque_w(cellule(1,0),-5) ; risque_P(cellule(1,0),-5) ;* *adjacent(cellule(0,0),cellule(0,1),E) ; risque_w(cellule(0,1),-5) ;risque_P(cellule(0,1),-5) ;* *visité(cellule(0,0)) ; trace(cellule(0,0), 1) ; risque_w(cellule(0,0),-5) ;* *risque_P(cellule(0,0),-5) ;* *risque (cellule(0,1),-10) ; risque (cellule(1,0),-10) ; risque (cellule(0,0),-10)* *adjacent(cellule(1,0),cellule(1,1),N)) ; risque_w(cellule(1,1),-5) ;* **risque_P(cellule(1,1),-5) ;** *adjacent(cellule(1,0),cellule(0,2),E)) ; risque_w(cellule(0,2),-5) ;risque_P(cellule(0,2),1) ;* *adjacent(cellule(1,0),cellule(0,0),O) ;* *visité(cellule(1,0)) ; trace(cellule(1,0), 2) ;* **risque (cellule(1,1),-10)** *; risque (cellule(0,2),-4) ;*

	adjacent(cellule(1,0),cellule(1,1),E)) ; risque_w(cellule(1,1),1) ; *adjacent(cellule(1,0),cellule(0,2),N)) ; risque_w(cellule(0,2),-5) ;risque_P(cellule(0,2),1) ;* *adjacent(cellule(1,0),cellule(0,0),S)) ;* *visité(cellule(1,0)) ; etape(3) ; trace(cellule(1,0), 3) ;* *risque (cellule(0,2),-4) visité(cellule(1,1)) ; etape(4) ; trace(cellule(1,1), 4) ;* *risque (cellule(1,2),-10) ; risque (cellule(2,1),-10)* **adjacent(cellule(1,1),cellule(1,2),E)) ; risque_w(cellule(1,2),-5) ;risque_P(cellule(1,2),-5) ;** **adjacent(cellule(1,1),cellule(2,1),N)) ; risque_w(cellule(2,1),-5) ;risque_P(cellule(2,1),-5) ;** **adjacent(cellule(1,1),cellule(1,0),O)) ;** **adjacent(cellule(1,1),cellule(0,1),S)) ;**
Evènement	*!explorer(cellule(x,y)) ;!démunie_RP(cellule(x,y)) ;!démunie_RW(cellule(x,y)) ;!selection* **!aller(cellule(2,1))**
Plan	**P7 ,P8**
Intention	**….**



(N_8)^{ème} cycle de raisonnement	
Situation	**Sit 2**
Croyance	*Direction(E) ;* **position (cellule(2,1))** *; ~posséde(or) ; ~Est_cellule(cellule(-1,y)) ;* *~Est_cellule(cellule(x,-1)) ; ~Est_cellule(cellule(max$_x$,y)) ; ~Est_cellule(cellule(x,max$_y$)) ;* **~possède(or) ; ┐visité(cellule(0,1)) ; sentir(brise) ; sent(odeur) ;voir(brillant) ;** *adjacent(cellule(0,0),cellule(1,0),N) ; risque_w(cellule(1,0),-5) ; risque_P(cellule(1,0),-5) ;* *adjacent(cellule(0,0),cellule(0,1),E) ; risque_w(cellule(0,1),-5) ;risque_P(cellule(0,1),-5) ;*

