République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed KHIDER - BISKRA
Faculté des Sciences et de Sciences de l'ingénieur
Département d'Informatique

N° d'ordre : ……...
Série : .....

Mémoire

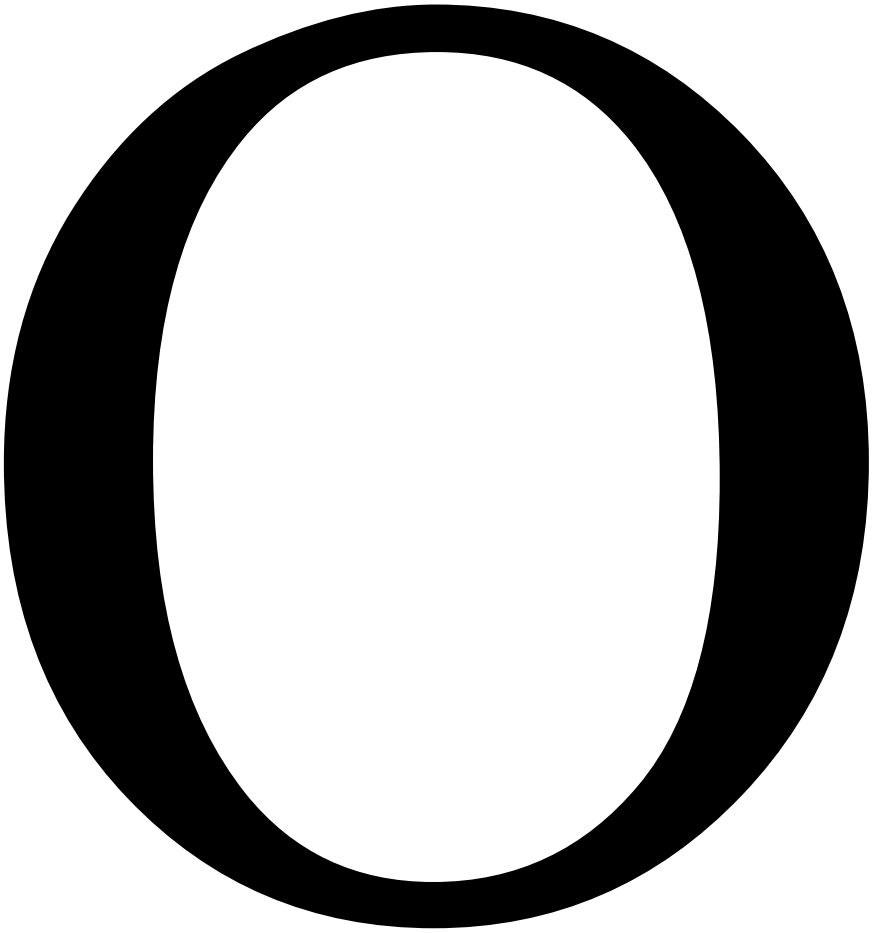
En vue d'obtention du diplôme de Magister en informatique
Option: Systèmes d'Informations Avancés et Intelligence Artificielle

Une approche basée agent-web pour le e-Learning

Réalisé par :
Melle. Naïma BAHI

Dirigé par :
Mr. Okba KAZAR

Année Universitaire: 2007 / 2008



SOMMAIRE

Chapitre 1

Introduction générale

Comment aider des apprenants pour qui l'accès à l'école est soudainement devenu impossible ? Une hospitalisation, les déplacements fréquents de la famille (des personnes qui voyagent), la pratique d'un sport de haut niveau voire une incarcération sont autant de situations qui ne permettent plus aux apprenants de fréquenter régulièrement une salle de classe. Dans ce cas et traditionnellement, des enseignants (souvent bénévolement) se rendent sur place mais se trouvent vite confrontés à des difficultés liées à la diversité des contenus et niveaux scolaires ainsi qu'à la disponibilité des apprenants (particulièrement vrai dans le milieu hospitalier).

Pour essayer de contourner ces problèmes et « amener l'école » à ces apprenants, les technologies de communication aujourd'hui disponibles apportent des perspectives de solution en autorisant la conception d'applications informatiques où deux interlocuteurs interagissent à distance via par exemple une fenêtre de dialogue ou un canal vidéo. En revanche, ces solutions n'apportent pas de soutien à l'interaction entre l'apprenant et l'enseignant au niveau de la connaissance. En effet, au sein d'une classe, chaque apprenant possède avec son enseignant un contexte de travail commun (son niveau, les problèmes à résoudre et qu'il a déjà résolu, les difficultés spécifiques liées à la matière enseignée et au problème en cours, etc.) qui n'existe pas dans le cas d'un enseignement à distance. Cela veut dire que pour qu'un enseignant puisse venir en aide efficacement à un apprenant, il faut au préalable reconstruire ce contexte commun. L'environnement informatique peut aider à cette reconstruction, par exemple en soutenant l'intervention d'un enseignant auprès d'un apprenant en lui apportant une information pertinente sur cet apprenant par rapport à la situation courante ou plus globalement par rapport à son historique, mais aussi en mettant judicieusement apprenants et enseignants en relation. Mais ce n'est pas tout. Assurer un suivi personnalisé de l'apprentissage permettant à chaque apprenant de recevoir des aides et des propositions prenant en compte sa propre trajectoire, favoriser les interactions d'apprenants entre eux dans une perspective du partage du savoir, mais aussi

favoriser la coopération d'enseignants pour une mise en commun des expertises de chacun sont des fonctionnalités qu'un outil d'aide à l'apprentissage et au métier d'enseignant doit proposer.

La situation d'apprentissage médiatisé par ordinateur est conçue comme un système Hommes-Machine caractérisé par la nature des pôles autour desquels s'articule le modèle (objet de l'apprentissage – Enseignant – Apprenant – Ordinateur), d'une part , par les relations que ces pôles entretiennent entre eux et avec leur environnement (institutionnel, en particulier), d'autre part.

Un Environnement Informatique d'Apprentissage Humain (EIAH) est un système informatique qui a pour objectif de favoriser l'apprentissage d'un domaine de connaissance par un apprenant. EIAH est l'un des exemples de E-Learning ou l'apprentissage électronique.

Les systèmes informatiques d'aide à l'apprentissage sont traditionnellement architecturés autour d'un module pédagogique unique : un tuteur artificiel. Il possède une expertise d'un domaine de connaissances et applique une stratégie d'enseignement pour interagir avec un apprenant afin de l'aider à résoudre un problème donné. Ce principe de fonctionnement en autonomie du couple apprenant-tuteur peut être satisfaisant jusqu'au moment où le système atteint ses limites ; la présence d'un enseignant humain, voire d'un autre apprenant devient alors indispensable. Ces systèmes peuvent donc être utilisés en complément d'un enseignement traditionnel, dans une classe par exemple. Toutefois, avec l'évolution des réseaux de communication tel l'Internet et les services associés comme les serveurs d'information de type Web par exemple, l'enseignement et les situations d'apprentissage se déplacent du cadre institutionnel de la salle de cours vers le domicile, l'entreprise, etc. Cela a mené vers la naissance de la formation à distance.

Plusieurs systèmes informatiques sont dédiés à la formation à distance. Ces systèmes sont ouverts et complexes. Un système ouvert est un système dont la structure peut changer dynamiquement. Ses composants ne sont pas connus à l'avance, changent au cours du temps, et sont essentiellement hétérogènes (en raison du fait qu'ils sont mis au point par différentes personnes, à différentes périodes, utilisant des techniques et outils de développement différents). Le système ouvert le plus connu est Internet. La formation à distance médiatisée par ordinateur s'appuie essentiellement sur Internet et exige le développement de technologies y relatives; il est évident que cette exigence induit de la

part des informaticiens la mise en place de systèmes ouverts. La formation à distance parce qu'elle se trouve au carrefour de plusieurs disciplines de sciences humaines (psychologie, sciences de l'éducation, sociologie, etc.) et des sciences de l'ingénieur (informatique, télécommunications) pose des problèmes complexes, diversifiés, interdépendants et difficilement prévisibles. Les outils adaptés pour faire face à la complexité dans le développement logiciel sont la modularité et l'abstraction. Le paradigme d'agent est un bon outil de modularisation, il permet de résoudre les problèmes posés par la formation à distance en développant des modules spécialisés (en termes de représentation et de type de problème résolu) dans un aspect particulier. L'interdépendance des problèmes de formation à distance est surmontée grâce à la coopération entre les agents mis en œuvre.

Autrement dit, la formation à distance en ligne exige de prendre en compte l'hétérogénéité des sources d'informations, des systèmes d'information et des centres de régulation des activités (au niveau de l'apprenant, du tuteur, du formateur, du concepteur, du coordonnateur de la formation) ; l'ouverture liée à l'ajout/retrait de sous-systèmes (logiciels, matériels, acteurs) sans stopper ou réinitialiser le système global ; la coopération entre les différents sous-systèmes pour produire des contenus pédagogiques, administrer, assister, évaluer les apprenants. Dans ce contexte, l'informatique sur laquelle s'appuie la formation à distance doit intégrer de nouveaux concepts, en s'appuyant sur des technologies fortement liées à Internet, afin de proposer des architectures logicielles qui permettent la coopération de multiples applications hétérogènes et distribuées.

Lorsqu'il s'agit de concevoir des systèmes informatiques distribués qui manipulent des connaissances hétérogènes, la technologie « agent » se révèle bien adaptée. En effet, les Systèmes Multi-Agents (SMA) présentent comme caractéristiques, d'une part de permettre le partage ou la distribution de la connaissance, et d'autre part, de faire coopérer un ensemble d'agents et de coordonner leurs actions dans un environnement pour l'accomplissement d'un but commun. Ces deux aspects correspondent bien aux besoins de résolution de problèmes et de coopération des différents acteurs d'un e-Learning, qu'ils soient humains ou artificiels. Un autre aspect prometteur est celui des agents dits « assistants » dont le rôle est de faciliter l'usage du système, grâce notamment à la personnalisation des offres et à la recommandation.

Il est donc nécessaire de concevoir aujourd'hui des systèmes qui prennent en compte la mobilité des apprenants, leur assurent un suivi individualisé afin de respecter

leur rythme d'apprentissage et mettent à leur disposition la présence humaine parmi l'ensemble des ressources pédagogiques accessibles.

Il est donc nécessaire de concevoir aujourd'hui des systèmes ou des environnements informatiques d'apprentissage humain qui prennent en compte la mobilité des apprenants, leur assurent un suivi individualisé afin de respecter leur rythme d'apprentissage et mettent à leur disposition la présence humaine parmi l'ensemble des ressources pédagogiques accessibles.

Ce travail s'inscrit dans cette optique. Notre contribution dans le domaine de la formation à distance consiste à concevoir et réaliser un système informatique capable d'initier l'apprentissage et de gérer un enseignement et un suivi individualisé. Nous utilisons les S.M.As pour bénéficier de leurs avantages.

Ce mémoire contient six autres chapitres. Dans le chapitre suivant, nous présentons un état de l'art sur les Systèmes Multi-Agents suivi par un chapitre dédié aux concepts du e-Learning. Le quatrième chapitre est consacrée à l'étude proprement dite du problème que l'on se propose de résoudre. Elle présente l'expression des besoins et les phases d'analyse et la conception du cycle de développement de notre application. Le chapitre cinq présente l'implémentation ainsi que la réalisation informatique. Les résultats et les tests sont validés dans le chapitre six. Nous terminons notre mémoire par une conclusion générale ayant pour objectif d'une part la synthèse de notre recherche et notre contribution dans le domaine du e-Learning, d'autre part nous n'oublions pas de signaler les perspectives possibles à ce travail.

Chapitre 2

Les Systèmes Multi Agents

2.1 Introduction

Le thème des systèmes multi-agents (S.M.As), s'il n'est pas récent, est actuellement un champ de recherche très actif. Cette discipline est à la connexion de plusieurs domaines en particulier de l'intelligence artificielle, des systèmes informatique distribués et du génie logiciel. C'est une discipline qui s'intéresse aux comportements collectifs produits par les interactions de plusieurs entités autonomes appelées agents, que ces interactions tournent autour de la coopération, de la concurrence ou de la coexistence entre ces agents. Ce chapitre est composé en deux grandes sections : les agents et les systèmes multi agents. La première section contient les définitions d'un agent, ses types, ses différentes architectures et les domaines d'application. La deuxième section introduit, tout d'abord, les notions de systèmes multi-agents (SMAs), les différentes applications et l'historique, et détaille par la suite les différentes questions que soulèvent la problématique des SMAs, en particulier : les interactions et la coopération, la coordination, la planification et la communication.

2.2 Agent

2.2.1 Qu'est ce qu'un agent?

Le concept d'agent a été l'objet d'études pour plusieurs décennies dans différentes disciplines. Il a été non seulement utilisé dans les systèmes à base de connaissances, la robotique, le langage naturel et d'autres domaines de l'intelligence artificielle, mais aussi dans des disciplines comme la philosophie et la psychologie [3]. Aujourd'hui, avec l'avènement de nouvelles technologies et l'expansion de l'Internet, ce concept est encore associé à plusieurs nouvelles applications comme agent ressource, agent courtier, assistant *personnel, agent interface, agent ontologique,* etc. Dans la littérature, on trouve une multitude de définitions d'agents. Elles se ressemblent toutes, mais différentes selon le

type d'application pour laquelle est conçu l'agent. A titre d'exemple, voici l'une des premières définitions de l'agent dûe à [12] :

Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents.

L'autonomie signifie que 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;

FRANCIS VAN AEKEN donne, dans [30], une définition beaucoup plus détaillée :

« *On appelle agent une entité physique ou virtuelle :*

a. qui est capable d'agir dans un environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement,

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, voire 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. »

Donc, l'agent est *une entité intentionnelle en interaction*. D'un autre coté, l'agent est vu *comme objet distribué* [16], surtout tournée vers une conception très "informaticienne" de l'agent, et se situe en contact étroit avec l'implémentation. Dans ce sens, un agent peut être défini comme un objet informatique (au sens des langages objets) dont le comportement peut être décrit par un "script", qui dispose de ses propres moyens de calcul (un agent est alors associé à un processus léger), et qui peut se déplacer de places en places (une place pouvant être un site informatique distant du site originel de l'agent) pour communiquer avec d'autres agents.

Ce qui précède est schématisé dans la **Figure(1)**, où :

Action Output

- Action est générée pour modifier l'environnement
- Pas de contrôle total sur l'environnement : contrôle partiel

Agent

Même si une action est exécutée deux fois, dans des circonstances différentes en apparence, elle peut avoir des effets différents ou peut échouer à avoir l'effet désiré. L'agent doit être préparé pour la possibilité d'échec de son action : environnements non déterministes. Normalement, un agent doit avoir un répertoire d'actions disponibles. Cet ensemble d'actions possibles représente les capacités de l'agent.

N.B : les actions ne sont pas toutes applicables dans toutes les situations.

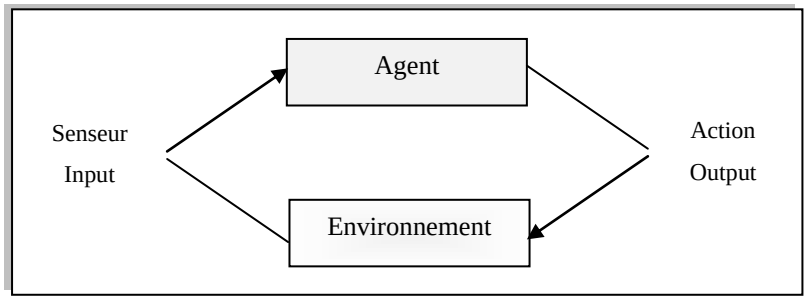


Figure 1 : Représentation de l'agent et son environnement

Les actions ont des pré-conditions associées qui définissent les situations possibles pour appliquer ces actions. Le problème clé que doit résoudre un agent est de décider quelles actions il doit exécuter pour mieux satisfaire ses objectifs.

Les architectures d'agents sont des architectures logicielles pour des systèmes de prises de décision qui sont embarqués dans un environnement.

La complexité du processus de prise de décision est liée à un certains nombre de propriétés de l'environnement [29]:

1. environnement Accessible / Non Accessible : dans un environnement accessible (observable) un agent peut obtenir de l'information complète, précise et à jour sur l'état de l'environnement.

2. environnement Déterministe / Non Déterministe : dans un environnement déterministe une action a un effet unique garanti.

3. environnement Statique / Dynamique : un environnement statique ne change que sous l'effet des actions d'un agent. Dans un environnement dynamique, les agents ne contrôlent pas entièrement la manière dont l' environnement change.

4. environnement Discret / Continu : un environnement est discret si il y a un nombre fini d'actions bien définies.

La classe d'environnement la plus complexe sont les environnements Inaccessibles, Non-Déterministes, Dynamiques et Continus.

Senseur Input

L'agent observe son environnement ; par des capteurs physiques dans le cas d'agents situés dans un monde réel ou des capteurs logiciels pour des agents logiciels.

''Les agents sont des systèmes informatiques, capables d'effectuer des actions autonomes dans un certain environnement pour atteindre certains objectifs. ''

2.2.2 Types d'agents

On distingue deux types d'agents [23]

1. **agent cognitif** *: c'est un agent qui possède une représentation de son environnement et de lui-même, il utilise cette représentation pour générer des plans explicites. Ces plans lui permettent d'atteindre ses objectifs. Il communique avec les autres à l'aide d'un langage de haut niveau* (*actes de langage* [25]).

2. **agent réactif** *: à l'opposition d'un agent cognitif, un agent réactif ne possède pas de modèle symbolique de son environnement de lui-même ou des autres agents, et son comportement peut être résumé en stimuli-réponse.*

Systèmes d'agents cognitifs	Systèmes d'agents réactifs
Représentation explicite de l'environnement	Pas de représentation explicite
Peut tenir compte de son passé	Pas de mémoire de son histoire
Agents complexes	Fonctionnement Stimulus/action
Petit nombre d'agents	Grand nombre d'agents

Tableau 1 : comparaison entre agents cognitifs et agents réactifs

2.2.3 Agent intelligent

B. Chaib-draa et al. dans [12] ont proposé la définition suivante pour un agent intelligent: Un agent intelligent est un agent qui possède une autonomie d'action flexible pour atteindre ses objectifs .

« flexible» c'est-à-dire l'agent dans ce cas est :

- capable de répondre à temps(réactif) : l'agent doit être capable de percevoir son environnement et 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 de compléter ses tâches ou aider ces agents à accomplir les leurs.

2.2.4 Architecture Abstraite d'un Agent [10]

S = {s1, s2,....} ensemble des états de l'environnement

A = {a1, a2,...} ensemble d'actions représentant les capacités de l'agent

Un agent peut être considéré comme une fonction Action : $S^* \rightarrow A$

qui fait correspondre une séquence d'états de l'environnement à une action.

Un agent modélisé par une fonction de cette forme s'appelle Agent Standard. Dans cette forme, un agent décide quelle action exécuter en se basant sur son historique : son expérience à ce point. Cette expérience est représentée par une séquence d'états de l'environnement : ceux que l'agent a rencontré.

Le non déterminisme du comportement d'un environnement peut être modélisé par une fonction :

$$env : S * A \rightarrow \varphi (S)$$

qui prend l'état courant de l'environnement $s \in S$ et une action $a \in A$ (accomplie par l'agent) pour donner un ensemble d'état de l'environnement env(s,a) qui est le résultat de l'exécution de l'action a dans l'état s.

Si l'exécution d'une action sur n'importe quel état donne est un ensemble contenant un seul élément (c a d tous les ensembles de la portée de env sont des singletons) alors l'environnement est déterministe et son comportement peut être prédis avec précision.

On peut représenter l'interaction de l'agent et de l'environnement comme une historique. Une historique h est une séquence :

$$h : s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_3} \ldots\ldots \xrightarrow{a_{u-1}} s_u \xrightarrow{a_u} .....$$

où : s_0 est l'état initial de l'environnement

a_u est la $u^{ième}$ action que l'agent a préféré exécuter et s_u est le u^{ieme} état de l'environnement (l'un des résultats possibles de l'exécution de l'action a_{u-1}dans l'état s_{u-1})

Si Action : $S^* \rightarrow A$ est un agent,

env : $S * A \rightarrow \varphi (S)$ est un environnement, et

s_0 est un état initial de l'environnement

La caractéristique comportementale d'un agent Action : $S^* \rightarrow A$ dans un environnement env : $S * A \rightarrow \varphi (S)$ est l'ensemble de toutes les historiques qui satisfont ces propriétés.

Si une propriété Ø est vraie pour toute ces historiques, alors cette propriété peut être considérée comme une propriété invariante de l'agent dans l'environnement.

La fonction de décision de l'agent peut être représentée par deux sous-systèmes : la perception et l'action (Figure2).

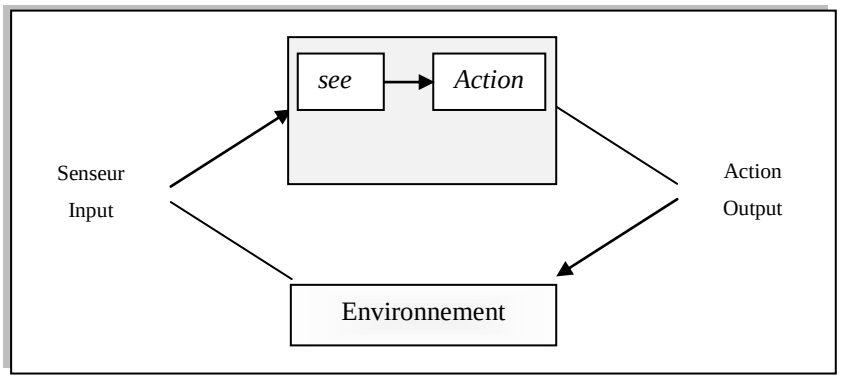


Figure (2) : Fonctions constituant la prise de décision de l'agent

La fonction *see* : - exprime la capacité de l'agent à observer son environnement(la perception).

- peut être implémentée en hard dans le cas d'un agent situé dans environnement physique réel (caméra vidéo, capteur infra rouge, robot mobile...). Pour un agent logiciel, les capteurs peuvent être des commandes systèmes ...

- Le résultat de la fonction *see* est une perception :

P = ensemble des perceptions (non vide)

Donc See est une fonction qui fait une correspondance entre les états de l'environnement et les perceptions *See :* S → P

La fonction A*ction* représente le processus de prise de décision de l'agent, elle sera maintenant *Action* : P* → A qui fait correspondre entre les perceptions et les actions.