	visité(cellule(0,0)) ; trace(cellule(0,0), 1) ; risque_w(cellule(0,0),-5) ; *risque_P(cellule(0,0),-5) ;* *risque (cellule(0,1),-10) ; risque (cellule(1,0),-10) ; risque (cellule(0,0),-10)* *adjacent(cellule(1,0),cellule(1,1),N)) ; risque_w(cellule(1,1),-5) ; risque_P(cellule(1,1),-5) ;* *adjacent(cellule(1,0),cellule(0,2),E)) ; risque_w(cellule(0,2),-5) ;risque_P(cellule(0,2),1) ;* *adjacent(cellule(1,0),cellule(0,0),O) ;* *visité(cellule(1,0)) ; trace(cellule(1,0), 2) ;* *risque (cellule(1,1),-10) ; risque (cellule(0,2),-4) ;* *adjacent(cellule(1,0),cellule(1,1),E)) ; risque_w(cellule(1,1),1) ;* *adjacent(cellule(1,0),cellule(0,2),N)) ; risque_w(cellule(0,2),-5) ;risque_P(cellule(0,2),1) ;* *adjacent(cellule(1,0),cellule(0,0),S)) ;* *visité(cellule(1,0)) ; etape(3) ; trace(cellule(1,0), 3) ;* *risque (cellule(0,2),-4)* *visité(cellule(1,1)) ; etape(4) ; trace(cellule(1,1), 4) ;* *risque (cellule(1,2),-10) ; risque (cellule(2,1),-10)* **adjacent(cellule(1,1),cellule(1,2),E)); risque_w(cellule(1,2),-5);risque_P(cellule(1,2),-5);** **adjacent(cellule(1,1),cellule(2,1),N)) ; risque_w(cellule(2,1),-5) ;risque_P(cellule(2,1),-5) ;** **adjacent(cellule(1,1),cellule(1,0),O)) ;** **adjacent(cellule(1,1),cellule(0,1),S)) ;**
Evènement	**!prend(or)**
Plan	**P11**
Intention	**vider la base de désire** *; prend_or ; +possède(or) ;*



Dans cette étape, l'agent trouve une brillance dans la cellule. Pour cela l'agent prend l'or et change son but pour sortir de la grille.

Conclusion Générale

Dans ce mémoire, nous avons commencé par une étude détaillée sur les agents et les systèmes multi-agents. Cette étude nous a ramené à déduire que les systèmes multi-agents offrent, aujourd'hui, une alternative intéressante pour la conception, la mise en œuvre ou la simulation et la compréhension de systèmes coopératifs distribués et ouverts. Ensuite, Nous avons mis l'accent sur les agents BDI et les différents travaux existants sur cette architecture. Et nous avons également constaté que tous ces travaux se fondent sur les trois attitudes mentales suivantes : Croyance, Désir et Intention (**B**elief, **D**esire and **i**ntention). Ensuite, nous avons présenté quelques concepts du contexte et des systèmes sensibles au contexte, et nous avons donné quelques systèmes, tenant compte du contexte comme (Orca, Context-Based Reasoning…etc.).

Dans ce mémoire, nos travaux sont concentrés sur la proposition d'une extension de l'architecture BDI qui prend en considération le contexte. Dans ce contexte, nous avons fixé des propriétés fondamentales, sur lesquels, on a basé notre architecture proposée. Ces propriétés sont :

- Comment représenter le contexte de l'agent?
- Comment mettre à jour la structure de données de contexte?
- Comment le contexte affecte dans le processus de prise de décision?

Pour la représentation du contexte, notre architecture représente le contexte par un ensemble de situations possibles de l'agent dans son environnement. Chaque situation regroupe deux catégories d'informations : Dans la première catégorie, l'agent représente les différents caractéristiques de sa situation dans l'environnement (eg : localisation= à la maison, temps= 8 :00 Am, température=35° …etc.). Et dans la deuxième catégorie, l'agent représente les différents buts à atteindre dans chaque situation (eg : Réduire la température à 25°). Cette représentation permet à l'agent d'être plus conscient de sa situation dans l'environnement et comment se comporter dans cette situation.

Et pour la mise à jour de la base de contexte, l'utilisateur initialise la base de contexte de chaque agent par les situations possibles qu'il puisse rencontrer dans son environnement. Et

pour cela nous avons adapté l'approche de (Miraoui M., Tadj C. et Amar C.B. 2008 [89]) sur notre architecture pour construire la base de contexte. Cette approche basée sur la détermination des éléments du contexte, l'énumération de toutes ses situations possibles et associer à chaque situation les buts appropriés dans ce contexte. Dans la mise en œuvre du système, le modèle que nous avons proposé, détermine deux cas possibles : Dans le premier cas, l'agent se trouve dans une situation qui existe déjà dans la base du contexte. Dans ce cas aucune mise à jour ne sera effectuée sur la base du contexte. Dans le deuxième cas, l'agent se trouve dans une situation qui n'existe pas dans la base du contexte. Dans ce cas l'agent demande l'intervention des autres agents du même environnement par l'envoi d'un message contenant toutes les perceptions de l'agent (x). Après avoir reçu le message, l'agent (y) cherche dans sa base de contexte toutes les situations pertinentes et les envoyer à l'agent (x). En recevant ces situations, l'agent (x) les ajoute dans sa base de contexte.