- **Agents Purement Réactifs**

Agents qui décident ce qu'il faut faire sans référence à leur historique. Prise de décision basée uniquement sur le présent sans aucune référence au passé. Le comportement d'un agent purement réactif peut être représenté par une fonction *Action* : S → A

- **Agents avec état (mémoire)**

On va maintenant considérer un agent qui sauvegarde son état (**Figure (3)**)

Etat : L'ensemble de tous les états internes de l'agent, on le note par I. La fonction de perception de l'agent est inchangée : *See :* S → P

La fonction action de sélection est définie par : *Action* : I → A

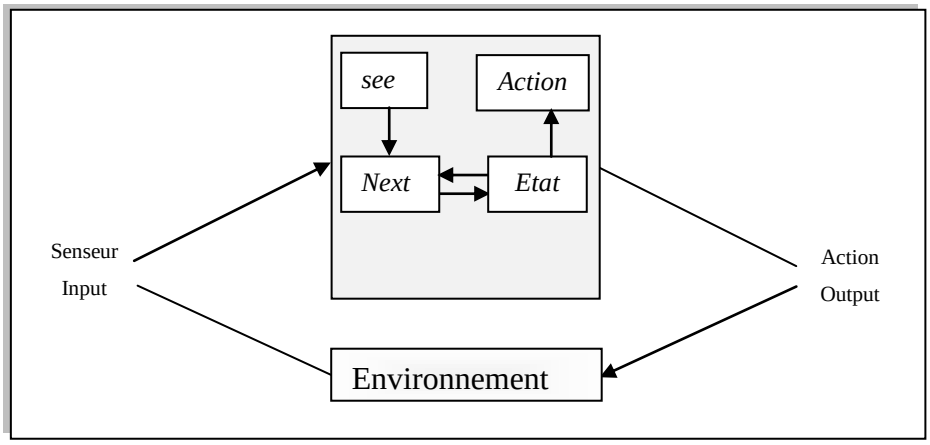


Figure (3) : Agent avec état (mémoire)

La fonction Next fait la correspondance entre une perception et un état interne avec un état interne *Next* : P * I → I

1. L'agent commence dans état interne i_0,
2. Il observe un état de l'environnement s et
3. Génère une perception *See*(s)

4. L'état interne de l'agent est mis à jour par la fonction *Next* et devient *Next*(i_0, *See*(s))

5. L'action sélectionnée par l'agent est alors *Action* (*Next*(i_0, *See*(s))

6. L'action est exécutée et l'agent entre dans un autre cycle.

2.2.5 Architectures concrètes d'un agent

On a modélisé la prise de décision d'un agent comme une fonction abstraite *Action* qui, en quelque sorte, on indique quelle action à exécuter dans les deux cas : un agent qui maintient un état et un agent qui ne maintient pas son état. On n' a pas indiqué comment implémenter cette fonction. Dans cette section on va préciser comment ?

En termes d'architecture, on a quatre classes d'agents [29,13,10]: *agents basés sur la logique*, a*gents , agents BDI* (Belief, Desire, Intention), et finalement des agents avec une a*rchitecture en couches*.

2.2.5.1 Agents basés sur la logique

Approche traditionnelle de construction de systèmes artificiels intelligents (IA Symbolique) suggère qu'un comportement intelligent peut être généré dans un système en :

- Une représentation symbolique de son environnement,
- Une représentation symbolique de son comportement, et
- Manipulation syntaxique de cette représentation.

Dans cette approche, la représentation symbolique sont les *formules logiques* et la manipulation syntaxique correspond à la déduction logique ou à la preuve de théorème. L'idée d'agents comme preuves de théorèmes est séduisante. La stratégie de prise de décision de l'agent est codée comme une théorie logique, et son processus de sélection de l'action se réduit à un problème de preuve.

La théorie φ peut être considérée comme une spécification de comment doit se comporter un agent. L'approche traditionnelle pour implémenter un système qui satisfait cette spécification implique le raffinement de spécification à travers une série d'étapes progressivement plus concrètes jusqu'à l'obtention de l'implémentation. Mais dans la vue d'agents en tant que démonstrateur de théorèmes, on n'a pas de raffinement. Dans ce cas, φ est considérée comme une spécification exécutable qui est directement exécutée pour produire le comportement de l'agent.

☑ **Avantages**

- Elégance et une sémantique propre

✘ **Inconvénients**

- La complexité de démonstration de théorèmes pose le problème de savoir si les agents (considérés comme des démonstrateurs de théorème) peuvent opérer dans un environnement à fortes contraintes temporelles.
- La fonction *See* de l'agent (sa capacité de perception) sera difficile à représenter du fait de la nature de l'environnement (complexe, dynamique et, éventuellement physique).

2.2.5.2 Les Architectures Réactives

Alternatives au paradigme I.A Symbolique basées sur:

- Le rejet de la représentation symbolique et de la prise de décision basée sur une manipulation syntaxique de ces représentations,
- L'idée que le comportement intelligent et rationnel est intimement lié à l'environnement de l'agent. Le comportement intelligent de l'agent n'est pas séparée, mais c'est un produit de l'inter action que l'agent maintient avec son environnement.
- L'intelligence émerge de l'interaction de comportements plus simples.

Ces alternatives sont qualifiées de :

- Comportementales (behavioral) : développer et combiner des comportements individuels,
- Localisées (situated) : Les agents sont situés dans un environnement. Ils ne sont pas dissociés de cet environnement.
- Réactives : le systèmes sont considérés comme simplement réagir à un environnement sans raisonnement.

Exemple : Architecture Subsumptive [27]

☑ **Avantages**

- Simplicité,
- Economie,
- Robustesse contre l'échec.

✖ **Inconvénients**

- Si les agents n'ont pas de modèles de l'environnement, ils doivent avoir des informations suffisantes disponibles dans leurs environnements locaux pour pouvoir déterminer une action acceptable.
- Le problème de la prise en compte des informations non locales.
- Le problème de l'apprentissage des agents de leurs propres expériences et améliorer leurs réponses avec le temps.

Solutions ?

Agents qui évoluent dans le temps pour accomplir certaines taches.

2.2.5.3 Les Architectures BDI (Belief- Desire – Intention)

Elles sont basées sur la compréhension du raisonnement pratique. Le processus de décider, moment par moment, quelle action exécuter pour aboutir à notre objectif.

Il y'a sept composantes dans un agent BDI (**Figure (4)**) :

1- Un ensemble de croyances courantes : **Beliefs** représentant les informations que l'agent a sur son environnement.

2- Une fonction de révision **brf** qui prends en entrée une perception de l'agent et la croyance courante de l'agent et fournit un nouvel ensemble de croyances.

3- Une fonction de génération **options générer** les options qui détermine les options disponibles à l'agent (ses désirs).

4- Un ensemble d'options courantes représentant les actions possibles disponibles à l'agent **Desirs**.

5- Une fonction filter **Filtre** qui représente le processus de délibération de l'agent et qui détermine les intentions de l'agent sur la base de ses croyances courantes, ses désirs et ses intentions.

6- Un ensemble d'intentions **Intentions** courantes représentant la concentration actuelle de l'agent : les objectifs qu'il essaye d'atteindre.

7- Une fonction de sélection de l'action (exécute) **Action** qui détermine l'action à exécuter sur la base des intentions courantes.

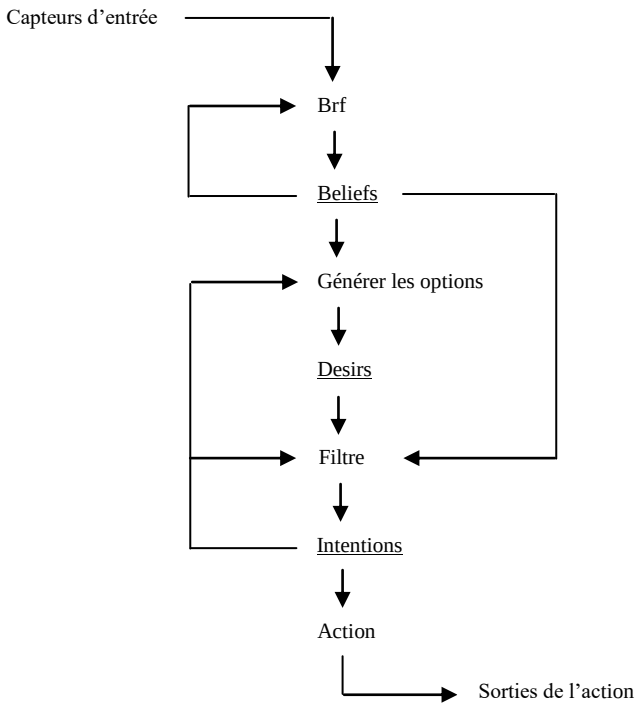


Figure (4) : Fonctionnement d'agent BDI

☑ **Avantages**

- Intuitif : on connaît le processus quoi faire et comment le faire. On a une idée de ce que sont les notions de croyances, désirs et intentions.
- Une décomposition fonctionnelle claire indiquant quels sont les sous systèmes nécessaires pour construire l'agent

✖ **Inconvénients**

- Comment implémenter d'une manière efficace les fonctions

2.2.5.4 Les Architectures en Couches

Etant la nécessité d'avoir un agent à comportement réactif et à comportement proactif, une décomposition évidente serait de créer des sous-systèmes pour chaque type de comportements.

On a donc une classe d'architectures dans lesquelles les différents sous-systèmes sont organisés sous forme d'hiérarchies de couches qui interagissent entre elles. On doit avoir, au moins, deux couches : une pour chaque comportement (réactif et proactif). Le type d'architecture en couches est lié aux flots d'information et de contrôle entre les couches.

• **Architecture en couche horizontale (Figure (5)):** Dans cette architecture les couches logicielles sont chacune reliées aux entrées des capteurs et aux sorties des actions. Chaque couche se comporte comme un agent et produit des propositions sur le type d'actions à exécuter.

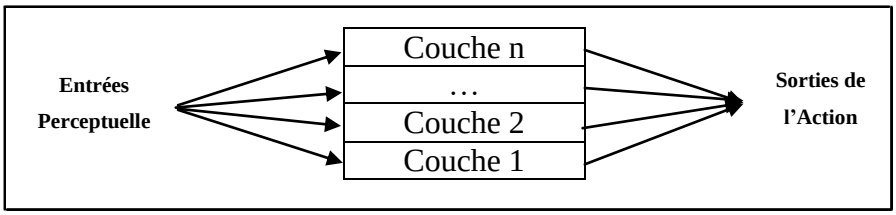


Figure (5) : Architecture Horizontale

avantage

- simplicité conceptuelle.

- **Problème**

Si nous avons un agent ayant n différents types de comportements, alors on implémente n couches différentes. Mais, puisque les couches sont en compétition, les unes avec les autres pour suggérer des actions, alors un problème relatif à la possibilité que le comportement global de l'agent ne soit pas cohérent.

- **Solution**

Pour assurer la consistance des architectures horizontales, elle contiennent généralement une fonction de médiation. Cette fonction a pour rôle de prendre des décisions concernant quelle couche aura le contrôle sur l'agent à n'importe quel moment.

- ***Problème***

Ce besoin de contrôle centralisé pose un problème : le concepteur doit considérer toutes les interactions possibles entre les couches. Si on a n couches dans l'architecture et que chaque couche peut suggérer m actions possibles, alors on a m^n interactions à considérer. En plus, l'utilisation d'un système de contrôle centralisé provoque un goulot d'étranglement dans la prise de décision de l'agent.

- ***Solution***

l'utilisation d'une architecture en couches verticale.

• **Architecture en couche verticale :** dans cette architecture les entrées des capteurs et les sorties des actions sont pris en charge par, au plus, un agent. Il y a des architectures verticales à une passe **(Figure 6-a)** et des architectures à deux passes **(Figure 6-b)**.

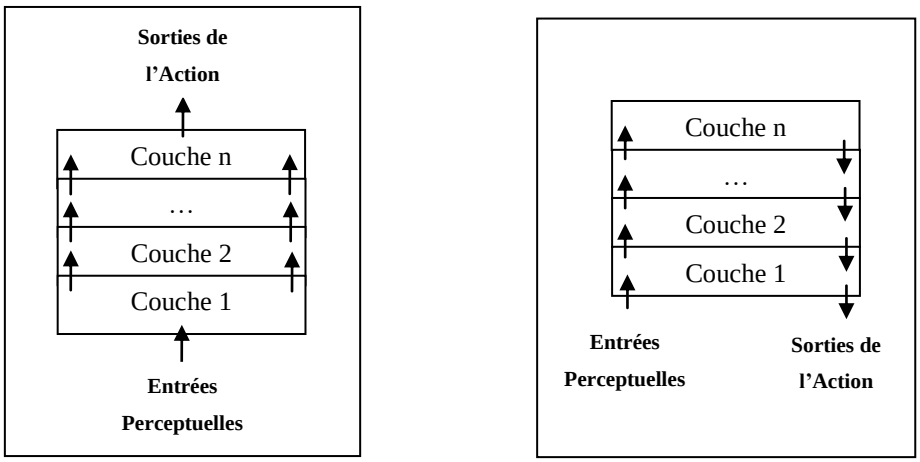


Figure (6) : a- architecture verticale à une passe **b- Architecture verticale à deux passes**

Dans l'architecture à une passe, le flot de contrôle passe séquentiellement par chaque couche, jusqu'à ce que la couche finale génère l'action de sortie.

Dans l'architecture à deux passes, les informations circulent dans un sens (de bas en haut : une passe) et le flot de contrôle revient en sens inverse.

☑ Avantages

- Dans les deux architectures en couches verticales (une passe et deux passes), la complexité d'interactions entre les couches est réduite. Nous avons n-1 interfaces entre n couches. Si chaque couche peut suggérer m actions. Nous aurons, donc au plus, $m^2(n-1)$ interactions à prendre en considération. Ceci est plus simple que dans le cas de l'architecture horizontale.
- Les architectures en couches sont les classes d'architectures les plus populaires.
- Les couches permettent une décomposition fonctionnelle naturelle.
- Dans cette architecture, il est facile de voir comment le comportement réactif, proactif et social sont générés par les couches réactive, proactive et sociale.

✖ Inconvénients

- Pour prendre une décision, il faut que le contrôle passe entre les différentes couches. Ceci n'est pas tolérant aux fautes. L'échec dans n'importe quelle couche aurait des conséquences sérieuses sur les performance de l'agent.
- Ces architectures n'ont pas la même clarté sémantique et conceptuelle des architectures sans couches (plates).

- L'interaction entre les couches (partiellement résolu par l'architecture en couches verticale à deux passes [11])

2.2.6 Les catégories d'applications des agents autonomes

Les programmes d'aujourd'hui interagissent avec leur usager selon un paradigme appelé manipulation directe. C'est-à-dire que le programme fait exactement ce que l'usager lui demande de faire. Le mécanisme de contrôle se situe alors dans l'être humain. Ils ne savent pas communiquer avec un être humain au même niveau que eux et ne fonctionne pas en dehors des commandes qui eux sont envoyées. La coordination des actions reste donc l'apanage de l'être humain. Ici encore, la machine est passive, attendant les ordres élémentaires que l'utilisateur veut bien lui envoyer.

Cependant, il pourrait être intéressant d'avoir des programmes qui prennent l'initiative dans certaines circonstances. Ainsi, un logiciel prendrait maintenant une part active pour aider l'usager à accomplir ses tâches. Les agents sont des entités autonomes qui sont mues par des désirs et qui poursuivent leurs propres buts. Ils peuvent communiquer avec d'autres agents (ou avec un utilisateur) dans un langage de haut niveau qui laisse place à l'élaboration d'interactions complexes

Des applications de tels agents se retrouvent principalement dans les lecteurs de courriels, les lecteurs de groupes de discussions actifs ainsi que les navigateurs actifs. On trouve [29,3]:

- Agent d'interface (ex. Microsoft agent) : les applications traditionnelles ne réagissent qu'à l'input direct de l'utilisateur pour exécuter des instructions spécifiques. En général, il n'y a pas d'aide proactive pour les tâches complexes. L'utilisateur pourrait être invité à collaborer avec un agent. Dans ce cadre, les deux pourraient prendre l'initiative de la communication et de la réalisation de tâches déterminées.
- Agents collaboratifs : le but d'un système à agents collaborateurs est de proposer, par la collaboration, des solutions qui vont plus loin que les capacités individuelles de chaque agent isolé. Le but peut être de permettre l'échange d'informations entre divers systèmes patrimoniaux (comme les mainframes, les systèmes experts et les datawarehouse). C'est peut-être aussi une solution pour les systèmes d'information distribués ou les connaissances distribuées, et même pour étudier le comportement social des relations humaines.

- Agents mobiles : sont chargé de migrer d'une machine à une autre pour exécuter par exemple un code et ne pas surcharger le réseau.
- Agents d'information/Internet : le world wide web c'est dans ce contexte que les agents d'information jouent essentiellement un rôle. Les agents d'information nous offrent un journal personnalisé, sachant où ils doivent chercher pour nous et comment *trouver* ces informations.
- Agents réactifs : les agents réactifs constituent une catégorie spéciale en ce sens qu'ils ne comprennent pas de modèle interne sur leur "environnement de travail". Ils réagissent avec de simples réponses stimulus.

2.2.7 Domaines d'applications des agents

Les agents sont utilisés dans plusieurs domaines en particulier [29]:

- Les systèmes de production industrielle et le contrôle de processus,
- Les systèmes de télécommunication,
- Le contrôle de trafic aérien,
- La gestion du trafic et des transports,
- La recherche et filtrage d'information,
- Le commerce électronique,
- Gestion des processus « business »,
- La gestion des ressources humaines et des compétences
- La gestion de la force de travail (mobile),
- La défense, gestion hospitalière,…

En fait, l'agent en tant qu'entité individuelle peut s'avérer limité dans des bien des cas, surtout au vu de ce qu'on voit aujourd'hui comme applications distribuées et pour lesquelles un ensemble d'agents mettant en commun compétences et connaissances parait plus que nécessaire. Ce type d'agents forment ce qu'on appelle des systèmes multi-agents.

2.3 Les systèmes multi-agents

2.3.1 Définition des systèmes multi-agents

Un *système multi-agent* [12] est un système distribué composé d'un ensemble d'agents. Contrairement aux systèmes d'IA, qui simulent dans une certaine mesure les

capacités du raisonnement humain, les S.M.A sont conçus et implantés idéalement comme un ensemble d'agents interagissant, le plus souvent, selon des modes de *coopération*, de *concurrence* ou de *coexistence*.

Un S.M.A est généralement caractérisé [12] par :

1. chaque agent a des informations ou des capacités de résolution de problèmes limitées, ainsi chaque agent a un point de vue partiel;
2. il n'y a aucun contrôle global du système multi-agent;
3. les donnés sont décentralisées;
4. le calcul est asynchrone.

Les S.M.A sont des systèmes idéaux pour représenter des problèmes possédant de multiples méthodes de résolution, de multiples perspectives et/ou de multiples résolveurs. Ces systèmes possèdent les avantages traditionnels de la résolution distribuée et concurrente de problèmes comme la modularité, la vitesse (avec le parallélisme), et la fiabilité (dûe à la redondance). Ils héritent aussi des bénéfices envisageable de l'Intelligence Artificielle comme le traitement symbolique (au niveau des connaissances), la facilité de maintenance, la réutilisation et la portabilité mais surtout, ils ont l'avantage de faire intervenir des schémas d'interaction sophistiqués. Les types courants d'interaction incluent la coopération (travailler ensemble à la résolution d'un but commun) ; la coordination (organiser la résolution d'un problème de telle sorte que les interactions nuisibles soient évitées ou que les interactions bénéfiques soient exploitées) ; et la négociation (parvenir à un accord acceptable pour toutes les parties concernées).

Bien que les S.M.A offrent de nombreux avantages potentiels [23], ils doivent aussi relever beaucoup de défis et de problèmes au niveau de la conception et l'implémentation des S.M.A [12].

Les S.M.A sont à l'intersection de plusieurs domaines scientifiques : informatique répartie et génie logiciel, intelligence artificielle, vie artificielle. Ils s'inspirent également d'études issues d'autres disciplines connexes notamment la sociologie, la psychologie sociale, les sciences cognitives et bien d'autres. C'est ainsi qu'on les trouve parfois à la base des [9]:

- systèmes distribués;
- interface personnes-machines;
- bases de données et bases de connaissances distribuées coopératives;

- systèmes pour la compréhension du langage naturel ;
- protocoles de communication et réseaux de télécommunications;
- programmation orientée agents et génie logiciel;
- robotique cognitive et coopération entre robots;
- applications distribuées comme le web, l'Internet, le contrôle de trafic routier, le contrôle aérien, les réseaux d'énergie, etc.

2.3.2 Historique

En 1980, un groupe de chercheurs s'est réuni pour discuter des défis concernant la résolution "intelligente" de problèmes dans un système comportant plusieurs résolveurs de problèmes [12]. Lors de cette réunion, il a été décidé que l'intelligence artificielle distribuée n'axerait pas ses travaux sur les détails de bas-niveau de la parallélisation ni sur comment paralléliser les algorithmes centralisés mais plutôt sur le fait de savoir comment un groupe de solutionneurs de problèmes pourrait coordonner ses efforts afin de résoudre des problèmes de manière efficace.

On peut dire que les S.M.A ont vu le jour avec l'avènement de l'intelligence artificielle distribuée (I.A.D). A ses début toutefois, l'I.A.D ne s'intéressait qu'à la coopération entre résolveurs de problèmes afin de contribuer à résoudre un but commun. Pour y parvenir, on divisait en général, un problème en sous problèmes, et on allouait ces sous-problèmes à différents résolveurs qui sont appelés à coopérer pour élaborer des solutions partielles. Celles-ci sont finalement synthétiser en une réponse globale au problème de départ. Ainsi donc, l'IAD au départ privilégiait le "problème à résoudre" tout en mettant l'accent sur la résolution d'un tel problème par de multiple entités intelligentes. Dans les SMA d'aujourd'hui, les agents sont (entre autres) autonomes, possiblement préexistants et généralement hétérogènes. Dans ce cas, l'accent est plutôt mis sur le fait de savoir comment les agents vont coordonner leurs connaissances, buts et plans pour agir et résoudre des problèmes.

Parmi les systèmes développés tout au début, il y avaient :

Le modèle des acteurs : Un des premiers modèles proposé à l'époque de l'I.A.D fut le modèle des Acteurs [12]. Les acteurs sont des composantes autonomes d'un système qui communiquent par messages asynchrones. Ils sont composés d'un ensemble de primitives.

Les acteurs s'avèrent être un modèle assez naturel pour le calcul parallèle. Cependant, les divers modèles d'acteurs font face à un problème de cohérence. Ils éprouvent également des difficultés à atteindre des buts globaux avec seulement des connaissances locales.

Le protocole Contract Net : Le protocole du *Contract Net* [12] fut une des premières solutions au problème d'allocation de tâches auquel fait face généralement un ensemble de résolveurs de problèmes. Dans ce protocole, les agents peuvent prendre deux rôles: *gestionnaire* ou *contracteur*. L'agent qui doit exécuter une tâche donnée (le *gestionnaire*) commence tout d'abord par décomposer cette tâche en plusieurs sous-tâches. Il doit ensuite annoncer les différentes sous-tâches au reste des agents de l'environnement. Les agents qui reçoivent une annonce de tâche à accomplir peuvent ensuite faire une proposition devant refléter leur capacité à remplir cette tâche. Le gestionnaire rassemble ensuite toutes les propositions qu'il a reçues et alloue la tâche à l'agent ayant fait la meilleure proposition.

Parmi les premières applications [12] développées à l'aide de la technologie multi-agents, on retrouve une application dans le contrôle du trafic aérien et une autre dans la surveillance de véhicules motorisés.