Enfin pour la trace de contexte dans le processus de décision, l'agent sélectionne les situations pertinentes à sa perception et construit une situation globale qui représente mieux la situation actuelle de l'agent dans son environnement. Ce mécanisme permet à l'agent de choisir le meilleur plan pour atteindre ses objectifs.

Suite à ce travail, nous pouvons envisager les perspectives suivantes :

- Développer un prototype qui permet de valider notre architecture, c'est-à-dire une implémentation de l'approche proposée.
- Proposer un processus d'apprentissage qui permet à notre architecture, d'améliorer sa base de contextes par de nouvelles situations contextuelles. Ce processus permet à l'agent BDI d'apprendre en étant isolé, à partir de sa propre expérience, et d'apprendre à travers sa communication avec les autres agents qui partagent avec lui son univers, profitant ainsi des expériences des autres.

Bibliographie

[1] Ferber, J., & Perrot, J. F. (1995). Les systèmes multi-agents: vers une intelligence collective. InterEditions.

[2] Demazeau, Y., & Costa, A. R. (1996, July). Populations and organizations in open multi-agent systems. In Proceedings of the 1st National Symposium on Parallel and Distributed AI (pp. 1-13).

[3] Briot, J. P. (2001). Principes et architecture des systèmes multi-agents (Vol. 217). Hermès Science Publications. (chapitre1 : introduction aux multi-agent).

[4] Jennings, N. R., Sycara, K., & Wooldridge, M. (1998). A roadmap of agent research and development. Autonomous agents and multi-agent systems, 1(1), 7-38.

[5] Briot, J. P. (2001). Principes et architecture des systèmes multi-agents (Vol. 217). Hermès Science Publications. (chapitre 2 modèle et architectures d'agents).

[6] Russell, S., & Norvig, P. (1995). Artificial intelligence: A modern approach.

[7] Nachet.B,(2014), *modèle multi-agent pour la concepetion de systèmes d'aide à la décision collective.*

[8] Briot, J. P. (2001). Principes et architecture des systèmes multi-agents (Vol. 217). Hermès Science Publications. (chapitre3: modèle de communication).

[9] Briot, J. P. (2001). Principes et architecture des systèmes multi-agents (Vol. 217). Hermès Science Publications. (*chapitre 5 : Environnements de développement*).

[10] Meyer, G. (2006). Formalisation logique de préférences qualitatives pour la sélection de la réaction d'un agent rationnel dialoguant (Doctoral dissertation, Université Paris Sud-Paris XI).

[11] Cohen, P. R., & Levesque, H. J. (1990). Intention is choice with commitment.Artificial intelligence, 42(2), 213-261.

[12] Rao, A. S., & Georgeff, M. P. (1995, June). BDI agents: From theory to practice. In ICMAS (Vol. 95, pp. 312-319).

[13] d'Inverno, M., Kinny, D., Luck, M., & Wooldridge, M. (1998). A formal specification of dMARS. In Intelligent Agents IV Agent Theories, Architectures, and Languages (pp. 155-176). Springer Berlin Heidelberg.

[14] Bratman, M. E., Israel, D., & Pollack, M. E. (1988). Plans and resource-bounded practical reasoning. Computational intelligence, 4(4), 349-355.

[15] Brézillon, P. (1999). Context in Artificial Intelligence: II. Key elements of contexts. Computers and artificial intelligence, 18, 425-446.

[16] Casali, A., Godo, L., & Sierra, C. (2005). Multi-context specification for Graded BDI-Agents. In Doctoral Consortium (p. 31).

[17] Bucur, O., Beaune, P., & Boissier, O. (2005, May). Architecture d'agent sensible au contexte pour la prise de décisions. In Proceedings of the 2nd French-speaking conference on Mobility and ubiquity computing (pp. 17-20). ACM.