2.3.3 Origines des S.M.A

- La première est due aux limites de l'intelligence artificielle classique sur le plan de la structuration et de l'organisation des connaissances. La difficulté qu'il y a à traduire un ensemble d'expertises sous une forme unifiée a amené les chercheurs à développer ce que l'on a d'abord appelé des systèmes multi-experts, c'est-à-dire des systèmes mettant en jeu plusieurs bases de connaissances plus ou moins coordonnées. Ce faisant, on a pu constater que le problème de la coopération entre plusieurs bases de connaissances se révélait un enjeu crucial qui dépassait de loin le problème de la multi-expertise [16].
- La deuxième trouve son origine dans la nécessité de trouver des techniques de modélisation et de simulation performantes dans le domaine des sciences du vivant au sens large du terme. L'évolution des écosystèmes habités, notamment, montrent qu'il est difficile de rendre compte de leur évolution par un ensemble d'équations différentielles. Une approche dans laquelle les individus sont directement représentés

sous forme d'entités informatiques semble une voie prometteuse et a contribué à l'essor du domaine.

- La troisième provient de la robotique. Le développement de la miniaturisation en électronique permet de concevoir des robots qui disposent d'une certaine autonomie quant à la gestion de leur énergie et à leur capacité de décision. On a pu alors montrer qu'un ensemble de petits robots ne disposant que de capacités élémentaires de décision et d'intelligence pouvait facilement rivaliser avec les performances d'un seul robot "intelligent", nécessairement plus lourd et plus difficile à gérer. Le problème alors revient, ici encore, à faire coopérer ces entités de manière à ce qu'elles assurent les fonctions désirées.
- Enfin, la quatrième est issue du développement de l'informatique et des systèmes distribués en particulier. Avec la généralisation des réseaux et des ordinateurs parallèles, il devient de plus en plus important de pouvoir faire coopérer plusieurs composants logiciels au sein d'environnements hétérogènes et distribués. Le problème ne réside plus dans le contenu des programmes, qui peuvent avoir des fonctionnalités diverses, mais dans leur capacité à collaborer avec d'autres programmes à la réalisation d'un objectif commun.

2.3.4 Architectures des S.M.A

On distingue deux types d'architectures pour les SMA [23]

- *SMA à contrôle centralisé ou a base de tableau noir.*
  - Composé de trois éléments :
    - Les connaissances (les agents)
    - Le tableau noir
    - Le mécanisme de contrôle
  - Pas de communication directe
  - Interaction via le partage d'un même espace de travail (le tableau noir)

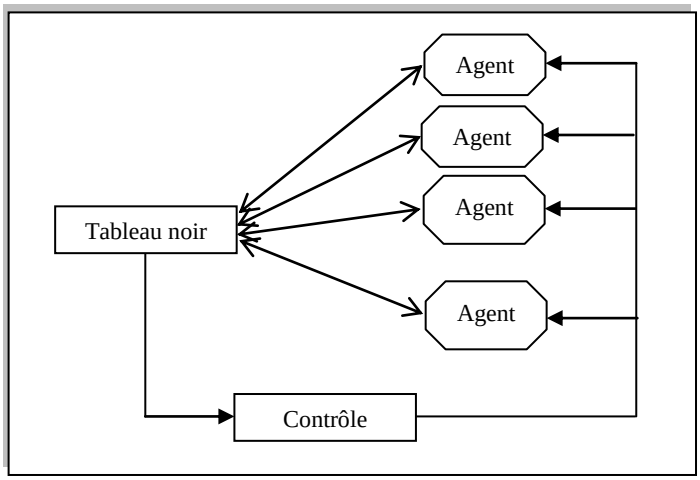


Figure (7) : Architecture centralisée du SMA

- *SMA a contrôle distribué*

  - Distribution totale des connaissances et du contrôle

  - Caractéristiques :

    • Traitement local

    • Communication par envoi de message

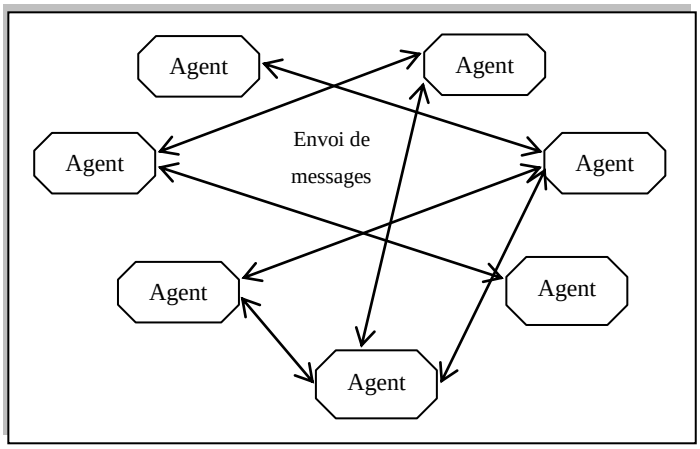


Figure (8) : Architecture distribuée du SMA

- Le langage d'Acteur est la technique la plus utilisée pour la mise en œuvre de ce type d'architecture

- Un Acteur regroupe au sein d'une même entité un ensemble de connaissances : les accointances et un script.

2.3.5 Interactions et communications entre agents

Pour un agent, interagir avec un autre constitue à la fois la source de sa puissance et l'origine de ses problèmes. C'est en effet parce qu'ils coopèrent que des agents peuvent accomplir plus que la somme de leurs actions des agents, mais c'est aussi à cause de leur multitude qu'ils doivent coordonner leurs actions et résoudre des conflits. Pour un agent, l'autre est à la fois le pire et la meilleure des choses.

Traiter le problème de l'interaction c'est se donner les moyens non seulement de décrire les mécanismes élémentaires permettant aux agents d'interagir [16], mais aussi d'analyser et de concevoir les différentes formes d'interaction que des agents peuvent pratiquer pour accomplir leurs tâches et satisfaire leurs buts. Tout d'abord, les agents doivent être capables, par le biais de la communication, de transmettre des informations, mais surtout d'induire chez l'autre un comportement spécifique. Communiquer est donc une forme d'action particulière qui, au lieu de s'appliquer à la transformation de l'environnement, tend à une modification de l'état mental du destinataire. Par exemple, demander à un autre d'exécuter une tâche, tend à provoquer chez l'autre une intention d'accomplir cette tâche et constitue donc une manière de satisfaire un objectif sans réaliser la tâche soi-même.

Les interactions des agents d'un S.M.A sont motivées par l'interdépendance des agents selon les trois dimensions suivants [12,13]: leurs buts peuvent être compatibles ou non; les agents peuvent désirer des ressources que les autres possèdent; un agent peut disposer d'une capacité nécessaire à un autre agent pour l'accomplissement d'un des plans d'action de dernier.

Nous nous intéressons aux deux critères principales dans les S.M.A : la coordination (l'interaction) et la communication.

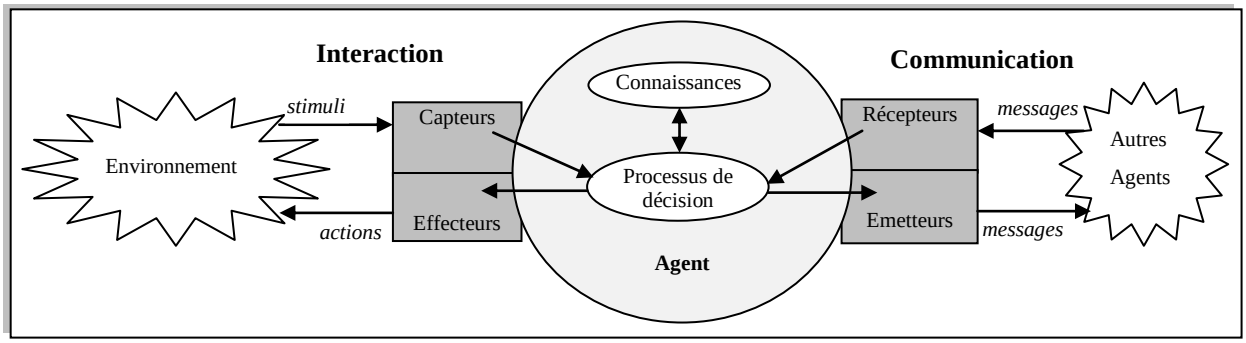


Figure (9) : Interactions et Communication

2.3.5.1 Coordination

Les agents communiquent pour mieux atteindre les buts qu'ils se sont fixés ou qu'on (système) leur impose. La coordination entre agents impose le partage d'un environnement. Le degré de coopération est fixé par le niveau de résolution collective recherché (maintien de propriétés, évitement de cas de blocage, partage optimum de ressources ...). Elle analyse la manière dont les actions des différents agents doivent être organisées dans le temps et l'espace de manière à réaliser les objectifs.

Il existe plusieurs types de coordination, suivant la nature des agents en jeu [10]. On parlera de *coopération* entre des agents non-antagonistes ou coopératifs et de *négociation* entre des agents compétitifs (agents égo-centrés).

De nombreux exemples de coordination existent dans la vie quotidienne : deux déménageurs déplaçant un meuble lourd, deux jongleurs échangeant des balles avec lesquelles ils jonglent, des personnes qui parlent à tour de rôle en se passant un micro, etc.

B. Chaib-draa et al. dans [12] notent que deux des composantes fondamentales de la coordination entre agents sont l'allocation de ressources rares et la communication de résultats intermédiaires.

Dans ce contexte, les agents doivent être capables de communiquer entre eux de façon à pouvoir échanger les résultats intermédiaires. Pour l'allocation des ressources partagées, les agents doivent être capables de faire des transferts de ressources.

Il existe trois processus fondamentaux de coordination [12]:

- ✓ *ajustement mutuel* : est la forme de coordination la plus simple qui se produit quand deux ou plusieurs agents s'accordent pour partager des ressources en vue d'atteindre un but commun. Habituellement les agents doivent échanger de nombreuses informations et faire plusieurs ajustements à leurs propres comportements en tenant compte des comportements des autres agents. Dans cette forme de coordination aucun agent n'a un contrôle sur les autres agents et le processus de décision est conjoint. Ce processus est utilisé en cas du travail en petit groupe.
- ✓ *supervision directe* : apparaît quand un ou plusieurs agents ont déjà établi une relation dans laquelle un des agents (l'agent superviseur) a un contrôle sur les autres (les agents subordonnés). Dès que la taille du groupe grandit et que le nombre de tâches augmente, le nombre de liens et la quantité d'informations échangées peuvent devenir rapidement un handicap sérieux. Un groupe important peut être partagé efficacement en

sous-groupes de manière à ce que la plupart des interactions s'effectuent dans les sous-groupes et que les quelques interactions nécessaires entre les sous-groupes soient prises en charge par les superviseurs de ces sous groupes.

- ✓ *coordination par standardisation* : le superviseur coordonne les activités en établissant des procédures que doivent suivre les subordonnés dans des situations identifiées.

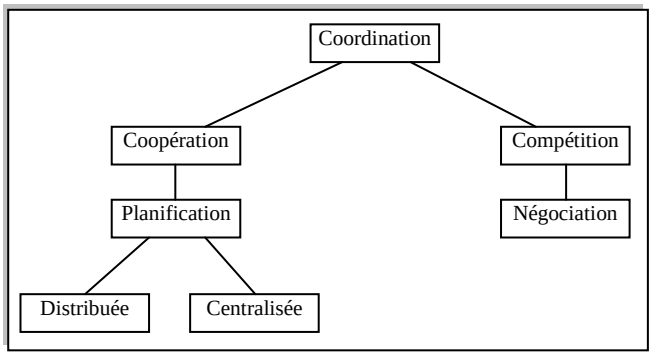


Figure (10) : coordination

2.3.5.2 Négociation entre agents

En général, une négociation [10] intervient lorsque des agents interagissent pour prendre des décisions communes, alors qu'ils poursuivent des buts différents (agents égocentrés [12]). Plus précisément, l'objectif de la négociation est de résoudre des conflits qui pourraient mettre en péril des comportements coopératifs. 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.

Parmi les protocoles les plus étudiés pour la négociation, il y a protocole du réseau contractuel ("Contract-Net"), il a été une des approches les plus utilisées pour les SMA [12] dont laquelle les agents coordonnent leurs activités grâce à l'établissement de contrats pour atteindre des buts spécifiques. Autre protocole est le système ConFract [14].

2.3.5.3 Coopération

Elle s'intéresse à la manière de répartir le travail entre plusieurs agents, qu'il s'agisse de techniques centralisées ou distribuées [16]. Elle présentes quelques avantages à savoir [13]:

- ✓ augmenter le taux de finalisation des tâches grâce au parallélisme;

- ✓ augmenter le nombre de tâches réalisables grâce au partage de ressources (information, expertise, dispositifs physiques, etc.);
- ✓ augmenter les chances de finaliser des tâches en les dupliquant et en utilisant éventuellement des modes de réalisation différents;
- ✓ diminuer les interférences entre tâches en évitant les interactions négatives (conflits entre les agents égo-centrés).

Mais Cela peut résulter en des coûts de communication élevés [13].

2.3.5.4 Planification dans un environnement multi-agent

l'intelligence artificielle qui cherche à répondre à la question: "Que doit-on faire?", c'est-à-dire quelle action poser et dans quel ordre. Comme un agent est une entité qui pose des actions, on voit rapidement l'intérêt de la planification pour les agents autonomes [12,13].

On peut obtenir une plus grande coordination si on oriente les comportements des agents vers des buts communs en établissant explicitement une division du travail entre les agents [12]. Des techniques comme la planification centralisée pour des groupes d'agents, la conciliation de plans, la planification distribuée, l'analyse organisationnelle, sont toutes des façons d'aider les agents à aligner leurs activités en assignant les tâches après avoir raisonné sur les conséquences de réaliser ces tâches dans des ordres particuliers.

Dans une approche de planification multi-agent, un plan multi-agent est un plan qui est crée afin que plusieurs agents puissent l'exécuter. La création d'un tel plan peut être faite par un seul ou par plusieurs agents.

En planification multi-agent centralisée un agent est responsable pour la création du plan qui spécifie les actions planifiées pour tous les agents concernés. Dans cette approche, les plans des agents sont d'abord créés de façon individuelle; ensuite un agent centralisateur rassemble ces plans et les analyse pour identifier les conflits. L'agent centralisateur en question essaye de résoudre les conflits en modifiant les plans locaux des autres agents et en introduisant des commandes de communication afin que les agents se synchronisent de façon appropriée.

Dans une approche de planification distribuée, les activités de planification sont réparties au sein d'un groupe d'agents. Cette approche est utilisée quand un seul agent ne peut pas avoir une vue globale des activités du groupe. B. Chaib-draa et al. dans [12] ont distingué, quant à lui, deux classes de problèmes en planification distribuée : la

planification dirigée par les tâches et la coordination de plans. Dans les systèmes de planification dirigés par les tâches, il y a un but initial ou une tâche initiale qui est décomposée en sous-buts ou sous-tâches et qui est répartie entre plusieurs agents (décomposition de problème “top-down”). Par ailleurs, la coordination de plan s’attaque à des situations dans lesquelles il existe des plans d’agents (vision centrée sur les agents) et le problème consiste à réconcilier ces plans avant de les exécuter dans un environnement commun.

En général, la planification multi-agent nécessite une forme ou une autre de synchronisation de plans qui peut être réalisée à divers moments : pendant la décomposition de plan, pendant la construction de plan ou après celle-ci.

Bien entendu, la planification contribue à la coordination, dans la mesure où lorsque les agents adoptent un plan « bien fait », ils agissent généralement de manière coordonnée.

2.3.5.5 Communication entre agents

Les agents peuvent interagir soit en accomplissant des actions linguistiques (en communiquant entre eux), soit en accomplissant des actions non-linguistiques qui modifient leur environnement. En communiquant [12,13], les agents peuvent échanger des informations et coordonner leurs activités. Dans les S.M.A deux stratégies principales ont été utilisées pour supporter la communication entre agents: les agents peuvent échanger des messages directement ou ils peuvent accéder à une base de données partagées (appelée tableau noir ou “blackboard”) dans laquelle les informations sont postées. Les communications sont à la base des interactions et de l’organisation sociale d’un S.M.A.

2.3.5.5.1 Transfert de plans ou de messages

Dans l’approche de transfert de plans, un agent X communique son plan en totalité à un agent Y et Y communique son plan en totalité à X . Cette approche est coûteuse en ressources de communication.

Dans des situations réelles en plus, il n’est pas possible de formuler à l’avance des plans en totalité. Par conséquent, on a besoin de communiquer des stratégies générales aux agents plutôt que des plans, et donc il est nécessaire que les agents puissent échanger des messages en utilisant protocoles précis. Les types de messages combinés avec les réponses attendues conduisent à la spécification de protocoles d’interaction.

2.3.5.5.2 Echange d'informations grâce à un tableau noir

En intelligence artificielle la technique du tableau noir ("blackboard") est très utilisée pour spécifier une mémoire partagée par divers systèmes .Dans un SMA utilisant un tableau noir, les agents peuvent écrire des messages, insérer des résultats partiels de leurs calculs et obtenir de l'information.

2.3.5.5.3 Actes de discours et conversations

Les philosophes du langage ont développé au cours des 25 dernières années la théorie des actes de discours qui considère que "dire quelque chose c'est en quelque sorte agir" [12]. Les paroles sont interprétées comme des actions appelées actes de discours. Les chercheurs en intelligence artificielle ont très tôt pensé à étudier comment les actes de discours peuvent être utilisés pour influencer les états mentaux (les croyances, les désirs, les intentions ou buts, les capacités, etc.) des agents.

On mesure ainsi l'importance d'associer à un modèle de la communication inter-agents un modèle de raisonnement sur les états mentaux, tout en tenant compte de la dimension sociale de l'interaction. Les relations sociales ont aussi un impact évident dans les conversations humaines : la forme et le contenu des actes de discours des locuteurs tiennent compte des rôles et statuts sociaux des participants à la conversation.

2.3.5.5.4 Communication utilisant KQML, ACL et les conversations

Le langage KQML [17] a été proposé pour supporter la communication inter-agents. Ce langage définit un ensemble de types de messages (appelés abusivement "performatifs") et des règles qui définissent les comportements suggérés pour les agents qui reçoivent ces messages. Les types de messages de KQML sont de natures diverses : simples requêtes et assertions (ex."ask", "tell"); instructions de routage de l'information ("forward" et "broadcast"); commandes persistantes ("subscribe", "monitor"); commandes qui permettent aux agents consommateurs de demander à des agents intermédiaires de trouver les agents fournisseurs pertinents ("advertise", "recommend", "recruit" and "broker").

Ces dernières années, KQML semble perdre du terrain au profit d'un autre langage plus riche sémantiquement ACL (pour Agent Communication Language). Un langage mis de l'avant par la FIPA qui s'occupe de standardisé les communications entre agents. ACL est basé également sur la théorie du langage et a bénéficié grandement des résultats de

recherche de KQML. Si toutefois, les deux langages se rapprochent au niveau des actes du langage, il n'en ait rien au niveau de la sémantique et il semble qu'un grand soin a été apporté au niveau de ACL tant au niveau de certains protocoles qui sont plus explicites qu'au niveau de la sémantique des actes eux mêmes.

2.3.6 Méthodes de conception des systèmes multi-agents

Les S.M.As sont souvent des systèmes complexes qui demandent de longs efforts de développement [12]. Du début des années 80 jusqu'au milieu des années 90, les efforts des chercheurs se sont surtout concentrés sur l'exploration de nouveaux concepts, la mise ne place de théories, la réalisation de divers prototypes de S.M.As. Ce bouillonnement d'idées est tout à fait normal dans un jeune domaine technologique : c'est ce que l'on a observé pour les bases de données dans les années 70, les systèmes experts et les systèmes orientés-objets dans les années 80. Les S.M.A se développent de proche en proche pour devenir une technologie tout entière. Une technologie avec ses éléments de base nécessaires pour fournir des produits bien faits à consommer et à utiliser par des clients du marché informatique.

La technologie S.M.A s'intègre dans le cadre du génie logiciel. Les S.M.A doivent offrir un outil pour le développement de logiciels et d'applications. De ce fait, différentes visions sont plausibles :

- Trouver des solutions à des problèmes par des S.M.A.
- Développer des SMA avec des outils classiques du GL.
- Développer des SMA avec d'autres S.M.A.
- …

En particulier, on s'intéresse au développement des SMA. La question pertinente : est que le développement des SMA a atteint son stade de maturité ? Existe -t- il de techniques, de méthodes et méthodologies qui définissent des consignes et des procédures à suivre pour construire un système multi-agents résolvant un certain problème ou atteignant un certain objectif ?

D'une autre part, est-il possible de suivre le style de développement classique du génie logiciel pour construire des S.M.A, à la différence claire qui distingue ceux-ci des autres ? La recherche dans le domaine GL agent est l'une des branches très active,

prometteuse et ambitieuse, en S.M.A. Des efforts, des explorations et des propositions, nous, mettent sur la route vers un génie logiciel orienté agents.

Cependant, on distingue, tout court, entre deux visions ou approches de développement. Une classique dite descendante (i.e. la plus part des méthodologies citées précédemment sont de ce type) et une autre moderne ou innovée dite ascendante [24]. Cette dernière, appelée aussi exploratoire, est caractéristique aux S.M.A, en particulier aux S.M.A liés à la vie artificielle. Dans cette approche, à partir de comportements individuels d'agents, on cherche à explorer le comportement global d'un groupe d'agents, coexistant dans un environnement commun. Une remarque a été bien expérimentée consiste en l'inégalité entre le comportement global du groupe et la simple somme des comportements locaux des différents agents [24]. On parle de comportement global *émergent*. Des recherches s'intéressent à exploiter ce principe d'émergence, pour résoudre des problèmes complexes. Dans ces techniques de résolution, aucun but, aucun plan ni aucune communication n'est explicite, tout est implicite et la solution du problème émerge des interactions inter agents [24].

La première approche de développement est héritée du génie logiciel classique. Résoudre un problème nécessite la construction d'un programme, d'un logiciel ou d'un système, dont l'exécution fournit la solution qu'on cherche. Cette construction passe par une suite de phases complémentaires. Ces phases définissent un cycle de développement. Le système synthétisé sera caractérisé par une structure (ou bien un squelette) et un comportement. Le comportement du système représente la dynamique (actions et tâches) nécessaire pour atteindre l'objectif du concepteur (qui est de résoudre le problème de départ). La plupart des traveaux, proposés à ce stade, s'appuient sur des techniques de modélisation empruntées à des méthodes connues en développement orienté-objets ou en ingénierie des connaissances pour aider à la construction des modèles d'agents, de l'architecture du S.M.A, pour la spécification des modèles d'organisation, d'interaction, etc.

Quand on exploite une approche basée agents, pour la résolution des problèmes, le système-solution sera un système multi-agents. On cherche à développer un S.M.A qui exhibe un comportement conduisant à résoudre un problème. Par conséquent, on doit chercher et explorer les éléments qui génèrent et influencent ce comportement. On doit donc savoir quels sont les éléments de base des S.M.A ? Qu'influence leur comportement ?

Que définisse ou redéfinisse leurs organisations ? Que cause, où qui gère leurs conflits survenant ?

Les interactions dans les S.M.A, sont vues comme des éléments de base générant et influençant leurs comportements. L'idée de penser à l'interaction, comme élément de base dans les SMA, n'est pas récente. L'interaction est à la base de constitution des organisations des systèmes multi-agents, l'organisation est un résultat émergent des interactions entre les agents [19]. En éthologie, l'interaction est vue aussi, comme un fondateur de l'intelligence. Le développement cognitif d'un être vivant est influencé par les échanges et les interactions entre les individus de sa société. Lorsque ces échanges et interactions, manquent ou disparaissent, l'individu reste dans un état primitif [24]. Autres [22] voient que la conception des S.M.A est mieux si elle est basée sur la distinction entre les différents composants d'un S.M.A (agent, environnement, interaction, organisation). Cette approche permet de décomposer le problème général de manière à analyser séparément chacune de ses parties (composants), à concevoir et développer des réponses à chacune d'entre elles, puis à intégrer l'ensemble en un système cohérent.

Malgré tout ça, on a besoin de méthodes pour faire du Génie Logiciel Orienté Multi-Agent [9].

2.4. Conclusion

Nous avons vu, tout au long de ce chapitre que la technologie agent et multi-agent n'est pas un concept voué à rester sur les tablettes des laboratoires de recherche puisque plusieurs exemples d'applications existent déjà.

Les personnes qui ont développé des S.M.A disent toutefois qu'il est difficile de concevoir et de bâtir un système multi-agent. En effet, la construction de ce type de système comportent toutes les difficultés inhérentes aux systèmes répartis, auxquelles s'ajoute le caractère flexible et sophistiqué des interactions entre agents. A cela s'ajoute le fait que les concepteurs de S.MA. font de nos jours face à deux difficultés majeures. Tout d'abord, l'absence de méthodologie systématique qui permettrait de spécifier et de structurer une application multi-agents. Ensuite, le manque d'outils commerciaux pour bâtir des S.M.A.

Comme on peut encore le constater le domaine des systèmes multi-agents demeure encore aujourd'hui un domaine rempli de défis à surmonter, autrement dit un domaine très ouvert pour la recherche.

Chapitre 3

Le e-Learning

3.1 Introduction

Les nouvelles Technologies de l'Information et de la Communication "TIC" améliorent profondément nos façons de nous informer, de communiquer et de nous former. Cette émergence technologique fait apparaître un nouveau mode d'apprentissage connu sous le nom de e-Learning. Celui-ci est basé sur l'accès à des formations en ligne, interactives et parfois personnalisées, diffusées par l'intermédiaire d'un réseau (Internet ou Intranet) ou d'un autre média électronique. Cet accès permet de développer les compétences des apprenants, tout en rendant le processus d'apprentissage indépendant du temps et du lieu. e-Learning est le but de cet chapitre. Après voir donner des définitions, l'historique et la typologie du e-Learning, nous allons présenter ses exigences ainsi sa projection sur le marché. Ensuite, les différents objets pédagogiques dans le e-Learning. les plates-formes, les différents normes et standards pour le e-Learning sont pris en charge dans ce chapitre grâce à leur nécessité dans notre application. A ce stade, nous allons présenter quelques travaux qui ont été réalisés dans ce contexte. Enfin, nous présentons les obstacles du e-Learning ainsi d'une conclusion du chapitre.

3.2 Définitions

L'enseignement par Internet (ou télé-enseignement ou e-Learning) constitue un des moyens pédagogiques actuels et prometteurs [6]. Il souffre cependant de défauts principalement liés à l'absence relative de l'enseignant et donc à la difficulté d'adaptation de l'enseignement au niveau et au comportement de l'apprenant.

Deux définitions du e-Learning peuvent être proposées :

- la première désigne un concept de formation dans lequel la technologie (e pour électronique) se place progressivement dans tous les plans de l'activité de la formation.

- La seconde, un peu plus restrictive, désigne comme e-Learning tout processus ou toute organisation de formation utilisant les technologies du Web.

C'est une méthode de formation / d'éducation qui permet théoriquement de s'affranchir de la présence physique d'un enseignant à proximité. En revanche, le rôle du *tuteur* (enseignant) distant apparaît avec des activités de facilitateur et de médiateur.

Plusieurs termes sont utilisés pour traduire le terme e-Learning, on trouve le terme *apprentissage en ligne* et parfois certains emploient le terme *e-Formation* [5].

3.3 Historique

Le concept de formation à distance est apparu [7] vers le milieu du dix-neuvième siècle, faisant référence aux études par correspondance. Il a connu une évolution considérable au fil des années, commençant par l'acheminement des cours sous forme de papier par poste ou par fax, passant par les cassettes audio et vidéo, puis la diffusion hertzienne via la radio et les émissions de télévision pour arriver à l'Enseignement Assisté par Ordinateur (EAO), grâce au développement des technologies éducatives et l'intégration de l'outil informatique pour la réalisation des cours interactifs. Les premiers systèmes d' Enseignement Assisté par Ordinateur sont apparus dans les années 1970. L'objectif était d' abord l'apprentissage en tant que transfert de connaissances. Une multitude de programmes éducatifs furent développés, mais vite délaissés car leur contenu était limité et leur utilisation rigide. L'aspect cognitif a été totalement ignoré avec peu de travaux de recherche, de diagnostic et d'adaptation de stratégies. Les connaissances et les décisions étaient préétablies, sans souci du comportement de l'usager. Mais, malgré leur application limitée, ces systèmes ont eu des retombées significatives dans le domaine de l'éducation. L'application des techniques de raisonnement offertes par l'Intelligence Artificielle (IA) et les Systèmes Experts (SE), dans le système éducatif, ont permis des innovations en introduisant un niveau d'interaction plus élevé entre l'apprenant et le système. C'est ce qui a donné naissance aux systèmes d'Enseignement Intelligemment Assisté par Ordinateur (EIAO) qui pallient les nombreux inconvénients des systèmes précédents. Les recherches effectuées afin d'adapter l'apprentissage au niveau de connaissances de l'apprenant a donné lieu à une nouvelle génération de systèmes appelés : Systèmes Tutoriels Intelligents (STI) ou systèmes d'apprentissage un à un (tuteur – apprenant). Ces systèmes ont pour but de reproduire le comportement d'un tuteur intelligent afin de dispenser un enseignement

personnalisé à l'utilisateur. Ces systèmes offrent une possibilité de génération dynamique d'exercice, des adaptations au niveau de difficultés selon les performances de l'étudiant ainsi que l'analyse de l'interprétation du comportement de l'étudiant. En effet, les Systèmes Tutoriels Intelligents sont capables de réaliser des inférences sur des connaissances de l'étudiant, et peuvent interagir intelligemment avec lui en adaptant dynamiquement les sujets à présenter en fonction des résultats acquis et du mode d'apprentissage qui lui convient le mieux.

Plusieurs systèmes ont été recensés [7] :

✓ *Les systèmes critiques* **:** ont pour but de guider l'apprenant dans la résolution de problèmes, en particulier les problèmes de conception, dans le but de l'amener vers une solution correcte ;

✓ *Les systèmes démonstrateurs* **:** utilisés pour imiter ou simuler un phénomène dans le but de l'enseigner à l'étudiant ;

✓ *Les systèmes hypermédias et les environnements interactifs d'apprentissage ou l'apprentissage par découverte* **:** permettent à l'apprenant d'apprendre tout en explorant, de manière guidée ou libre, l'objet d'apprentissage qui est simulé ;

✓ *Les systèmes procéduraux* **:** utilisés pour enseigner les procédures nécessaires pour accomplir une tâche donnée ;

✓ *Les systèmes sociaux* **:** font intervenir des agents externes (enseignants, étudiants) pour communiquer avec l'apprenant pendant la résolution du problème ;

✓ *Les systèmes socratiques* **:** fonctionnent à l'aide d'un dialogue avec l'apprenant en utilisant un dialogue socratique ou des questions/réponses. Ils sont très appropriés pour l'apprentissage des stratégies de résolution de problèmes et la présentation des informations factuelles.

3.4 Typologies du e-Learning

Différentes typologies cohabitent dans le e-Learning. On peut définir une formation en ligne selon deux axes [5]:

- L'axe de l'espace formateur - apprenant. Il peut exister une grande proximité, lorsque le professeur donne un cours devant ses étudiants, on parle de formation en *présentiel*. Lorsque le formateur n'est pas en contact avec l'apprenant, on parlera de formation à distance ou de formation *distancielle*.

▪ L'axe du temps de la communication. Si la communication est directe, immédiate, on parle d'outil *synchrone*. Lorsqu'un délai existe entre une question et sa réponse on parlera d'outil *asynchrone*.

❖ **Formation asynchrone** : Dans ce type de formation, l'échange avec les autres apprenants ou avec les tuteurs s'effectue via des modes de communication ne nécessitant pas une connexion simultanée. Il peut s'agir de forums de discussion ou bien encore de l'échange d'e-mails. Par ailleurs, ce mode de formation repose souvent sur un apprentissage dit "autodirigé", avec des cours, des exercices et des évaluations automatisées, impliquant une certaine autonomie de l'apprenant. Elle est appelée aussi l'*autoformation* [21].

❖ **Formation synchrone :** Ce type de formation implique la connexion simultanée des participants à une session de formation. Ils peuvent communiquer en temps réel, soit par web-conférence ou visioconférence. Ils peuvent également partager des applications et interagir sur celles-ci au moment où le tuteur leur donne la main sur le document. Le tuteur peut intervenir sur l'écran d'apprenant. Il analyse ainsi avec plus d'efficacité les capacités et les connaissances acquises par les apprenants. La classe virtuelle [21] est l'un des exemple de la formation synchrone. Elle permet un échange à distance et en temps réel avec le formateur.

Parmi les autres outils disponibles [21], on peut aussi citer : Les campus ou universités virtuels d'entreprise, la télévision interactive, les bornes interactives, les simulations, les jeu-formation, les agents intelligents.

Quand la typologie de formation est 100% apprentissage en ligne [5], l'apprenant va suivre sa session d'apprentissage en ligne entièrement à distance. Il n'interagira pas directement en vis-à-vis avec le ou les formateurs. Il pourra néanmoins naturellement avoir des contacts avec eux via des outils type forum, chat, téléphone, email. En revanche, le mode d'apprentissage mixte [5] (appelé aussi Blended Learning en anglais) désigne l'utilisation conjointe du e-Learning et du mode classique d'apprentissage appelé souvent « présentiel ». En général, l'apprenant va ainsi alterner entre des sessions à distance en ligne et des sessions en face-à-face avec le ou les formateurs. Un modèle souvent utilisé est ainsi d'effectuer une première introduction au sujet avec une ressource à distance, puis une période en face à face avec un enseignant suivra. Une session de débriefing est souvent aussi ajoutée en fin de formation, quelque temps après celle suivie en face à face**.**

3.5 Le e-Learning et ses exigences

Le processus d'apprentissage traditionnel peut être caractérisé par [7]:

- *Une autorité centralisée* : le contenu est sélectionné par l'enseignant ;
- *Une forte livraison des informations et connaissances* : les enseignants transmettent les informations et connaissances aux apprenants ;
- *Un manque de personnalisation* : le contenu pédagogique doit satisfaire les besoins de l'ensemble des apprenants ;
- *Un processus d'apprentissage statique et linéaire* : un contenu inchangé.

Le coût cher, la lenteur et parfois le manque de concentration sont les principales conséquences d'une telle organisation sur l'apprentissage. Cependant, les marchés d'apprentissage de nos jours demandent des méthodes *just-in-time* pour assister le besoin de savoir *need-to-know* des apprenants. En effet, il est clair qu'un nouveau mode d'apprentissage guidé par les exigences de rapidité, du juste à temps est nécessaire.

La rapidité ne demande pas seulement un contenu adéquat du matériel d'apprentissage (très spécifique) mais aussi des mécanismes puissants pour organiser un tel matériel. Ainsi, l'apprentissage doit être un service en ligne personnalisé, initié par les profils des utilisateurs et les besoins pédagogiques. Ceci peut être résolu par un système d'apprentissage de type e-Learning, orienté apprenant, personnalisé avec un processus d'apprentissage non linéaire et dynamique. L'objectif est de remplacer les anciennes façons temps/place/contenu de l'apprentissage prédéterminé par des processus d'apprentissage : à temps, à la place de travail et à la demande. Il représente "*l' ensemble des outils et des informations qui permettent d'améliorer la performance 'via' l'utilisation d' Internet et des technologies de l' information.*" [21].

Les avantages [21] de ces nouveaux modes de formation sont nombreux : simplicité d'accès, souplesse des horaires, gain de temps, interactivité, flexibilité, autonomie, personnalisation du parcours permettant à l'utilisateur de s'affranchir de toute contrainte.

Les inconvénients [21] ne sont pas absents pour autant. Exigeant une forte concentration du stagiaire, elles provoquent rapidement une certaine fatigue. Autre frein : le risque d'être souvent interrompu par un collègue ou par un coup de téléphone. Les sessions peuvent enfin être perturbées par des problèmes de connexions au réseau internet.

3.6 e-Learning et le marché

A l'heure actuelle, quatre grands types d'acteurs sont présents sur le marché du e-Learning [21]:

- Les acteurs traditionnels de l'éducation et de la formation ainsi que les éditeurs de contenus pédagogiques,

- Les spécialistes du e-Learning

- Les grands groupes du secteur informatique et bureautique (80% de l'offre actuelle),

- Les nouveaux intermédiaires qui créent, hébergent et commercialisent des cours en ligne.

Ces différents acteurs se disputent les trois principales branches de l'industrie qui sont [21]:

- *L'offre de technologies* : Domaine des sociétés qui développent et commercialisent des plates-formes logicielles permettant l'intégration, la diffusion et la gestion des contenus (IBM, Saba, Click2Learn...),

- *L'offre de contenus* : Domaine des entreprises qui développent et éditent les contenus standards ou personnalisés (Auralog, Pearson Education, Thomson Learning...),

- *L'offre de services* : Sociétés de conseils, de maintenance et de services : création de sites web, tutorat, certification, outils d'évaluations..(GlobalKnowledge, Learn2.com, Smartforce...).

Le web est une source intarissable d'informations sur le e-Learning. De nombreux sites, à vocations différentes, sont présents. [6]

Des sites spécialistes permettent de se renseigner sur les technologies existantes pour le e-Learning. Ils sont nombreux et beaucoup en français. L'un des plus importants est celui de la Commission Européenne qui propose plusieurs rubriques afin de rassembler tous les projets en cours ou réalisés au sein de l'Union Européenne.

Il y a des sites commerciaux, qui ne diffèrent généralement que par l'interface graphique ou encore par le nombre ou la qualité des cours. Ces sites utilisent le terme e-Learning mais ne proposent que de simples cours aux formats pdf ou.doc : c'est à l'utilisateur de choisir. On parle souvent aussi de e-formation. Les technologies pour ce genre de site ne sont donc pas très adaptatives.

Lors de la conférence TICE (Technologie de l'Information et de la Communication pour l'Enseignement) qui a eu lieu en novembre 2004, de nombreux travaux concernant des Universités en ligne ont été présentés. Grâce au e-Learning, les étudiants peuvent, suivant les Universités, accéder à un certain nombre de cours pour pouvoir passer les examens de fin d'année avec les autres étudiants de la même section. Cela permet à des étudiants qui sont dans l'incapacité de se déplacer ou trop éloigne, mais qui souhaitent suivre les cours, de pouvoir le faire à leur rythme et même de pouvoir contacter l'enseignant pour poser des questions. Ce système est bien implanté dans certaines universités et, pour certains diplômes, les cours ne sont accessibles qu'en ligne.

3.7 Les objets pédagogiques

Un objet pédagogique, ou Learning Object (LO), peut être défini [7] comme "toute entité, sur un support numérique ou non, pouvant être utilisée pour l'apprentissage, l'enseignement ou la formation". Notant que "les objets pédagogiques peuvent être, par exemple, des transparents, des notes de cours, des pages Web, des logiciels de simulation, des programmes d'enseignement, des objectifs pédagogiques, etc."

Un LO peut prendre la forme de tout type d'élément intervenant dans l'apprentissage ; il peut faire référence à des directives pour aider l'apprenant dans son cours ou à du suivi de l'apprenant.

3.7.1 Le curriculum

Le curriculum [7] peut être vu comme "l'ensemble structuré des expériences d'enseignement et d'apprentissage (objectifs de contenu, objectifs d'habilité, objectifs spécifiques, cheminements ramifiés et règles de progression, matériels didactiques, activités d'enseignement et d'apprentissage, etc.) planifiées et offertes sous la direction d'une institution scolaire en vue d'atteindre des buts éducatifs prédéterminés.". Il est défini formellement comme "une représentation structurée de la matière à enseigner en terme de capacité (capabilité) au sens de Gagné, d'objectifs dont la réalisation contribue à l'acquisition de capacités et de ressources didactiques (exercices, problèmes, démonstrations, vidéos, simulations, etc.). Tous ces éléments sont organisés dans des structures de connaissances destinées à soutenir l'enseignement d'une matière."

Dans cette définition :

- Une *capacité* est une connaissance, acquise ou développée, permettant à une personne de réussir dans l'exercice d'une activité physique ou intellectuelle.
- Un *objectif d'enseignement* est une description d'un ensemble de comportements, ou performances, qu'un apprenant doit être capable de démontrer suite à un apprentissage. Par exemple : analyse, synthèse, évaluation, acquisition, compréhension, application, etc.
- Une *ressource didactique* est un moyen tactique (exercice, problème, test, simulation, etc.), utilisé par le système d'enseignement pour supporter l'apprentissage d'un apprenant.

Il existe trois grandes catégories de ressources [7]:

- *Les ressources intelligentes* sont des ressources didactiques portant sur une activité précise à réaliser par un étudiant. Elles sont généralement assorties d'un modèle de connaissances leur permettant d'être implémentées à n'importe quel niveau du STI pour supporter l'interaction avec l'étudiant et suivre la trace du raisonnement de ce dernier.
- *Les ressources tutorielles* sont capables de gérer l'interaction avec un étudiant durant une session tutorielle. Un certain nombre d'actions didactiques lui sont déléguées par le tuteur laissant ainsi le contrôle à la ressource tutorielle.
- *Les ressources banales* sont des ressources qui jouent plutôt un rôle passif dans le processus d'apprentissage dans la mesure où elles sont incapables de gérer l'interaction avec l'étudiant qui les utilise et ne peuvent pas comporter une évaluation sur les actions de l'étudiant contrairement aux ressources tutorielles et intelligentes.

3.7.2 Le cours

Le cours peut être défini comme un document structuré selon trois composants [7]:

- **Le graphe de cours :** cette partie comporte un ensemble d'objectifs retenus pour le cours considéré et un ensemble de liens vers des ressources pédagogiques définies dans le curriculum.

- **La partie structurelle :** cette partie décrit de manière structurée et hiérarchique l'ordonnancement des objectifs à réaliser pour atteindre les objectifs terminaux définis pour le cours.

- **La partie descriptive :** cette partie comprend :
  - Le titre du cours,
  - Sa description,
  - Une finalité (but du cours),

- Un ensemble d'objectifs dans lesquels l'enseignant exprime son intention pédagogique : *acquisition*, *compréhension*, *application*, *analyse*, *synthèse* et *évaluation*.

- Un ensemble de thèmes ou sous-thèmes correspondant à un élément du contenu de la matière à enseigner. Un ou plusieurs objectifs spécifiques et opérationnels peuvent être définis en fonction des capacités que l'on veut faire acquérir à l'étudiant.

Un cours est défini [7] "*soit par des objectifs d'enseignement (ou d'apprentissage) ayant une finalité précise, soit par un ensemble de connaissances que l'étudiant doit acquérir*"

Le cours est l'ensemble des ressources, informations et connaissances sélectionnées pour représenter une formation particulière ou un savoir.

Les moyens de communication et d'interaction entre apprenants et formateurs ne cessent de progresser : chaque apprenant peut dialoguer avec son tuteur et ses pairs en utilisant des plateformes pédagogiques.

3.8 Les plates-formes pour le e-Learning

Une plate-forme pédagogique est un logiciel qui assiste la conduite des formations ouvertes et à distance. Elle est basée sur des techniques de travail collaboratif et regroupe les outils nécessaires aux trois principaux acteurs de la formation : apprenant, tuteur, administrateur. Elle fournit à chaque acteur un dispositif qui a pour première finalité l'accès à distance au contenu pédagogique, l'auto-apprentissage, l'auto-évaluation et le télé-tutorat via l'utilisation des moyens de travail et de communication à plusieurs : visioconférence, e-mail, forums, chats, annotations, tableaux blancs partagés, etc [18]. Le but est donc de combler la perte de cohésion et de stimulation de la salle que peut sentir l'apprenant devant sa machine.

L'usage de ces systèmes est relativement standard, le tuteur crée des parcours de formation type, incorpore des ressources pédagogiques multimédias et de suivi des activités des apprenants. L'apprenant, peut consulter en ligne ou télécharger les contenus pédagogiques qui lui sont recommandés, effectuer des exercices, s'auto-évaluer et transmettre des travaux à son tuteur pour les corriger. La communication entre apprenant et tuteur peut être individuelle ou en groupe. Il est possible de créer des thèmes de discussion

et collaborer à des travaux communs en utilisant des moyens de travail et de communication à plusieurs. L'administrateur, de son côté, assure l'installation et la maintenance du système, gère les droits d'accès, crée des liens vers d'autres systèmes et ressources externes. Ainsi, une plate-forme peut comporter des fonctionnalités relatives à la gestion des compétences, à la gestion des ressources pédagogiques, à la gestion de la qualité de la formation, etc.

Depuis quelques années, de nombreuses initiatives visant à constituer des ensembles de ressources pédagogiques, dans le but de les partager et de les réutiliser, ont vu le jour. L'alimentation de ces ensembles est souvent basée sur un réseau de contributeurs, qui en contrepartie de leur contribution peuvent bénéficier de l'ensemble des ressources. Différents types de systèmes sont proposés [7]:

- **Les Learning Management Systems (LMS)** : ce sont des systèmes informatiques de gestion de formation. Ces systèmes assurent le déploiement des contenus éducatifs aux apprenants. Ils gèrent l'accès des usagers aux différents modules de formation, permettent aux pédagogues de suivre l'avancement des apprenants des modules (temps écoulé, historique des réponses, etc.) et de gérer l'orientation de ces apprenants d'un module à un autre. comme exemples de LMS nous citons: WebCT, Ingenium, WBT Manager, etc.
- **Les Learning Content Management Systems (LCMS) :** Des systèmes informatiques de gestion de contenu de formation. Ces systèmes permettent à des experts d'un domaine, à des développeurs, de coopérer (via le Web) pour créer des contenus éducatifs (ou Learning Contents) aussi réutilisables que possible. ASPEN est un exemple d'environnement entrant dans cette catégorie.

Avec l'évolution des techniques, des infrastructures de réseau et des normes, le nombre de plates-formes et environnements de formation a augmenté de manière significative, et ce de façon assez rapide. A chaque contexte de formation peut correspondre un ensemble de fonctionnalités adaptées et donc une plate-forme potentielle. Cependant, dans un choix raisonné d'une plateforme, il semble indispensable de bien définir un cahier des charges pour le projet de la formation envisagée et ses objectifs. C'est pour cela que plusieurs tentatives de standardisation de ces environnements ont été proposées.

3.9 Normes et standards dans le domaine éducatif

L'intérêt du e-Learning est de proposer un ensemble de cours aux apprenants mais aussi de faciliter la mise en place des cours pour les enseignants. Si un enseignant souhaite proposer ses cours pour plusieurs plates-formes, l'utilisation d'une norme permet à l'enseignant de n'écrire qu'une seule fois son cours. Avec l'émergence des Technologies de l'Information et de la Communication (TIC), des normes et standards dans le domaine des technologies éducatives sont en développement.