[18] Mahraz.A, 2012, "modèle multi-agent pour l'aide à la décision de groupe,".

[19] EL GHAYAM, Y. La Sensibilité au contexte dans un environnement mobile.

[20] Miraoui, M. (2009). Architecture logicielle pour l'informatique diffuse: modélisation du contexte et adaptation dynamique des services (Doctoral dissertation, École de technologie supérieure).

[21] Pasquier, L. (2002). Modélisation de raisonnements tenus en contexte: application à la gestion d'incidents sur une ligne de métro (Doctoral dissertation, Paris 6).

[22] Turner, R. M., Rode, S., & Gagne, D. (2013). Toward Distributed Context-Mediated Behavior for Multiagent Systems. In Modeling and Using Context (pp. 222-234). Springer Berlin Heidelberg.

[23] Turner, R. M. (1999). A model of explicit context representation and use for intelligent agents. In Modeling and Using Context (pp. 375-388). Springer Berlin Heidelberg.

[24] Newell, A. (1982). The knowledge level. Artificial intelligence, 18(1), 87-127.

[25] Brooks, R. A. (1989). A robot that walks; emergent behaviors from a carefully evolved network. Neural computation, 1(2), 253-262.

[26] McCarthy, J., & Hayes, P. J. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Readings in artificial intelligence*, 431-450.

[27] Finin, T., Fritzson, R., McKay, D., & McEntire, R. (1994, November). KQML as an agent communication language. In Proceedings of the third international conference on Information and knowledge management (pp. 456-463). ACM.

[28] Bratman, M. (1987). Intention, plans, and practical reason.

[29] Rao, A. S., & Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. KR, 91, 473-484.

[30] Konolige, K., & Pollack, M. E. (1993, August). A representationalist theory of intention. In IJCAI (pp. 390-395).

[31] Krümpelmann, P., Thimm, M., Ritterskamp, M., & Kern-Isberner, G. (2008, May). Belief operations for motivated BDI agents. In Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1 (pp. 421-428). International Foundation for Autonomous Agents and Multiagent Systems.

[32] Pokahr, A., Braubach, L., & Lamersdorf, W. (2005). A goal deliberation strategy for BDI agent systems. In Multiagent System Technologies (pp. 82-93). Springer Berlin Heidelberg.

[33] Stratulat, T. (2002). Systèmes d'agents normatifs: concepts et outils logiques(Doctoral dissertation, Université de Caen).

[34] McCarthy, J. (1979). Ascribing mental qualities to machines.

[35] Devooght, K. (2007). Modélisation de la capacité d'un agent intentionnel en fonction de ses activités.

[36] Pokahr, A., Braubach, L., & Lamersdorf, W. (2005). Jadex: A BDI reasoning engine. In Multi-agent programming (pp. 149-174). Springer US.

[37] Dignum, F., Dunin-Keplicz, B., & Verbrugge, R. (2001). Agent theory for team formation by dialogue. In Intelligent Agents VII Agent Theories Architectures and Languages (pp. 150-166). Springer Berlin Heidelberg.

[38] Padgham, L., & Lambrix, P. (2000, July). Agent capabilities: Extending bdi theory. In AAAI/IAAI (pp. 68-73).

[39] Cap, M., Dastani, M., & Harbers, M. (2011, May). Belief/goal sharing bdi modules. In The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3 (pp. 1201-1202). International Foundation for Autonomous Agents and Multiagent Systems.

[40] McCarthy, J. (1993). Notes on formalizing context.

[41] Morris, A., Ross, W., & Ulieru, M. A Multi-Agent Context-Management System for RECON Intelligence Analysis.

[42] Georgeff, M., Pell, B., Pollack, M., Tambe, M., & Wooldridge, M. (1999). The belief-desire-intention model of agency. In *Intelligent Agents V: Agents Theories, Architectures, and Languages* (pp. 1-10). Springer Berlin Heidelberg.

[43] Bucur, O., Beaune, P., & Boissier, O. (2005, May). Définition et représentation du contexte pour des agents sensibles au contexte. In Proceedings of the 2nd French-speaking conference on Mobility and ubiquity computing (pp. 13-16). ACM.

[44] Edmonds, B. (2002, July). Learning and exploiting context in agents. InProceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3 (pp. 1231-1238). ACM.

[45] Dastani, M., de Boer, F. S., Dignum, F., Van Der Hoek, W., Kroese, M., & Meyer, J. J. (2002). Programming the deliberation cycle of cognitive robots. In*Proc. of the 3rd International Cognitive Robotics Workshop*.

[46] Olaru, A., Florea, A. M., & Seghrouchni, A. E. F. (2013). A context-aware multi-agent system as a middleware for ambient intelligence. Mobile Networks and Applications, 18(3), 429-443.

[47] Cohen, P. R., & Levesque, H. J. (1990). Persistence, intention, and commitment. Reasoning about actions and plans, 297-340.

[48] Rao, A. S., & Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. KR, 91, 473-484.

[49] Sadek, D. (1991). Attitudes mentales et interaction rationnelle: vers une théorie formelle de la communication. *Unpublished thèse, Université Rennes*, *1*.

[50] GHOMARI, A. R. (2013). Approche Méthodologique d'Acquisition de Connaissances Agrégées à base d'Agents cognitifs coopérants pour les systèmes d'aide à la décision stratégiques.

[51] Müller, J. P., & Pischel, M. (2011). The agent architecture inteRRaP: Concept and application.

[52] Amraoui, A., Benmammar, B., Krief, F., & Bendimerad, F. T. (2012, October). Négotiations à base d'Enchères dans les Réseaux Radio Cognitive. In*NOTERE (Nouvelles Technologies de la Répartition)/CFIP (Colloque francophone sur l'ingénierie des protocoles) 2012*.

[53] Mazouzi, H. (2001). *Ingénierie des protocoles d'interaction: des Systèmes Distribués aux Systèmes Multi-agents* (Doctoral dissertation, Paris 9).

[54] Dignum, F., Morley, D., Sonenberg, E. A., & Cavedon, L. (2000). Towards socially sophisticated BDI agents. In MultiAgent Systems, 2000. Proceedings. Fourth International Conference on (pp. 111-118). IEEE.

[55] Pereira, D., Oliveira, E., Moreira, N., & Sarmento, L. (2005, December). Towards an architecture for emotional BDI agents. In *EPIA* (Vol. 5, pp. 40-47).

[56] De Silva, L., Sardina, S., & Padgham, L. (2009, May). First principles planning in BDI systems. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2* (pp. 1105-1112). International Foundation for Autonomous Agents and Multiagent Systems.

[57] Schilit, B., Adams, N., & Want, R. (1994, December). Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on* (pp. 85-90). IEEE.

[58] Brown, P. J. (1995). The stick-e document: a framework for creating context-aware applications. *ELECTRONIC PUBLISHING-CHICHESTER-*, *8*, 259-272.

[59] Brown, P. J., Bovey, J. D., & Chen, X. (1997). Context-aware applications: from the laboratory to the marketplace. *Personal Communications, IEEE*, *4*(5), 58-64.

[60] Ryan, N., Pascoe, J., & Morse, D. (1999). Enhanced reality fieldwork: the context aware archaeological assistant. *Bar International Series*, *750*, 269-274.

[61] Ward, A., Jones, A., & Hopper, A. (1997). A new location technique for the active office. *Personal Communications, IEEE*, *4*(5), 42-47.

[62] Schmidt, A., Aidoo, K. A., Takaluoma, A., Tuomela, U., Van Laerhoven, K., & Van de Velde, W. (1999, January). Advanced interaction in context. In*Handheld and ubiquitous computing* (pp. 89-101). Springer Berlin Heidelberg.

[63] Brézillon, P., & Pomerol, J. C. (1999). Contextual knowledge sharing and cooperation in intelligent assistant systems. *Le Travail Humain*, 223-246.

[64] Chen, G., & Kotz, D. (2000). *A survey of context-aware mobile computing research* (Vol. 1, No. 2.1, pp. 2-1). Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College.

[65] Dey, A. K. (2001). Understanding and using context. *Personal and ubiquitous computing*, *5*(1), 4-7.

[66] Henricksen, K., Indulska, J., & Rakotonirainy, A. (2002). Modeling context information in pervasive computing systems. In *Pervasive Computing* (pp. 167-180). Springer Berlin Heidelberg.