Avant d'étudier les principaux standards et normes, nous donnons deux définitions afin de les distinguer [7] :

Norme: c'est comme un ensemble de règles fonctionnelles ou de prescriptions techniques relatives à des produits, à des activités ou à leurs résultats, établies par consensus de spécialistes et consignées dans un document produit par un organisme, national ou international, reconnu dans le domaine de la normalisation. Par exemple, ISO, CEN, AFNOR.

Standard : c'est un ensemble des règles et des prescriptions techniques établies pour une organisation et qui servent à fixer les caractéristiques permettant de définir un élément de matériel ou de construction utilisé pour un projet donné. Par exemple, le cas des recommandations du W3C, de l' IEEE, etc.

Dans cette section, on présente les besoins en normalisation dans le domaine éducatif avant de décrire les différentes tentatives de normalisation développées.

3.9.1 Besoins en normalisation dans le domaine éducatif

L'archivage et la production de ressources numériques dans les systèmes éducatifs se traduit, généralement, par un jeu de méta-données [7]. Ce jeu s'exprime souvent par une fiche de bibliothèque électronique regroupant l'ensemble des informations pertinentes caractérisant le document : titre, auteurs, format, date d'édition, etc. Le développement à grande échelle des platesformes d'apprentissage à distance dans des environnements techniques différents nécessite d'avoir [7]:

- un format standard, universel, pour ces méta-données dans le but de faciliter l'échange et l'accessibilité,
- une définition fine pour les ressources pédagogiques.

Le principal objectif de la normalisation est de réaliser l'interopérabilité entre les composants d'une infrastructure ou d'un système, afin de généraliser l'application d'outils dans des contextes différents de ceux prévus dans le développement originel. Par exemple, les protocoles HTTP (HyperText Transfer Protocol), URL (Uniform Resource Locator) et HTML (HyperText Mark-up Language) normalisent le protocole pour la demande, l'identification et la structure des documents dans le WWW (World Wide Web). Un autre exemple bien illustratif est la normalisation de la taille de papier en A4 (une norme allemande DIN).

Dans le domaine de la formation en ligne, la normalisation répond à cinq objectifs [7]:

- **L'accessibilité :** permet de faciliter la recherche, l'identification, l'accès aux contenus et composants de la formation.
- **La réutilisabilité :** permet de réutiliser les mêmes objets pédagogiques à différentes fins, dans différentes applications, dans différents contextes et via différents modes d'accès.
- **L'adaptabilité :** permet la modularisation des contenus et des composants pour mieux répondre aux besoins des utilisateurs.
- **L'interopérabilité :** permet les échanges entre composants logiciels grâce à des interfaces communes.
- **La durabilité :** permet d'éviter le développement à nouveau des formats de contenus et des composants dans le cas de changement de support logiciel ou matériel.

3.9.2 Quelques standards et normes

- **LOM (Learning Object Metadata)**

LOM [6] est un standard pour les méta-donnees de caractérisation (ce n'est pas une norme). Il décrit l'objet pédagogique selon neuf catégories [7]. Dans chacune d'entre elles, plusieurs éléments peuvent être répétés (parfois de façon récursive). LOM est le schéma de méta-donnees le plus détaillé et offre un ensemble de vocabulaire de référence. Il s'agit d'un fichier XML qui décrit les caractéristiques du document (dans le cas du e-Learning, il décrit le contenu d'un cours ou d'un exercice).

Du coté utilisateur, il permet de retrouver et d'échanger des ressources pédagogiques. Du coté producteur, il permet de partager l'information dans un contexte où les ressources sont nombreuses et leurs productions coûteuses, de réutiliser les ressources ou leurs composants et enfin d'être interopérable avec des systèmes de Gestion de Formation.

- **AICC (Aviation Industry Computer based training Committee)**

En 1988, des compagnies aériennes, des constructeurs aéronautiques, des producteurs d'enseignements assistés par ordinateur fondent l'Aviation Industry CBT (Computer Based Training [18]) Committee (AICC), se réunissent pour définir des spécifications techniques communes pour les produits d'enseignement assisté par ordinateur qu'ils utilisent.

AICC [6] a progressivement été étendu à l'ensemble des problématiques liées à la formation électronique. La compatibilité avec cette norme permet notamment l'interopérabilité entre plates-formes et contenus hétérogènes offrant ainsi des possibilités d'évolution et d'enrichissement accrus. En ce qui concerne le e-Learning, l'AICC définit la structure des contenus, les modes de communication entre la plate-forme de formation et les contenus pédagogiques.

- **SCORM (Shareable Courseware Object Reference Model)**

C’est un programme du département de la défense des Etats-Unis et du bureau des sciences et technologies de la Maison Blanche visant à mettre en œuvre à grande échelle des systèmes d'e-formation [18]. SCORM [6] est une norme qui s'inspire d'AICC. Il met en place les règles d'un modèle de gestion de l'apprentissage par l'utilisation du Web. Cette initiative doit permettre aux enseignants d'intégrer les cours qu'ils créent dans d'autres applications, sous différentes plates-formes. Le contenu doit être indépendant des contraintes de mise en forme de façon à autoriser son intégration dans différentes applications. Le contenu devra aussi utiliser des interfaces et des données normalisées. Le SCORM comprend un Format de Structure de Cours (Course Structure Format) basé sur le langage XML et qui permet de transférer plus facilement des contenus en définissant les éléments, la structure et les références externes.

- **IMS (Instructional Managment System)**

IMS est une norme d'indexation en métadonnées [18]. Il inclut désormais la façon de présenter les contenus (mise en forme de cours), les profils des usagers, les questions et

les tests. L'IMS a pour objectifs principaux de définir des spécifications techniques pour l'interopérabilité des applications et services de l'éducation distribuée et de supporter l'incorporation des spécifications dans les technologies du Web. Ces spécifications doivent répondre à des principes de base : l'interopérabilité, l'accessibilité, la réutilisation, la pérennité, l'indépendance et la portabilité [6].

3.10 Description des objets pédagogiques

3.10.1 Les méta-données [7]

Le terme "méta" vient du grec et dénote quelque chose de nature plus élevée ou plus fondamentale. La première définition attribuée au terme "méta-donnée" est issue des recherches sur les bases de données "les méta-données sont des données sur les données". Depuis, la complexité et la variété des systèmes d'information ayant accru, les méta-données constituent des structures et descriptions émises à un niveau d'abstraction supérieur (méta) et relatives à un niveau inférieur (ou référence). On utilise généralement les méta-données pour parler d'information descriptive à propos de ressources du Web. Toutefois les méta-données peuvent répondre à de nombreux objectifs, que ce soit l'identification d'une ressource satisfaisant un besoin particulier d'information, l'évaluation de sa pertinence ou enfin pour garder la trace des caractéristiques d'une ressource à des fins d'entretien ou d'utilisation à long terme. De nos jours, différentes communautés d'usagers comblent de tels besoins en utilisant une grande variété de normes de méta-données.

Afin de décrire une ressource on utilise des méta-données. Une notice contenant un ensemble d'attributs, ou éléments, nécessaires pour décrire la ressource en question est établie. Par exemple, dans les bibliothèques, on utilise un ensemble de notices de méta-données comprenant des éléments spécifiques tels que : auteur, titre, date de création ou de publication, sujet et cote, afin de retrouver un livre ou un document sur les tablettes.

Le lien entre une notice de méta-données et la ressource qu'elle décrit peut être fait de deux façons :

- Les méta-données peuvent être contenues dans une notice séparée du document, comme c'est le cas pour une notice dans un catalogue de bibliothèque.
- Les méta-données peuvent être intégrées dans la ressource elle-même.

C'est avec l'augmentation de l'édition électronique et des bibliothèques numériques que l'intérêt mondial pour les pratiques et standards de méta-données a véritablement explosé.

L'adoption à grande échelle de normes descriptives et de nouvelles pratiques pour les ressources électroniques va améliorer la possibilité de trouver des ressources pertinentes sur Internet. A l'instar de Weibel et Lagoze, deux leaders dans le développement des méta-données: "*L'association de méta-données descriptives standardisées avec des objets en réseau offre un potentiel d'amélioration substantiel des possibilités de découverte de ressources: en permettant des recherches basées sur des champs (e.g., auteur, titre), en permettant l'indexation d'objets non-textuels et en permettant l'accès à un contenu de substitution, ce qui est différent de l'accès au contenu de la ressource elle même*" [7]. C'est ce besoin en méta-données descriptives standardisées que plusieurs modèles de métadonnées veulent combler.

3.10.2 Quelques modèles de méta-données existants

3.10.2.1 Dublin Core (DC)

Crée en 1995, DC rassemble entre autres des bibliothécaires, des documentalistes, des informaticiens. Il définit des méta-données génériques et développe des outils pour implémenter des méta-données dans les ressources. La norme du Dublin Core comprend 15 éléments de données descriptives relatifs aux ressources d'information, visant à appuyer la découverte de ressources dans les applications accessibles par le Web. Elle représente le résultat d'une série d'ateliers internationaux entre experts. Au cours de ces ateliers, un large consensus a été atteint, relativement à la définition des ressources, aux normes d'encodage, à la recherche documentaire et à toute une gamme de sujets. Dublin Core constitue la principale solution de rechange pour la description de ressources d'applications telles que les passerelles de sujet et les nombreuses collections numérisées. Il représente également la base d'un vaste réseau sémantique interopérable s'appuyant sur un ensemble d'éléments de base pouvant être utilisé à grande échelle. La norme Dublin Core n'empêche pas l'utilisation d'autres éléments nécessaires à des mises en œuvre locales.

Plus récemment, des groupes de travail sectoriels ont été mis en place, en particulier le Dublin Core Education pour le domaine de l'éducation.

3.10.2.2 Learning Object Metadata (LOM)

Actuellement le standard LOM (*Learning Object Metadata*) spécifie la syntaxe et la sémantique des méta-données pédagogiques et définit les attributs nécessaires pour une description complète des ressources pédagogiques. Il consiste en un ensemble minimal de caractéristiques indispensables pour gérer les objets pédagogiques (*Learning Object*).

3.11 Travaux sur le e-Learning

3.11.1 Utilisation de l'algorithme des fourmis

Les premiers processus, très limités, d'interaction entre l'apprenant et le système sont appelés systèmes tutoriels. Ils ont évolué vers des systèmes dit adaptatifs capables de générer du matériel didactique selon le comportement global de l'apprenant. Il s'agit des systèmes dits intelligents (STI) qui ont réellement permis, grâce à des techniques d'intelligence artificielle, de créer un processus complètement interactif adapté à l'apprentissage. Donc il est important de parler des recherches concernant le e-Learning adaptatif. Les premières adaptations proposées sont la possibilité pour l'apprenant de pouvoir visualiser ses cours en fonction de ses préférences ou encore ne pas refaire un exercice déjà effectué auparavant.

L'autonomie de l'apprenant est un des objectifs du e-Learning. Sterenn AUBERT dans [6] a voulu, avant de réfléchir à la technique employée pour gérer l'adaptation des cours, modéliser son système de e-Learning adaptatif en utilisant UML. Cette modélisation UML comprend deux types de diagrammes, le diagramme de cas d'utilisation qui montre les différents acteurs de ce système et les rôles qu'ils peuvent tenir, et les diagrammes de classes (point de vue orienté objet) qui indiquent les relations entre les différentes classes. Après il vient de définir la partie UML du système. Il a réfléchi aux moyens d'adapter les cours et aux paramètres qui vont rentrer en jeu. L'intelligence artificielle répond à ce genre de problématique. Dans ce domaine, les recherches sont encore récentes et les résultats encore incertains. Il faut donc savoir choisir l'algorithme qui correspond le mieux aux besoins et étudier les algorithmes génétiques qui existent afin de trouver le meilleur pour l'adaptation des cours d'une plate-forme e-Learning. Il a choisi l'utilisation de l'algorithme des fourmis. Les choix importants que doivent être fait, sont de définir avec plus de précision les caractéristiques d'un profil d'un apprenant, ainsi que celles d'un cours et les paramètres de la fonction de fitness. Où les nœuds du graphe correspondaient aux pages HTML du cours et les arcs les liens qui sont possibles vers d'autres pages HTML. Ceci est une bonne conception, cependant il a envisagé une autre modélisation. En effet, les cours écris par les enseignants-auteurs sont en XML, or il s'avère qu'un document XML peut être représenté par un graphe. Il est donc intéressant d'exploiter ce graphe plutôt que de travailler sur des pages HTML qui restent statiques. L'avantage d'exploiter un fichier XML

est que le cours obtenu est dynamique et exploite réellement toutes les connaissances et les données récoltées lors de l'apprentissage. Le travail avec les algorithmes génétiques reste rencontre un problème principal est celui de comment choisir la bonne fonction objectif ou de sélection [6].

3.11.2 Baghera

C'est une plate-forme d'enseignement et d'apprentissage à distance, elle propose les bases d'environnements informatiques pour l'apprentissage humain (EIAH) vu comme une communauté éducative composée d'agents humains et non-humains qui coopèrent et travaillent ensemble pour l'éducation des élèves [26]. Cependant, il est important de se souvenir que l'objectif n'est pas essentiellement de développer une architecture distribuée, ce qui a déjà été réalisé dans le cadre d'autres travaux, mais d'apporter par la voie de la modélisation d'agents dédiés, une solution à l'obstacle que constitue jusqu'ici l'absence d'un modèle didactique et/ou pédagogique universel permettant de guider les apprentissages sous un contrôle centralisé et déterminé a priori. Les premiers tests d'usage de la plate-forme ont montré son adéquation à la spécification initiale des comportements individuels et coopératifs des agents. L'analyse de ces tests a cependant révélé une difficulté qui doit être levée : la communication entre les agents demande la prise en compte d'une grande variété de formes et de niveaux de représentation des connaissances directement liée à la multiplicité de leurs rôles (fonctions). Cette gestion de la communication à niveau "connaissance" est un point de passage obligé pour aller vers le système dont le groupe de travail cherche à préciser les principes de conception et dont la fonctionnalité globale "éducante" sera le résultat des interactions et de la coopération d'une société d'agents spécialisés selon des types de compétences didactiques ou pédagogiques clairement identifiés.

3.11.3 SIGFAD

SIGFAD [4] est un système multi-agents destiné à être couplé aux plates-formes informatiques de formation à distance. SIGFAD implémente des fonctionnalités permettant aux tuteurs de maintenir les groupes, les dynamiser et de bien conduire la session de formation à distance (FAD). il s'appuie sur l'architecture BDI (Beliefs-Desires-Intentions) pour mesurer un certain nombre de critères sur les groupes et les individus engagés dans une session de FAD. Ces critères, bien que construits à posteriori, peuvent être calculés au

fil d'une session de FAD et mis à la disposition des utilisateurs en temps réel. La recherche concerne la formation à distance (FAD), plus spécifiquement le soutien des interactions dans des groupes de taille restreinte (8 à 15 personnes). Ceci nous a donné l'occasion d'observer plusieurs phénomènes, d'identifier des problématiques spécifiques à ce mode de formation et de recueillir un corpus important de données qu'on permettent aujourd'hui de définir et de répondre à plusieurs questions de recherche. Le système multi-agents (SMA) est destiné à être couplé à des plates-formes de formation à distance afin d'y implémenter des fonctionnalités permettant d'apprécier l'état du groupe : personnes présentes, absentes, dormantes ; l'état du groupe en fonction du pourcentage de personnes actives ; la productivité d'un utilisateur donné ; le niveau de réalisation d'une activité donnée.

3.12 Les freins au développement

Malgré des offres multiples, les entreprises restent prudentes quant à la mise en place d'un projet de e-Learning pour lequel les contraintes techniques restent lourdes et les freins culturels persistants. Trois types d'obstacles existent [21] :

- Des freins propres à une industrie récente en cours de structuration : fragilité financière des start-up, manque de standards et de normes, découragement des investisseurs.
- Des freins liés aux caractéristiques de l'offre : manque de qualité des contenus, compétence des prestataires, difficultés de gestion des droits liés aux contenus, inadéquation de l'offre par rapport aux besoins "métiers" des entreprises.
- Des freins techniques : problèmes de compatibilité entre les systèmes d'information internes et externes auquel la formation en ligne fait appel, problèmes internes de sécurité.

Nonobstant les coûts et les obstacles, l'e-Learning devrait bénéficier à terme de facteurs de croissance puissants : augmentation du taux de pénétration d'internet, développement des réseaux haut débit s'accompagnant de l'augmentation de la bande passante, développement des solutions personnalisables, multiplications des offres intégrées, réduction des coûts de formation.

3.13 Conclusion

Dans ce chapitre, nous avons présenté l'évolution des systèmes du e-Learning et ses différents typologies. Nous avons remarqué que le marché est plein d'outils, des technologies pour satisfaire les besoins d'évolution de e-Learning. C'est pour cela on peut trouver plusieurs plates-formes d'apprentissage, des différents standards et normes. Nous avons aussi étudié l'utilisation des méta-données et des banques de ressources pédagogiques pour l'organisation des ressources. Plusieurs travaux sont faits en exploitant les différents moyens informatiques pour le e-Learning mais ils restent rencontre des problèmes.

Un examen des fonctionnalités des systèmes informatiques dédiés [4] au e-Learning et particulièrement la formation à distance (FAD) montre que finalement, ces systèmes sont ouverts, complexes, évolutifs. Un système ouvert est un système dont la structure peut changer dynamiquement. Ses composants ne sont pas connus à l'avance, changent au cours du temps, et sont essentiellement hétérogènes (en raison du fait qu'ils sont mis au point par différentes personnes, à différentes périodes, utilisant des techniques et outils de développement différents). Le système ouvert le plus connu est Internet. La FAD médiatisée par ordinateur s'appuie essentiellement sur Internet et exige le développement de technologies y relatives ; il est évident que cette exigence induit de la part des informaticiens la mise en place de systèmes ouverts. La FAD parce qu'elle se trouve au carrefour de plusieurs disciplines de sciences humaines (psychologie, sciences de l'éducation, sociologie, etc.) et des sciences de l'ingénieur (informatique, télécommunications) pose des problèmes complexes, diversifiés, interdépendants et difficilement prévisibles. Les outils adaptés pour faire face à la complexité dans le développement logiciel sont la modularité et l'abstraction. Le paradigme d'agent est un bon outil de modularisation, il permet de résoudre les problèmes posés par la FAD en développant des modules spécialisés (en termes de représentation et de type de problème résolu) dans un aspect particulier. L'interdépendance des problèmes de FAD est surmontée grâce à la coopération entre les agents mis en œuvre.

La FAD en ligne exige de prendre en compte l'hétérogénéité des sources d'informations, des systèmes d'information et des centres de régulation des activités (au niveau de l'apprenant, du tuteur, du formateur, du concepteur, du coordonnateur de la

formation). L'ouverture est liée à l'ajout/retrait de sous-systèmes (logiciels, matériels, acteurs) sans stopper ou réinitialiser le système global. L'évolutivité a pour objectif de s'adapter aux changements continuels de l'environnement dans lequel s'inscrit la FAD. La réactivité cela consiste à se configurer en fonction de l'évolution des activités à mettre en œuvre, en fonction des compétences disponibles et des objectifs d'apprentissage. La coopération entre les différents sous-systèmes est pour produire des contenus pédagogiques, administrer, assister, évaluer les apprenants. La coordination des différentes plates-formes de FAD qui ne manqueront pas de se regrouper en environnements de campus virtuel. Dans ce contexte, l'informatique sur laquelle s'appuie la FAD doit intégrer de nouveaux concepts, en s'appuyant sur des technologies fortement liées à Internet, afin de proposer des architectures logicielles qui permettent la coopération de multiples applications hétérogènes et distribuées. Dans le chapitre suivant, nous présenterons notre modèle pour le e-Learning à base du paradigme agent.

Chapitre 4

Modélisation du système

4.1 Introduction

Il existe de nombreux travaux de recherche ainsi que des pratiques éducatives issues des expérimentations pédagogiques et informatiques menées en formation à distance et, plus particulièrement, en formation en ligne. Ces travaux s'intéressent à la conception et la réalisation de systèmes informatiques permettant d'assister l'apprenant dans sa formation par exemple BAGHERA [26]. C'est un Environnement Informatique d'Apprentissage Humain (EIAH) « distant » et multi-agent pour l'apprentissage de la démonstration en géométrie. Il s'agit d'un outil de démonstration automatique est accessible aux élèves pour les aider dans leur travail de géométrie et ils peuvent interagir avec d'autres élèves ou enseignants connectés au moyen d'une fenêtre de dialogue. Les enseignants peuvent connaître l'état d'avancement des travaux des élèves ; une éventuelle intervention de l'enseignant auprès d'un élève est ainsi facilitée. SIGFAD [4] implante des fonctionnalités permettant aux tuteurs de maintenir les groupes, les dynamiser et de bien conduire la session de formation à distance.

Ces plates-formes actuelles de télé-formation ne disposent pas d'outils permettant de faire un suivi individualisé de l'apprenant. Ce suivi est indispensable vu le nombre important des apprenants qui abandonnent la formation.

Notre contribution consiste à proposer un système adaptatif permettant d'assurer un suivi automatique et continu de l'apprenant. Le système est en outre ouvert et générique pour supporter n'importe quel objet d'apprentissage.

Dans le cadre d'une autoformation, l'apprenant doit avoir la liberté et le choix du parcours à suivre pour assimiler l'objet de l'apprentissage via une approche constructiviste du savoir au lieu d'une consommation massive des documents. Cette liberté doit être assistée par un suivi régulier et continu. Traditionnellement, le suivi est assuré par un enseignant. L'objectif principal de notre environnement de formation ouvert et distant est

de proposer une alternative à l'enseignement présentiel, par conséquent l'outil informatique doit intégrer l'assistance et le suivi parmi ses fonctionnalités.

Nous débutons ce chapitre par une argumentation de notre utilisation du S.M.A. Nous présentons par la suite la phase d'analyse et l'expression des besoins pour obtenir une modélisation de l'application sous forme de système multi-agents. Cette phase contient une description générale des fonctionnalités que nous souhaitons voir figurer dans le système d'enseignement et d'apprentissage à distance. Nous parlerons dans la suite de ce chapitre de la modélisation de notre système. La conclusion est présentée dans la dernière section.

4.2 Pourquoi une modélisation par agent

Lorsqu'il s'agit de concevoir des systèmes informatiques distribués qui manipulent des connaissances hétérogènes, la technologie « agent » se révèle bien adaptée. En effet, les systèmes multi-agent se caractérisent par le partage ou la distribution de la connaissance et sont capables de faire coopérer un ensemble d'agents et de coordonner leurs actions dans un environnement pour l'accomplissement d'un but commun. Ces deux aspects correspondent bien aux besoins de résolution de problèmes et de coopération des différents acteurs d'un EIAH, qu'ils soient humains ou hybrides. Un autre aspect prometteur est celui des agents dits « assistants » dont le rôle est de faciliter l'usage du système, grâce notamment à la personnalisation des offres et à la recommandation. La nécessité d'un suivi individualisé de l'apprenant pour prévoir et diminuer le pourcentage des abandons, nous impose de choisir une solution souple et adaptative.

Répondre aux besoins des didacticiens, consiste à concevoir et réaliser un système informatique qui doit revêtir des caractéristiques suivantes :

• ouvert, la structure peut changer dynamiquement à cause du web.

• autonome, un tel système doit simuler le rôle de l'enseignant et initier ainsi l'apprentissage.

• adaptatif, le système est destiné à un apprentissage individualisé, donc il faut prendre en compte les différents profils des apprenants.

Donc le système est complexe. Cette complexité, qui se traduit aussi par le nombre important de données à manipuler et la dynamique de la situation à traiter, nous a conduit à choisir les systèmes multi-agent pour la modélisation.

4.3 Analyse et Expression des besoins

Nous présentons dans cette section une description générale des fonctionnalités que nous souhaitons voir figurer dans le système d'enseignement et d'apprentissage à distance.

4.3.1 Présentation générale

Il s'agit de concevoir un système d'enseignement et d'apprentissage. Cette application informatique doit permettre à des apprenants d'apprendre ce qu'ils leur intéressent comme formation de façon autonome tout en ayant recours, si besoin est, à l'aide d'un enseignant en respectant les lois de l'administrateur.

Ce système s'adresse donc à trois types d'utilisateurs : des apprenants, des enseignants et l'administrateur qui accèdent à l'application au moyen d'un équipement terminal banalisé (ordinateur individuel) connecté à l'internet, en spécifiant un nom d'utilisateur et un mot de passe.

On distingue deux types d'utilisateur enseignant : le tuteur et l'auteur. Le premier organise les cours (donne le programme à suivre dans l'étude), l'autre prépare et rédige les cours en suivant l'organisation de tuteur.

L'administrateur fait la gestion générale du système éducatif.

Notre système modélise le e-Learning asynchrone. Par conséquent, chaque type d'utilisateurs est défini par ses ressources dans l'application et non pas par son contact présentiel avec les autres.

Notre système permet :

- ✓ Plusieurs formations en même temps.
- ✓ L'ajout des nouvelles formations.
- ✓ Une formation peut suivre le système modulaire.
- ✓ La formation peut être un seul module ou avoir plusieurs modules.
- ✓ Avoir une formation à plusieurs modules avec des coefficients différentes.
- ✓ Variétés des programmes à suivre pour chaque module ; c'est à dire, un seul module peut avoir plusieurs organisations du cours.
- ✓ Chaque organisation de cours peut avoir plus d'une rédaction.
- ✓ Le suivi individuel de chaque apprenant.
- ✓ L'évaluation individuel de chaque apprenant.
- ✓ L'évaluation des auteurs et des tuteurs.

- ✓ La possibilité de diminuer la durée de formation.
- ✓ Appliquer les décisions de l'administrateur dans des contextes prédéfinis (concernant les auteurs, les tuteurs,…).

4.3.2 Interface du système

Lorsqu'un utilisateur est connecté, une interface graphique lui donne des explications d'accès à l'application. Elle lui permet de connaître :

- Les formations disponibles.
- Les modules constituant une formation donnée.
- Les coefficients des différents modules.
- La durée de formation de chaque module .
- Les avis des nouvelles formations pour attirer les enseignants afin de faire les cours.
- Les inscriptions des nouveaux utilisateurs.
- Authentification des anciens utilisateurs (nom d'utilisateur, mot de passe) pour la sécurité d'accès.
- Les énoncés des différentes nouvelles décisions de l'administrateur.

Grace aux différentes changements dans les composants du système éducatif (formations, modules , auteurs, tuteurs,…), les fonctions de l'interface doit être effectuées en temps réel.

4.3.3 Utilisateur apprenant

Notre système offre à l'apprenant les fonctionnalités suivantes :

- Pour chaque module de la formation choisie, un nouveau apprenant a la liberté de choisir l'organisation qu'il veut suivre.
- Trouver des explications supplémentaires à certains concepts. Si les explications données ne satisfont pas les besoins de l'apprenant, il a la possibilité de demander d'autres.
- Il peut demander de faire un test (examen) à n'importe quel point du cours pour tester son niveau de compréhension personnel.
- Des examens ou des tests selon sa progression dans le cours.

Notons que nous distinguons entre deux types d'apprenants :

1. Des apprenants ayant l'objectif de formation pour eux même. Ils sont obligés de respecter le temps ou le délais de formation décidé par l'administrateur du système éducatif sans aucune remarque.
2. Des apprenants qui sont des employeurs dans des entreprises. Ils ont pour objectif de savoir les nouvelles dans leurs domaines. A cause des compétitions entre les entreprises, le facteurs de temps est très intéressants. Donc, ce type des apprenants peut négocier avec le système pour qu'ils soient en accord à une durée spécifique. Cette durée est liée d'un coté aux capacités de l'apprenant et les contraintes de son entreprise, de l'autre coté, aux possibilités fournies par le système et son état.

Après qu'un nouveau apprenant a choisit l'organisation du cours à suivre dans chaque module, le système classifie l'apprenant sous la supervision d'un des auteurs rédigeant le cours de cette organisation. Le système sélectionne l'auteur ayant le moindre nombre d'apprenants suivant ses cours. Dans le cas de plusieurs auteurs qui ont le même nombre, la sélection sera aléatoire. De cette façon, les auteurs de chaque organisation participent effectivement dans le système éducatif. Ainsi, ce mécanisme permet d'évaluer le maximum d'auteurs d'une organisation et par conséquent, l'évaluation de l'organisation elle-même qui implique l'évaluation de son tuteur.

Notons que ce mécanisme n'est pas applicable dans le cas d'un apprenant de type employeur. Car, l'objectif est de satisfaire le besoin de l'apprenant et non pas l'équilibre du système éducatif.

Dans les formations synchrones présentièlles traditionnelles (dans les salles), les apprenants sont sous le contrôle des enseignants soit dans le cours soit dans les examens et les différents tests. L'attention de l'apprenant est attiré par les différentes activités de l'enseignant. Dans le e-Learning et particulièrement la e-formation asynchrone, l'apprenant est sans contrôle à cause de l'absence de l'enseignant.

Pour diminuer ce manque, notre système offre des fonctions à savoir :

- ✓ Si l'apprenant ne progresse pas bien dans le cours, il subira des alertes et des tests surprises.
- ✓ De temps en temps, le système remplace la routine de l'apprenant par des vidéos, des images ou des textes expressifs.

A la fin de la formation de chaque apprenant, ce dernier est dit réussi si les conditions suivantes sont vraies :

- la terminaison du cours de chaque module est réalisée, sinon, il est considéré comme un abandon dans ce module.
- Son résultat final ou sa moyenne générale dépasse ou égal la moyenne éliminatoire précisée par l'administrateur du système.
- Si la formation de l'apprenant suit le système modulaire, il doit obtenir la moyenne éliminatoire ou plus dans chaque module.

4.3.4 Utilisateur tuteur

Le tuteur doit proposer l'organisation des cours du module ou formation choisi. Il doit prendre en compte le volume horaire et le coefficient donnés par l'administrateur du système.

L'organisation du cours contient les éléments qu'il constituent. Chaque élément est associé par un niveau de formation. Un niveau signifie le détail dans le cours dépendamment à un temps donné. Cette décomposition en niveaux a pour but d'adapter la présentation des cours selon le type de l'apprenant, particulièrement pour les employeurs. Il faut que la diminution du temps de formation n'influe pas sur les objectifs.

Nous proposons quatre niveaux de détails. Ils sont de 25%, 50%, 75% et 100% du temps origine de la formation ou du module :

- Niveau 1 : occupe 25% de la durée originale de la formation.
- Niveau 2 : occupe 50% de la durée originale de la formation.
- Niveau 3 : occupe 75% de la durée originale de la formation.
- Niveau 4 : occupe 100% de la durée originale de la formation.

Niveau 4 concerne les apprenants n'ayant pas des pressions de finir la formation rapidement. Donc, les niveaux de diminution de temps de formation sont 25%, 50%, 75%. L'apprenant de niveau i apprend les éléments des cours des niveaux j : $j \leq i$.

En se basant sur l'organisation, l'horaire autorisé et l'importance de module dans la formation (coefficient), le tuteur décide et choisi les niveaux convenables et possibles.

Un cours est défini par ses éléments qui le constituent. Selon le tuteur, l'organisation du cours peut contenir des concepts supplémentaires aidant l'apprenant à bien comprendre le cours.

L'apprenant doit apprendre tous les éléments constituant le cours. Le contraire avec les concepts, ils ne sont pas obligatoires et l'apprenant a la liberté d'étudier n'importe quel concept.

Le tuteur partage la durée spécifiée au module entre les différents composants du cours. Donc, chaque composant a sa durée partielle comme une valeur approximative. Les durées partielles aident les auteurs de ne pas détailler plus. Ainsi, elles sont utilisées pour calculer les durées de formations de module en différents niveaux afin de diminuer la durée totale de la formation pour les employeurs.

Une autre tâche pour le tuteur concerne les examens. Il doit proposer le nombre minimum des examens à faire par l'apprenant. Il doit aussi associer à chaque examen son temps en fonction de durée de formation du module. (exemple : au milieu, au bout, ou 80% de durée de formation,…).

Le système permet au tuteur d'évaluer son organisation à travers l'évaluation des différents auteurs associés à lui.

4.3.5 Utilisateur Auteur

L'auteur doit suivre l'une des organisations proposées pour les cours d'un module ou d'une formation choisi. Il rédige les éléments des cours et les concepts constituant l'organisation suivie.

L'auteur prend en compte le temps précisé pour chaque élément de cours pour ne pas détailler plus. Il peut donner plusieurs explications ou différents exemples associés à un seul concept.

L'auteur doit poser des questions concernant les différents points du cours. Il ne s'occupe pas d'assembler les questions pour composer les examens. Cette tâche est faite par le système.

Le système aide l'auteur d'ajouter/modifier/supprimer des questions. Il lui donne des informations sur les différents résultats obtenus par les apprenants aux différentes questions pour les évaluer. La modification d'une question consiste à supprimer l'ancienne et poser la nouvelle.

Si l'auteur veut faire des changements sur son cours, il doit demander de s'inscrire comme un nouveau auteur. Il peut demander de supprimer l'ancienne rédaction de cours.

4.3.6 Administration du système

L'administration du système a la responsabilité de la gestion des comptes des utilisateurs apprenants et enseignants. Quelque soit le type d'utilisateur, un compte se caractérise par un nom d'utilisateur et un mot de passe pour l'identification et l'autorisation d'accès au système, et par d'autres identificateurs (nom et prénom de l'utilisateur).

Parmi les autres taches effectuées par l'administration du système, nous citons :

- Décider de l'ensemble des formations à autoriser dans le système éducatif ;
    - Ajouter de nouvelles formations.
    - Supprimer une formation existante.
- Donner les différents composants et les attributs définissant une formation ;
    - Définir les composants de chaque formation (modules).
    - Préciser si la formation suit le système modulaire ou non. Dans un système modulaire, la moyenne de chaque module doit être supérieure ou égale une moyenne éliminatoire donnée par l'administrateur. Ce dernier doit poser la moyenne générale à atteindre pour qu'un apprenant soit réussi.
    - Poser les coefficients des modules constituants une formation.
    - Fixer la durée d'apprentissage spécifiée pour chaque module.

La suppression d'un compte existant entraîne la suppression de l'autorisation d'accès correspondante et la suppression des ressources liées à cet compte.

L'administrateur (personne) à l'aide de notre système peut donner des décisions concernant les auteurs et les tuteurs ou des critiques afin d'améliorer les résultats du système éducatif.

Remarque : Notre système ne permet pas de faire des modifications sur la même ressource (question / rédaction du cours/ organisation) pour éviter la complexité. C'est pour cela que nous préférons de supprimer (s'il est nécessaire) l'ancienne et ajouter la nouvelle.

4.3.7 Environnement de travail et accès au système

Un des points clés étant la mobilité, les apprenants et enseignants doivent accéder à l'application avec un équipement informatique « minimal ». Pour répondre à ce besoin, il semble que l'utilisation d'un ordinateur relié au réseau Internet et équipé d'un logiciel de navigation pour accéder au système soit une solution adéquate.

En effet, Internet constitue aujourd'hui le médium de communication le plus développé pour interconnecter des équipements informatiques et l'accès à ce réseau tend à se démocratiser très rapidement. D'autre part, l'utilisation d'un navigateur Web constitue la meilleure alternative si l'on veut faire fonctionner une application en s'affranchissant d'une part du type d'équipement informatique (et donc du système d'exploitation qui conditionne l'utilisation de logiciels), et d'autre part de l'installation de logiciel propriétaire préalablement à toute utilisation.

L'accès à l'application se fait donc par l'intermédiaire d'un ordinateur disposant d'un accès au réseau Internet en utilisant un navigateur dans lequel il faudra spécifier l'adresse du site Web de l'application.

4.4 La modélisation

4.4.1 Caractérisation (indexation) des différents objets de la formation à distance asynchrone

- L'apprenant est caractérisé par :
  - La formation suivie.
  - Le tuteur suivi dans chaque module (organisation).
  - L'auteur suivi dans chaque module (rédaction du cours).
  - La progression dans le cours de chaque module.
  - Les concepts étudiés.
  - Les résultats des différents tests dans chaque module.
  - L'ensemble des concepts qu'il n'a pas satisfait par leurs explications.
  - L'historique des questions constituant ses différents tests.
- Le tuteur est défini dans le système par :
  - La formation choisie.
  - Le module choisi.
  - Son organisation du cours (éléments du cours et les concepts à expliquer).
- L'auteur est identifié par :
  - Le tuteur (l'organisation) à suivre.
  - Les cours posés à la disposition des apprenants.
  - Les questions qu'il pose.

- Une organisation du cours est composée d'un ensemble d'éléments plus les concepts supplémentaires. Chaque élément de cours ou un concept est défini par :
  - Un titre.
  - Sa durée approximative : un pourcentage par rapport la durée générale spécifiée au module.
  - Le niveau de détail.

Exemple 1

Soit l'organisation suivante du cours du module 'SMA' de la formation 'Informatique':

	Titre	Niveau de détail	Durée partielle(%)
Les éléments du cours	Introduction	25%	1%
	Définition d'un agent	25%	2%
	Définition formelle d'un agent	50%	2%
	Architecture Abstraite d'un agent	75%	3%
	Architectures concrètes d'un agent	75%	5%
	Agent intélligent	25%	2%
	Définition des SMA	25%	2%
	Architectures des SMA	25%	2%
	Intéraction entre agents	25%	5%
	Coopération	50%	5%
	Coordination	50%	5%
	Négotiation	50%	5%
	Communication entre agents	25%	5%
	Tableau noir	50%	5%
	KQML	50%	5%
	ACL	50%	5%
	Primitives KQML	75%	8%
	Primitives ACL	75%	8%
	Méthodes de conception des SMA	100%	13%
	Outils d'implémentation des SMA	100%	12%
Concepts	Methodes de conception classiques	100%	1%
	Intélligence artificielle	25%	1%
	Intélligence collective	25%	1%
	Intélligence distribue	25%	1%
	La logique	75%	1%

Figure (11): un exemple d'une organisation de

Supposons que la durée autorisée pour prendre ce module est 3 heures. Un apprenant de niveau 50% apprend les éléments de cours attribués par les niveaux 25% et 50%. Donc, le temps de formation de module propre à cet apprenant est 54% de 3 heures ; c'est à dire:1.62 heures.

- Question proposée par un auteur est composée de :
  - Le thème: c'est la liste des concepts ou les éléments de cours constituant le sujet de la question.
  - Le niveau de détail autorisé pour prendre cette question.
  - Le contenu de la question : la problématique posée.
  - La liste des réponses proposées : nous utilisons le système QCM (Question aux Choix Multiples), où l'apprenant doit choisir celle (s) juste (s). Chaque élément de cette liste est composé de :
    - ☒ Le contenu de la réponse.
    - ☒ Sa vérité : Vrai / faux.

Exemple 2

Soit la question suivante posée dans le cadre de module présenté dans l'exemple 1:

Le thème	Définition des SMA	Intélligence distribuée
Le niveau de détail	25%	
Le contenu de la question	Pourquoi les SMA sont classés dans le cadre de l'intélligence distribuée?	
La liste des réponses proposées	Le contenu de la réponse	vérité
	Ensemble d'entités travailant d'une manière intélligente	faux
	Ensemble d'entités intélligentes divisant une tâche entre eux pour la réaliser sans contrôle général	vrai

Figure (12): un exemple d'une question

4.4.2 Conventions sur des notions

Un système éducatif est défini par un ensemble des formations. Chacune d'elles est composée d'un ensemble de modules. Le cours d'un module peut avoir plusieurs organisations. On peut trouver divers rédactions suivant une seul organisation. Donc, nous avons trois éléments essentielles et qui sont : la formation, l'organisation et la rédaction.

Dans ce qui suit, nous représentons les différents états de chacun des éléments précédents.

a. Les états d'une rédaction

i. Autorisé : la rédaction est prêtes aux différentes utilisations. C'est-à-dire, les nouveaux apprenants peuvent être inscrit pour apprendre le cours d'un module présenté par cette rédaction. L'auteur de cette rédaction peut accéder à son compte et il peut faire tout ce qui est autorisé à lui. La rédaction sera autorisée à l'utilisation juste après l'inscription effective de son auteur.

ii. Interdit : l'utilisation de la rédaction est interdite. L'ordre de l'interdiction est donnée par l'administrateur du système. Dans ce cas, la rédaction est physiquement existe dans le système, bien qu'elle n'est prise pas en considération.

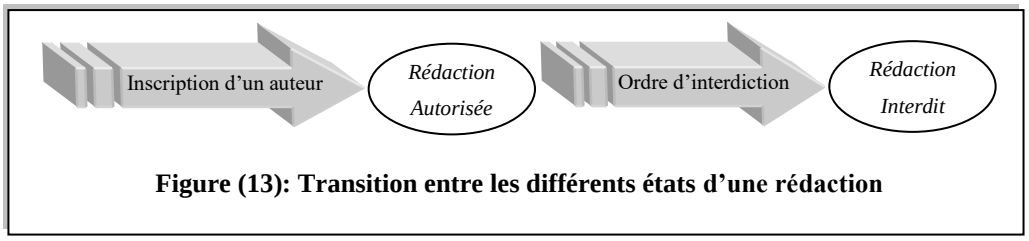


Figure (13): Transition entre les différents états d'une rédaction

b. Les états d'une organisation

i. Autorisé : l'organisation est prêtes aux différentes utilisations. C'est-à-dire, les nouveaux apprenants peuvent être inscrit pour apprendre les rédactions de cours du module suivant cette organisation. Ainsi, elle a le droit de la publication pour attirer l'attention des auteurs. Le tuteur de cette organisation peut accéder à son compte et il peut faire tout ce qui est autorisé à lui. Une organisation sera autorisée à l'utilisation juste après l'inscription effective de son tuteur.

ii. Disponible : une organisation est dite ″disponible″ si elle a au moins une rédaction Autorisée.

iii. Interdit : l'utilisation de l'organisation est interdite. L'ordre de l'interdiction est donnée par l'administrateur du système. Dans ce cas, l'organisation est physiquement existe dans le système, bien qu'elle n'est prise pas en considération dans les endroits cités précédemment. Son existence ne sera pas remarquée par les visiteurs de l'application sauf les anciens qui sont inscrits comme des membres (apprenants, auteurs) suivant cette organisation ; les apprenants auront terminée normalement tandis que les auteurs ne peuvent pas accéder à leurs comptes. Donc, il n'existe pas des nouvelles inscriptions de n'importe quel type (apprenant, auteur) sous cette organisation.

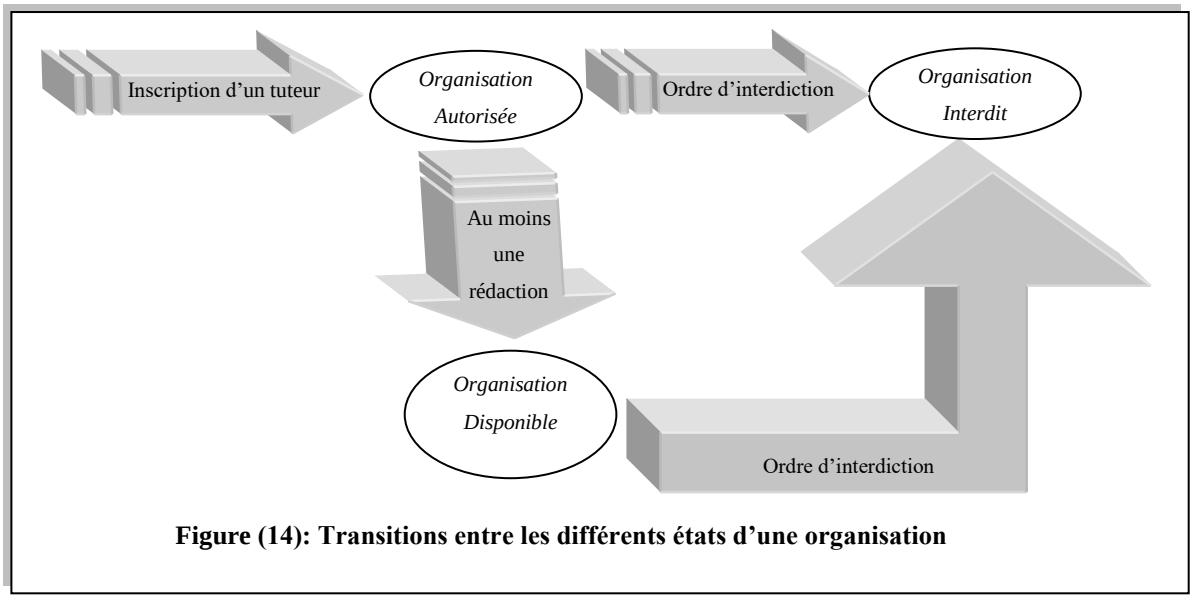


Figure (14): Transitions entre les différents états d'une organisation

c. Les états d'une formation

i. Défini : cet état est associé à une formation grâce à sa définition par l'administrateur du système. Le processus de définition d'une formation contient la déclaration de ses différents composants et attributs.

ii. Autorisé : une formation dans un état *Défini* sera dans un état *Autorisé* à cause d'une permission d'utilisation et de publication donnée par l'administrateur du système.

iii. Disponible : une formation *Disponible* a pour chacun de ses modules au moins une organisation *Disponible*.

iv. Interdit : l'affectation de cet état est une conséquence à une décision de l'administrateur du système. L'existance de la formation ne sera pas remarquée par les visiteurs de l'application sauf les anciens qui sont inscrits comme des membres (apprenants, auteurs, tuteurs) appartiennent à cette formation ; les apprenants auront terminée normalement tandis que les auteurs et les tuteurs ne peuvent pas accéder à leurs comptes. Donc, il n'existe pas des nouvelles inscriptions de n'importe quel type (apprenant, auteur, tuteur).