[67] Brézillon, P., Borges, M., Pino, J. A., & Pomerol, J. C. (2004, July). Context-awareness in group work: Three case studies. In *Proc. of*.

[68] Petrelli, D., Not, E., Strapparava, C., Stock, O., & Zancanaro, M. (2000, April). Modeling context is like taking pictures. In *Proc. of the Workshop" The What, Who, Where, When, Why and How of Context-Awareness" in CHI2000*.

[69] Adam, C., Gaudou, B., Hickmott, S., & Scerri, D. Agents BDI et simulations sociales.

[70] Gwizdka, J. (2000, April). What's in the context. In *Computer Human Interaction* (Vol. 2000).

[71] Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G., Altmann, J., & Retschitzegger, W. (2003, January). Context-awareness on mobile devices-the hydrogen approach. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on* (pp. 10-pp). IEEE.

[72] Krause, M., & Hochstatter, I. (2005). Challenges in modelling and using quality of context (qoc). In *Mobility aware technologies and applications* (pp. 324-333). Springer Berlin Heidelberg.

[73] Turner, R. M. (1995, October). Intelligent control of autonomous underwater vehicles: The Orca project. In *Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on* (Vol. 2, pp. 1717-1722). IEEE.

[74] Turner, R. M. (1999). Context-mediated behavior: An approach to explicitly representing contexts and contextual knowledge for AI applications. In *Working Notes of the AAAI-99 Workshop on Modeling and Using Context in AI Applications, AAAI Technical Report*.

[75] Turner, R. M. (1989). When reactive planning is not enough: Using contextual schemas to react appropriately to environmental change. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 940-947).

[76] Turner, R. M. (1998). Context-mediated behavior for intelligent agents.*International Journal of Human-Computer Studies*, *48*(3), 307-330.

[77] Gonzalez, A. J., & Ahlers, R. H. (1995, May). Context-based representation of intelligent behavior in simulated opponents. In *Proceedings of the Computer Generated Forces & Behavioral Representation Conference* (pp. 489-493).

[78] Schank, R. C., & Abelson, R. P. (2013). *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press.

[79] Henricksen, K., & Indulska, J. (2004, March). Modelling and using imperfect context information. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on* (pp. 33-37). IEEE.

[80] Henricksen, K., Indulska, J., McFadden, T., & Balasubramaniam, S. (2005). Middleware for distributed context-aware systems. In *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE* (pp. 846-863). Springer Berlin Heidelberg.

[81] Henricksen, K., & Indulska, J. (2006). Developing context-aware pervasive computing applications: Models and approach. *Pervasive and mobile computing*, *2*(1), 37-64.

[82] Chen, H., Finin, T., & Joshi, A. (2005). *Semantic web in the context broker architecture*. MARYLAND UNIV BALTIMORE DEPT OF COMPUTER SCIENCE AND ELECTRICAL ENGINEERING.

[83] McCarthy, J., & Buvac, S. (1997). Formalizing context (expanded notes).

[84] Bauer, J., Kutsche, R. D., & Ehrmanntraut, R. (2003). Identification and modeling of contexts for different information scenarios in air traffic.*Technische Universität Berlin, Diplomarbeit*.

[85] Meunier, J. P., & Peraya, D. (2010). Introduction aux théories de la communication. Armando Editore.

[86] Brazier, F., Dunin-Keplicz, B., Treur, J., & Verbrugge, R. (1996, November). Beliefs, intentions and desire. In *proceedings of KAW* (Vol. 96, p. 5).

[87] van der Torre, L., & Weydert, E. (1998). Goals, desires, utilities and preferences. In *Proceedings of the ECAI'98 Workshop on Decision Theory meets Artificial Intelligence*.

[88] Thangarajah, J., Padgham, L., & Harland, J. (2002). Representation and reasoning for goals in BDI agents. *Australian Computer Science Communications*, *24*(1), 259-265.

[89] Miraoui, M., Tadj, C., & Amar, C. B. (2008). Architectural survey of context-aware systems in pervasive computing environment. *Ubiquitous Computing and Communication Journal*, *3*(3), 1-9.