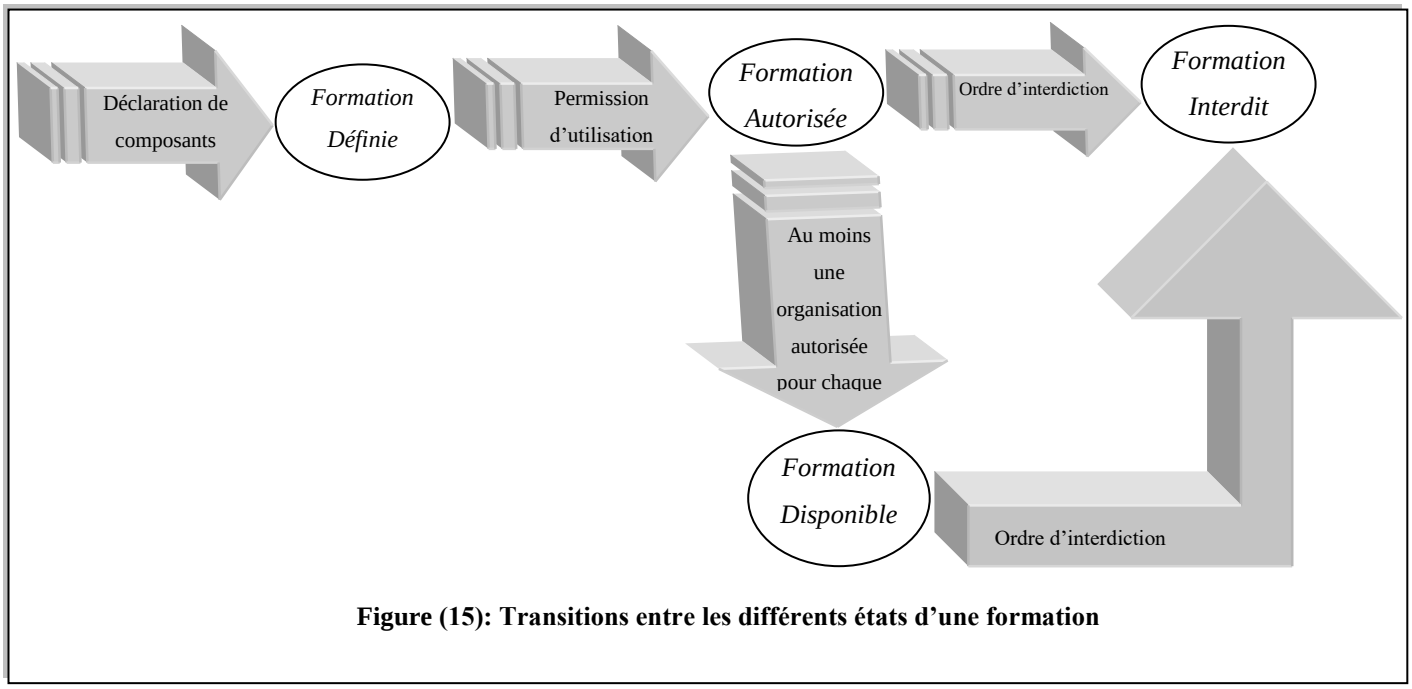


Figure (15): Transitions entre les différents états d'une formation

4.4.3 Architecture générale du système

4.4.3.1 Agents

Le choix d'un modèle d'agent se fait essentiellement en fonction des besoins qui sont exprimés pour cet agent au cours de la phase d'analyse. Tous les agents doivent disposer de moyens de communication de haut niveau, de capacité de raisonnement et de prise de décision. Ces besoins nous conduisent à choisir un modèle d'agent cognitif pour nos agents. Le schéma ci-dessous présente l'architecture générale d'un agent cognitif.

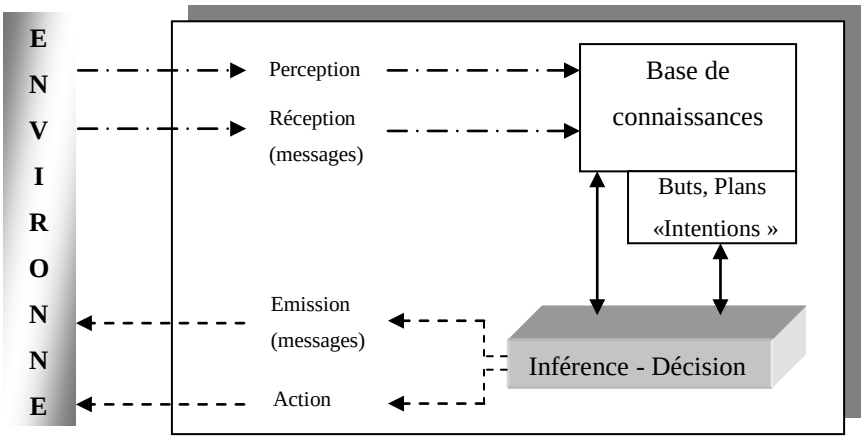


Figure (16): Architecture d'un agent cognitif

Les agents cognitifs fonctionnent suivant un processus en boucle de type Perception/Décision/Action.

Le type de l'application nous dirige d'utiliser les agents collaboratifs. Les tâches définissant notre système sont distribuées entre eux. Chacun d'eux travaille d'une façon de donner aux autres de l'aide (des connaissances, des plans, exécutions certaines fonctions,…).

4.4.3.2 Société d'agents

Nous identifions un agent appelé « *Hôte* » pour chaque utilisateur (de type apprenant, tuteur, auteur ou administrateur) qui visite le site.

Nous identifions un agent « *Apprenant* » et un agent « *Testeur* » pour chaque apprenant qui dispose d'un compte, un agent « Organisateur », un agent « «Auteur » pour chaque enseignant respectivement de type tuteur, auteur. En plus, nous avons besoin d'un agent « *Administrateur* » pour l'administration de système.

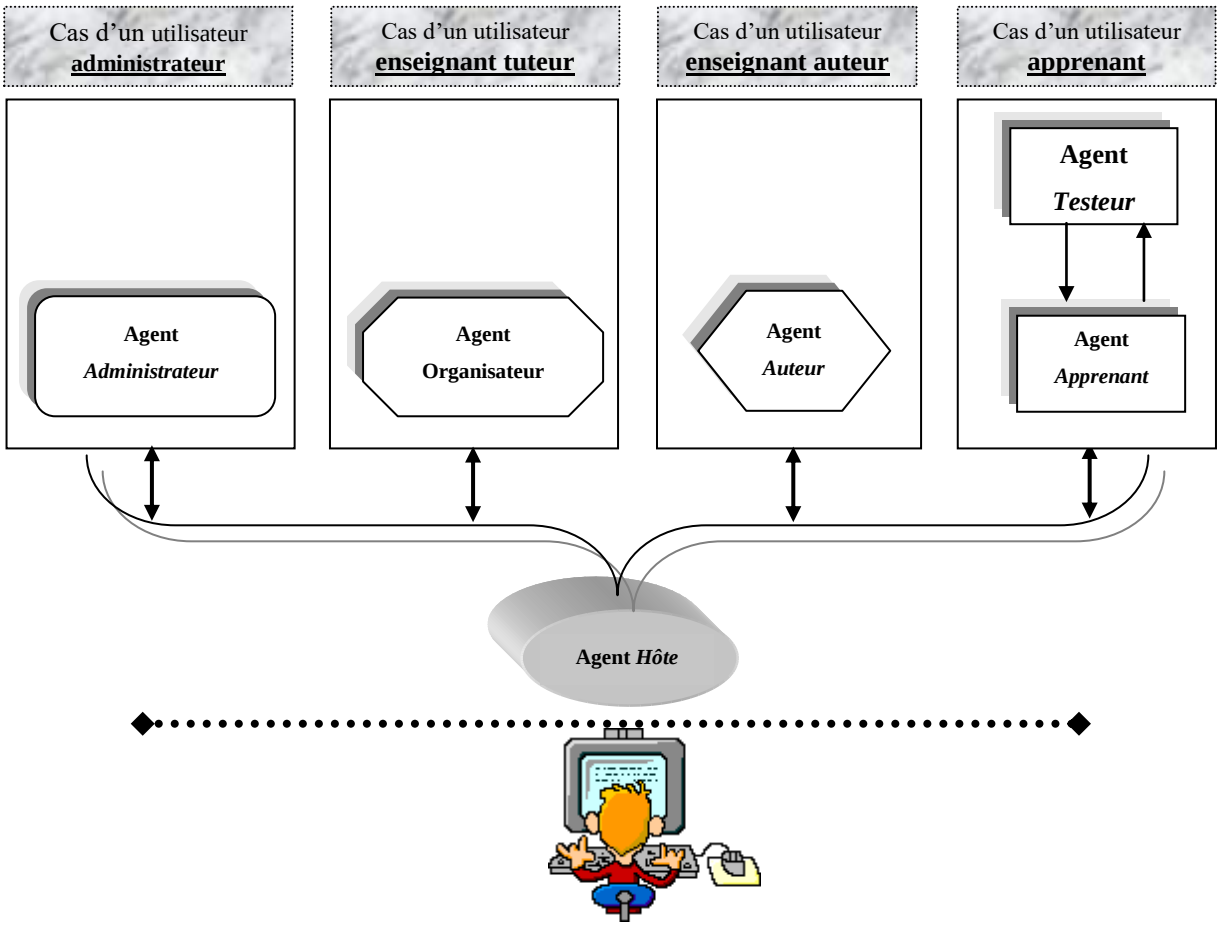


Figure (17): Architecture générale de notre système

L'Agent *Hôte*

Il est responsable de la représentation en temps réel du contenu du système au visiteur de l'application. Ainsi, il assure la sécurité d'accès aux différents comptes dans le cas des utilisateurs déjà inscrit, et de donner des comptes aux nouveaux utilisateurs qui décident de s'inscrire. L'existence de cet agent est liée étroitement à chaque visite au site de l'application.

L'Agent *Administrateur*

Il est créé dans le cas d'un utilisateur représentant l'administration du système. Cet agent est le compagnon de l'administrateur (personne) durant tous ses travaux dans le système éducatif. Il lui offre :

- un contrôle dans l'ajout / suppression des formations, modules, ….
- un bilan des différents résultats concernant des divers aspects du système éducatif.
- une aide pour prendre des décisions en se basant sur les résultats obtenus.
- l'exécution des différentes décisions dans le bon contexte pour éviter les problèmes.

L'Agent *Apprenant*

Il est considéré comme un lien réel entre le système et l'apprenant qui désire se former. Cet agent peut, en cours du temps, apprendre les intérêts de l'apprenant, son niveau et son progression dans le cours de chaque module et dans la formation entièrement.

Dans chaque module, l'agent *Apprenant* se concentre de suivre l'apprenant en se basant sur l'organisation suivie.

Il joue le rôle de l'enseignant comme un moteur d'activité dans les salles présentièlles. Ce rôle est traduit, au niveau de l'agent, par les tests surprises, les conseilles, les remarques, les critiques,…

Une autre tâche de cet agent présentée par la prise en charge des procédures d'inscription d'un nouveau apprenant. L'exécution de cette tache est liée à la demande de l'Agent *Hôte*.

L'Agent *Testeur*

Il fonctionne en interaction avec l'agent *Apprenant.* Son fonctionnement est lié aux différentes préoccupations de l'agent *Apprenant.*

Il est responsable de préparer les différents tests (examens) dépendamment du niveau de l'apprenant et pas nécessairement comme un ensemble des blocs statiques

diffusés à tous les apprenants. Donc, les examens sont donnés à l'apprenant d'une manière adaptative à sa progression dans les cours.

L'Agent *Organisateur*

Il est associé à chaque utilisateur de type tuteur. Son rôle est d'accompagner le tuteur au cours de son travail. Il l'assiste dans la proposition d'une organisation. Ainsi, il lui offre des bilans des différents résultats concernant les auteurs qui lui suivent. En plus, il est considéré comme un aide d'appliquer les décisions prises par le tuteur.

Une autre tâche de cet agent consiste la prise en charge des procédures d'inscription d'un nouveau tuteur. L'exécution de cette tache est liée à la demande de l'Agent *Hôte*.

L'Agent *Auteur*

Il est associé à chaque utilisateur de type auteur. Son rôle est d'accompagner l'auteur au cours de son travail. Il l'assiste dans la rédaction du cours en suivant l'organisation choisie. Ainsi, il lui offre des bilans des différents résultats concernant les apprenants qui suivent ses cours.

Il lui donne une souplesse de gérer ses questions (ajout/ modification/suppression). Il prend en charge ces opérations afin d'éviter les problèmes de bloquer le déroulement du système éducatif.

Une autre tâche de cet agent est l'exécution des procédures d'inscription d'un nouveau auteur. L'exécution de cette tache est liée à la demande de l'Agent *Hôte*.

4.4.3.3 Communication entre agents

On peut différencier entre plusieurs sous sociétés d'agents dans notre système. Chacune d'elles accompagne un type d'utilisateur. Donc, nous avons quatre sociétés d'agents à savoir :

- société d'agents accompagnant l'apprenant = {agent Hôte, agent Apprenant, agent Testeur},
- société d'agents accompagnant l'auteur = { agent *Hôte*, agent *Auteur*},
- société d'agents accompagnant le tuteur = { agent *Hôte*, agent *Tuteur*},
- société d'agents accompagnant l'administrateur du système = {agent *Hôte*, agent *Administrateur*}.

Nous utilisons comme mode de communication entre agents composant notre système l'envoi de messages. Il est utilisé (voir **Figure (18)**) entre les agents qui appartiennent à la même société d'agents sur la même station physique (PC).

Les agents posent des informations (questions, cours, organisations, différentes décisions,…) dans des structures de stockage (des bases de données,…) (voir **Figure (18)**). Ces informations (données) peuvent être utilisées par d'autres agents.

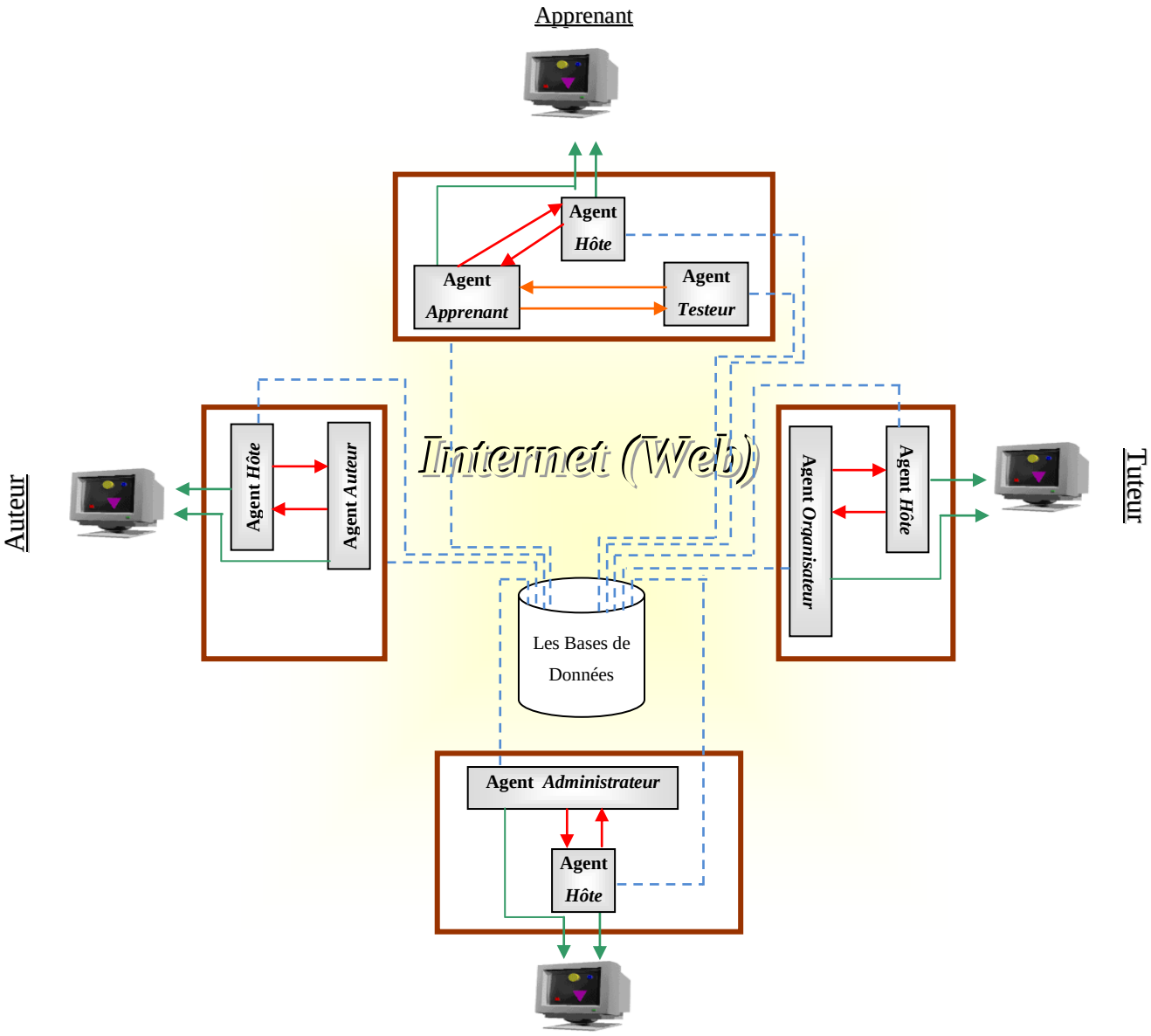


Figure (18): Modes de communication

Les bases de données: sont les structures de stockages que nous utilisons pour sauvegarder les informations nécessaires pour le déroulement de notre système. Nous distinguons entre quatre bases de données à savoir :

- *BdFormation*: elle accumule toutes les informations des différentes formations constituants le système éducatif (les modules, les tuteurs, les auteurs,…).
- *BdQuestion* : elle se compose des questions posées par les différents auteurs.
- *BdOrgRed* : elle contient les organisations des cours et leurs rédactions.
- *BdApprenant* : les données (informations) stockées dans cette base de données concernent les apprenants.

Les **Figures (19)**, **(20)**, **(21)** et **(22)** expliquent les types de messages envoyés entre les agents de chacune des sociétés précédentes.

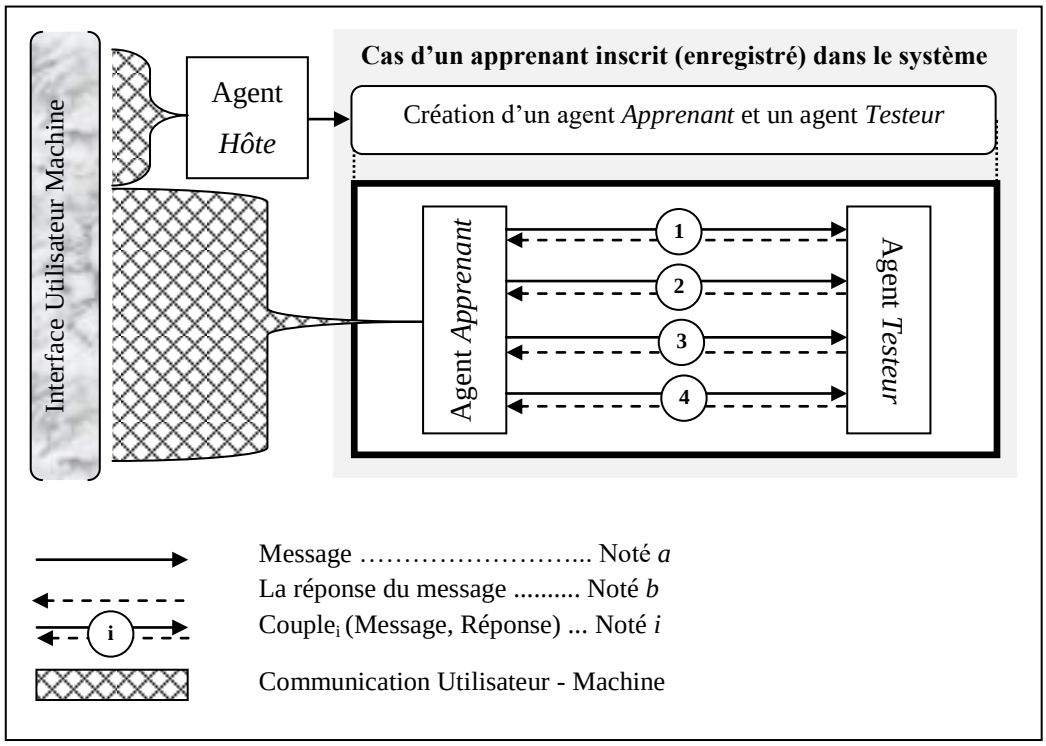


Figure (19) : Les messages entre agents dans le cas d'un utilisateur de type

1.a = Demande de préparer un test (examen) pour l'apprenant. Ce message est paramétré comme suit : - le module,

- la progression de l'apprenant dans le cours,
- le type de test : un test complet ou un court test. Le premier signifie un test contient tout ce qui est appris par l'apprenant. Le deuxième est un test contient quelques questions pas nécessairement touchent tout ce qui appris par l'apprenant.

1.b = Réponse positive : le test,

Réponse négative : impossibilité de préparer un test au moment donné.

2.a = Les réponses de l'apprenant dans le test.

2.b = Le résultat obtenu par l'apprenant dans le test.

3.a = Demande de présenter le résultat obtenu par l'apprenant dans un test, module ou la formation.

3.b = Le résultat.

4.a = Demande d'un bilan des différents résultats obtenus par l'apprenant.

4.b = Le bilan.

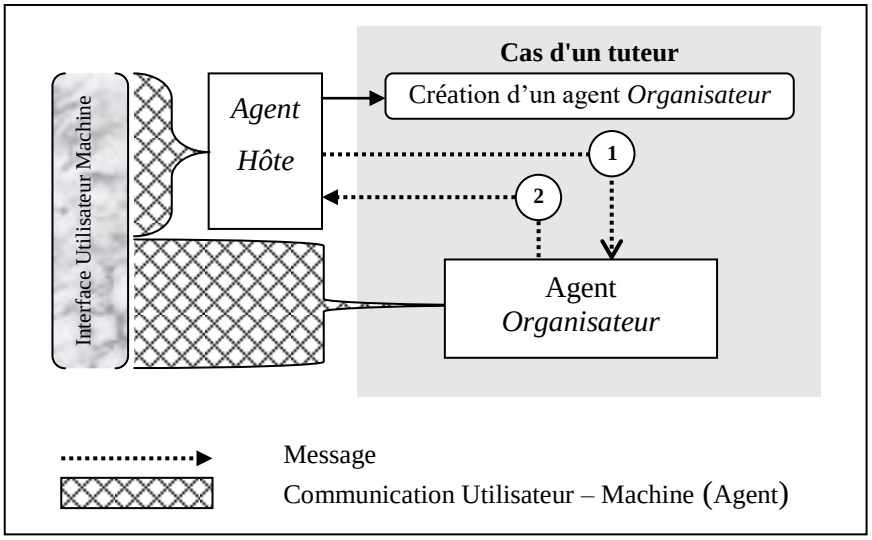


Figure (20) : Les messages entre agents dans le cas d'un utilisateur de type tuteur

1 = Demande d'assister le tuteur dans la proposition d'une organisation de cours pour qu'il sera inscrit. Ses paramètres sont : la formation et le module.

2 = Réponse de la demande 1; Réponse positive : organisation bien posée,

Réponse négative : le tuteur ne respect pas ses engagements.

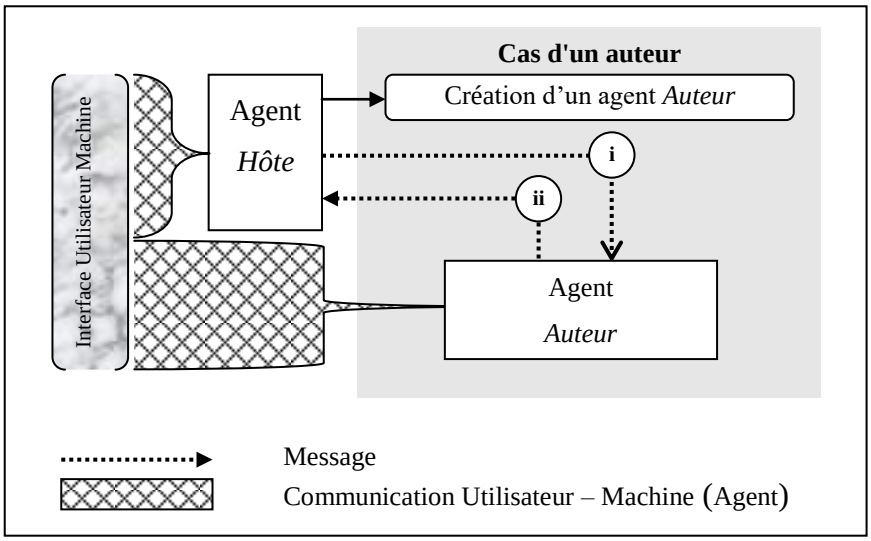


Figure (21) : Les messages entre agents dans le cas d'un utilisateur de type auteur

i = Demande d'assister l'auteur dans la proposition d'une rédaction de cours pour qu'il sera inscrit. Ses paramètres sont : la formation, le module et l'organisation (le tuteur) à suivre.

ii = Réponse de la demande 1; Réponse positive : rédaction bien posée,

Réponse négative : l'auteur ne respect pas ses engagements.

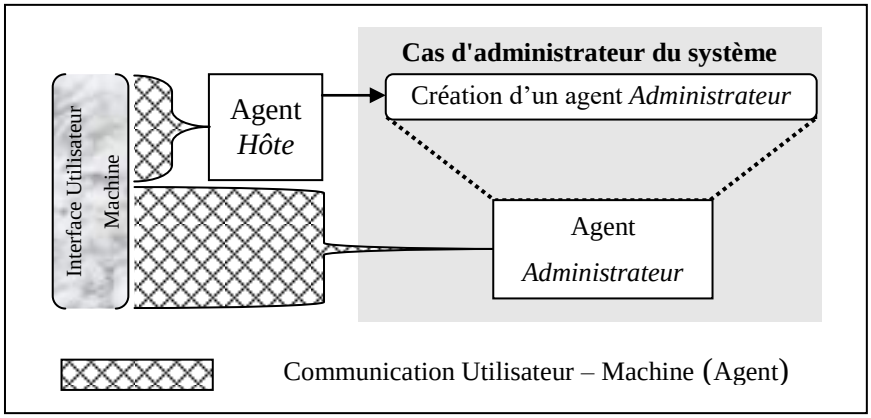


Figure (22) : Les messages entre agents dans le cas d'administrateur du système

Remarque : Dans les quatre derniers schémas, la création d'agents correspond au type d'utilisateur est effectuée par l'agent *Hôte*.

4.4.3.4 Architecture de chaque agent

Dans cette section, nous détaillons deux types d'architectures pour chaque agent : architecture structurelle et architecture fonctionnelle.

- **Cas 1 : Agents associés à chaque visiteur du site (un utilisateur de n'importe quel type)**

Cas1: (1) Agent *Hôte*

***Cas1: (1).a* Architecture structurelle**

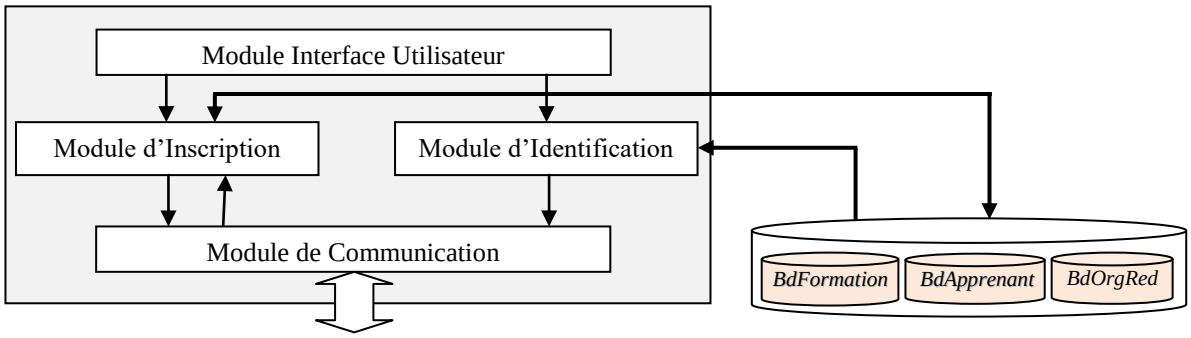


Figure (23) : Architecture structurelle d'agent *Hôte*

L'agent *Hôte* assure l'inscription des nouveaux utilisateurs et l'identification des utilisateurs déjà inscrits par les modules : **Module d'Inscription** et **Module d'Identification** respectivement. Le module d'Inscription demande les informations nécessaires de l'utilisateur et initialise son profil en fonction de ces informations. Le module d'Identification demande à l'utilisateur connecté de donner son nom d'utilisateur et mot de passe pour vérifier son inscription (enregistrement).

Le **Module d'Interface Utilisateur** est responsable de la présentation de ce qui disponible dans le système éducatif et les différentes fonctionnalités du système sous forme d'une interface graphique.

Enfin, le **Module de Communication** se charge de la construction des messages et leur envoi et la réception des messages reçus. A travers ce module, l'agent Hôte communique avec les autres agents.

***Cas1: (1).b* Architecture fonctionnelle**

Sa création est liée à l'accès au site. Ses différents savoirs sont présentés comme suit :

- **Télécharger et afficher les formations disponibles**

Une formation est dite disponible si elle existe dans le système éducatif à condition qu'elle est autorisée par l'administration de système et elle a au moins une rédaction autorisée pour chaque module. L'agent *Hôte* affiche donc toutes les formations disponibles avec leurs informations nécessaires.

Les nouveaux apprenants peuvent s'inscrire seulement dans les formations disponibles.

- **Télécharger et afficher les formations autorisées**

Une formation est dite autorisée si elle existe dans le système éducatif (son état est Défini) à condition qu'elle est autorisée par l'administration de système.

L'objectif de l'affichage des formations autorisées est d'attirer l'attention des enseignants (tuteurs et auteurs) afin qu'elles seront disponibles.

Les nouveaux apprenants n'ont pas le droit de s'inscrire dans les formations de ce type.

- **Télécharger et Afficher les organisations autorisées pour chaque module d'une formation disponible ou autorisée**

L'objectif de l'affichage des organisations autorisées est d'attirer l'attention des auteurs afin qu'elles seront disponibles.

Dans le cas d'une formation disponible, les nouveaux apprenants n'ont pas le droit de choisir une organisation de ce type.

- **Télécharger et Afficher les organisations disponibles pour chaque module d'une formation disponible**

Les organisations disponibles d'une formation disponible sont affichées pour que l'apprenant avoir une vision générale des différents contenus proposés dans chaque module. L'utilité de cet offre est aussi de montrer aux enseignants ce qu'il est proposé; un auteur peut choisir une organisation à suivre, un tuteur peut proposer une organisation en évitant la redondance et la duplication des contenus proposés du module.

- **Vérifier l'existence d'un ancien utilisateur**

Quand un utilisateur veut accéder à son compte, l'agent *Hôte* vérifie son existence comme un membre dans le système éducatif par le nom d'utilisateur et le mot de passe.

Si le résultat de la vérification est positif, l'agent *Hôte* crée les agents compagnons d'utilisateur dépendamment de son type (un agent *Apprenant* et un agent *Testeur* pour un

utilisateur apprenant, un agent *Organisateur* pour un tuteur, un agent *Auteur* pour un auteur et un agent *Administrateur dans le cas contraire).*

- **Inscription d'un nouveau utilisateur**

L'agent *Hôte* se comporte de quatre façons par rapport à une demande d'inscription en fonction du type de l'utilisateur :

comportement 1 : pour un apprenant.

comportement 2 : pour un apprenant de type employeur qui demande de diminuer la durée de la formation.

comportement 3 : pour un tuteur.

comportement 4 : pour un auteur.

Dans chacun des comportements précédents, un utilisateur est inscrit c'est-à-dire il a un compte. L'accès à ce compte est à travers un nom utilisateur et un mot de passe. Ces deux identificateurs sont données par l'agent *Hôte*.

comportement 1

Supposons que l'apprenant a choisi de se former dans une formation *f*. Sa demande d'inscription est possible seulement dans le cas de *f* est disponible.

Pour chaque module *m* de *f*, l'apprenant doit choisir une organisation disponible *org* qu'il veut suivre. Ensuite, l'agent *Hôte* se passe au choix de la rédaction de cours (auteur) que l'apprenant doit apprendre suivant *org*.

Ce mécanisme consiste à choisir l'auteur ou bien la rédaction autorisée ayant un nombre minimum d'apprenants (en cours de formation ou terminés leurs formation).

comportement 2

L'apprenant (employeur) peut négocier pour diminuer la durée de formation selon ses besoins et les contraintes de travail. Il présente la durée désirée.

L'agent *Hôte* cherche, pour chaque module, un niveau de détail dans le cours avec une durée de formation tel que la collection de cours est adéquate à la durée totale de formation demandée par l'apprenant employeur.

Si ce recherche résulte plus d'une collection, l'agent choisi celle ayant la plus grande durée de formation.

Dans le cas où les modules ont des coefficients différents, l'agents choisi la collection des cours qui satisfait la demande de l'apprenant dans laquelle les cours des

modules avec les grands coefficients ont les niveaux les plus détaillés (les grands niveaux) que possible (la priorité est orientée coefficient).

Dans le cas où il n'y a aucune collection de cours qui satisfait le besoin de l'apprenant, l'agent *Hôte* lui propose des durées de formations qu'il peut en choisir une.

Cette recherche des collections des cours adéquate à l'apprenant est faite par les algorithmes de recherche d'un chemin optimal (Jukstra, Ford, Bellman,…).

Notons que ce processus touche seulement les organisations disponibles et les rédactions autorisées.

comportement 3

dans le cas d'un tuteur l'agent *Hôte* crée un agent *Organisateur* pour suivre les procédures d'inscription.

L'agent *Hôte* n'inscrit pas le tuteur qu'après avoir que l'agent *Organisateur* lui informe que le tuteur applique bien les procédures ce qui le dirige vers une organisation bien faite.

comportement 4

Un nouveau auteur a la possibilité de s'inscrire pour rédiger le cours d'un module d'une formation *f* autorisée ou disponible suivant une organisation *org* autorisée ou disponible. L'agent *Hôte* crée un agent *Auteur* pour suivre les procédures d'inscription.

L'agent *Hôte* n'inscrit pas l'auteur qu'après avoir que l'agent *Auteur* lui informe que le tuteur applique bien les procédures ce qui le dirige vers une rédaction de cours bien faite.

Remarque

Dans les modes d'inscriptions précédents, les comportements sont contenus dans des cycles (cas des inscriptions refusées) jusqu'à ce que l'utilisateur atteint son désire. Donc, pendant chaque cycle, il y a des sauvegardes locales. Ces sauvegardes sont supprimés à chaque nouvelle demande d'inscription.

Quand la demande d'inscription est acceptée et l'utilisateur confirme son désire de s'inscrire, l'agent *Hôte* confirme à lui son nom d'utilisateur et un mot de passe. Les informations nécessaires de l'utilisateur seront sauvegardées dans les structures de stockages.

Cas 2 : Agents associés à chaque apprenant

Cas2: (1) *Agent Apprenant*

Il est le compagnon de l'apprenant pendant l'ouverture de son compte (session). Ce compagnon joue le rôle de l'interface de système devant l'utilisateur de type apprenant.

Cas2: (1).a Architecture structurelle

Le **Module d'Interface Apprenant** fournit une interface graphique des différentes fonctionnalités du système présentés dans le cas d'un utilisateur de type apprenant.

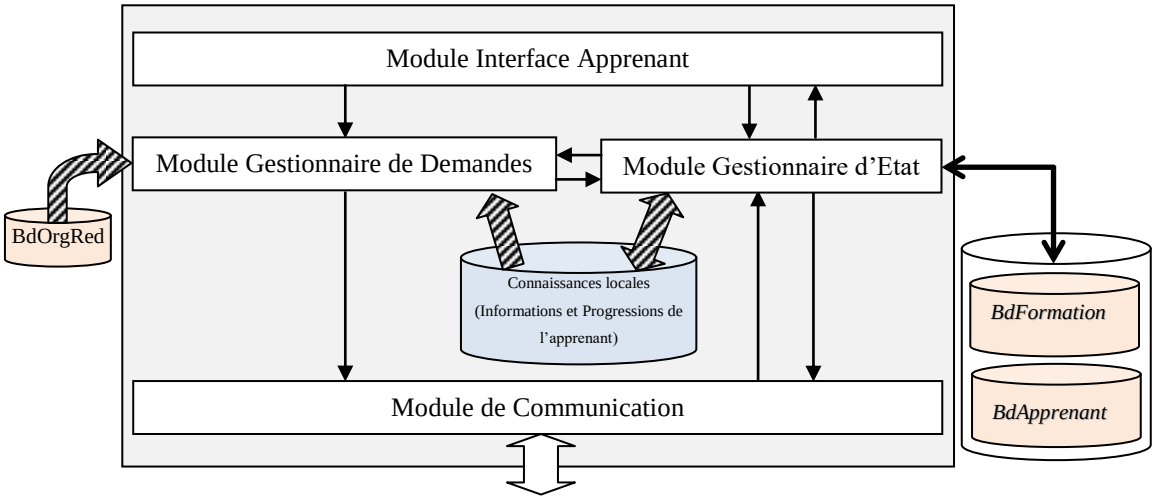


Figure (24) : Architecture structurelle d'agent *Apprenant*

Les divers préoccupations et demandes de l'apprenant sont prises en charge par le **Module Gestionnaires de Demandes**. Bien que, le suit de l'apprenant en cours de l'apprentissage est fait par le **Module Gestionnaires d'Etat**.

Le **Module de Communication** est, comme ses équivalents dans les autres agents, responsable du traitement des messages entrant et de l'élaboration des messages sortants.

Cas2: (1).b Architecture fonctionnelle

Les savoirs de l'agent *Apprenant* sont comme suit :

- **Suivi de la progression de l'apprenant**

Chaque apprenant inscrit peut consulter et réviser les cours des différents modules d'une manière libre. Donc, il peut migrer entre les différents modules. Quand l'apprenant se déconnecte ou fait une transition vers un autre module, l'agent sauvegarde le point d'arrêt de l'apprenant dans le cours et calcule le temps restant pour chaque module.

Quand l'apprenant se connecte, l'agent *Apprenant* télécharge les temps restant à l'apprenant dans les différents modules. Pour chacun de ces derniers, l'agent évalue l'avancement de l'apprenant. Dans le cas d'un retard énorme, l'agent l'alerte.

- **Demande d'un test**

L'apprenant demande de faire un test dans un module à n'importe quel point dans le cours. L'agent *Apprenant* envoi la demande à l'agent *Testeur*. Cette demande est associée par la progression de l'apprenant dans le cours avec les concepts étudiés.

Si la demande ne peut être faite, l'apprenant sera informé qu'il ne peut pas passer un test.

- **Affichage du résultat d'un test**

L'agent *Apprenant* reçoit de l'agent *Testeur* le résultat du test fait par l'apprenant. Si l'apprenant veut répéter un autre test, l'agent *Apprenant* renvoi une autre demande de test à l'agent *Testeur*.

- **Changement de la routine de l'apprenant**

De temps en temps, l'agent affiche à l'apprenant des images, des vidéos ou des textes expressifs ou pour loisir. Ces interruptions sont faites après un bon ou un mauvais résultats d'un test, quand l'apprenant ne progresse pas bien dans le cours ou d'une manière aléatoire pour changer le stress de l'éducation.

- **Passage d'un test programmé par le tuteur**

Supposons que l'apprenant est en phase d'apprentissage de cours d'un module, quand l'un des temps proposés par le tuteur pour faire un test se déclenche l'agent *Apprenant* demande à l'agent *Testeur* de préparer un test à l'apprenant.

- **Recherche d'une explication d'un concept**

Le tuteur propose un ensemble de concepts à expliquer. Chaque enseignent auteur doit prendre en considération ces concepts. L'auteur peut donner plusieurs explications à un seul concept.

Quand l'apprenant demande une explication pour comprendre un concept. L'agent *Apprenant* cherche la réponse dans les cours proposés par l'auteur. Si la réponse n'est pas convaincante à l'apprenant, l'agent cherche s'il y a une réponse alternative posée par le même auteur, sinon il cherche la réponse dans les explications posées par les autres auteurs suivant la même organisation (tuteur). Cette recherche se déroule jusqu'à la satisfaction de l'apprenant.

L'agent sauvegarde l'historique de question/réponse de l'apprenant et ses réactions vers les réponses reçues. La sauvegarde contient :

- Le concept
- La liste des réponses reçues. Chaque réponse est identifiée par :
  - ✓ Auteur
  - ✓ Identificateur de la réponse pour différencier les réponses du même concept données par un seul auteur.
  - ✓ Réaction de l'apprenant vers la réponse (positive/négative).

Pendant la durée d'ouverture de la session de l'apprenant, l'historique et le comportement de l'apprenant sont considérés comme des connaissances (base de faits) propres à l'agent *Apprenant*. Ces connaissances changent et elles contiennent :

- la progression de l'apprenant dans chaque module.
- L'ensemble de concepts étudiés par l'apprenant dans chaque module.

- **Faire un test pour attirer l'attention de l'apprenant**

L'agent *Apprenant* demande à l'agent *Testeur* de préparer un test de courte durée pour l'apprenant pour l'éveiller. Ce test contient au plus cinq questions.

- **Consultation d'état dans le système**

L'apprenant peut consulter son état dans le système éducatif. L'agent *Apprenant* qui lui donne un relevé de ce qu'il a fait jusqu'à ce moment. Ce relevé contient :

- les concepts qu'il a étudié,
- le temps restant,
- sa progression dans le cours,
- les résultats des différents tests qu'il a fait.

Le relevé est créé à partir des connaissances de l'agent *Apprenant* sauf le dernier composant de relevé est qui créé en coopération avec l'agent *Testeur* (ses connaissances).

- **Evaluation du résultat final de l'apprenant**

Après que l'agent *Apprenant* reçoit les résultats finaux de l'agent *Testeur*, il décide si l'apprenant a réussi ou a échoué.

Donc, l'agent *Apprenant* a un bilan des moyennes finales dans tous les modules. L'agent *Apprenant* compare la moyenne générale avec la moyenne éliminatoire de la formation suivie. L'apprenant sera ajourné dans le cas d'une valeur inférieur.

Dans le cas d'un système modulaire, une autre évaluation à faire si la condition précédente est vérifiée. L'agent *Apprenant* vérifie si l'apprenant a obtenu la moyenne éliminatoire dans chaque module, sinon, il sera considéré comme ajourné.

Cas2: (2) ***Agent Testeur***

Il se charge de fonctionner grâce à des demandes (messages) envoyées par l'agent *Apprenant*.

Cas2: (2).a **Architecture structurelle**

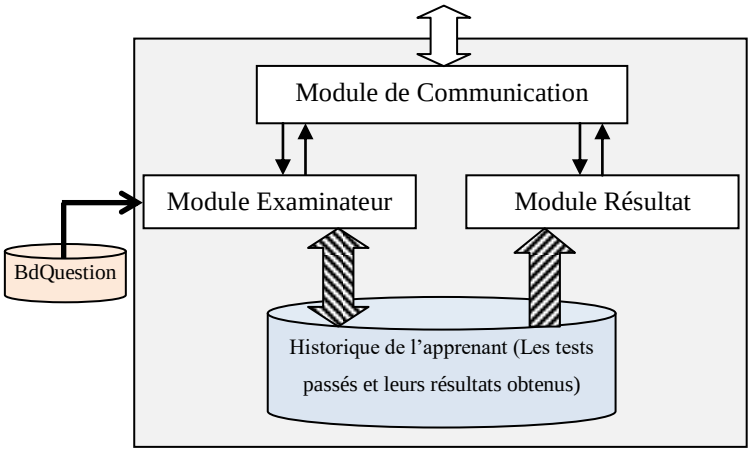


Figure (25) : Architecture structurelle d'agent *Testeur*

Module de Communication : c'est l'interface de l'agent. Il gère la communication de l'agent avec les composants de son environnement. Il est responsable de l'envoi et la réception des messages.

Module d'Examinateur : ce module est responsable de la préparation des tests (examens). Il corrige aussi les réponses rendus par l'apprenant.

Le module Résultat : il calcule et présente les résultats obtenus dans les différents tests, différents modules ou le résultat total de la formation à un instant donné.

Cas2: (2).b **Architecture fonctionnelle**

- **Préparer un test**

Un test est spécifié à un module précis. Donc, l'agent *Testeur* prépare un test contient seulement les questions de module concerné.

Le contenu d'un test est liée au contexte de l'apprenant:

- La progression de l'apprenant dans le cours,
- Les concepts étudiés par l'apprenant,

- Les tests déjà réalisés par l'apprenant.

Les deux premiers points sont des connaissances de l'agent *Apprenant*. Ce dernier les envoi avec la demande de préparer un test à l'agent *Testeur*.

De cette manière, chaque apprenant sera testé indépendamment des autres et dépendamment de ce qu'il apprend.

A la réception d'une demande de préparation d'un test, l'agent *Testeur* teste s'il reste des questions convenables au contexte de l'apprenant (l'historique). S'il n'existe pas, il envoi à l'agent *Apprenant* que sa demande de préparer un test ne peut pas être accepté.

Pendant la préparation d'un test (examen), l'agent *Testeur* doit respecter les règles suivantes :

- Choisir les questions qui sont posées par l'auteur de cours suivi par l'apprenant.
- Concernant les concepts supplémentaires, l'agent sélectionne les questions dont leurs thèmes sont les concepts étudiés par l'apprenant. Leurs sources sont les auteurs de cours qui suivent la même organisation que l'auteur de cours suivi par l'apprenant dans un module.
- Eviter de sélectionner les questions déjà vues par l'apprenant dans des tests passés (connaissances de l'agent *Testeur*).

Les questions sont classées d'une manière décroissante en se basant sur leurs thèmes; Soient $q_1,\ldots, q_n$ des questions, $\Omega(q_i)$ est la taille de la liste des thèmes de la questions q_i. Ces questions seront classées en dépendance à $\Omega(q_i)$ où $i = 1\ldots n$. Nous appelons une question d'ordre j si $\Omega(q_i) = j\ ;\ j \in N^+$.

L'agent Testeur sélectionne une seule question pour chaque liste de thèmes ; où les éléments de thèmes des questions doivent être étudiés par l'apprenant (comparaison avec sa progression dans le cours). L'agent commence par les questions d'ordre 1, suivi par les questions d'ordre 2, …

Donc, le nombre des questions dans un test est lié à la progression de l'apprenant dans le cours et les questions posées.

Dans le cas d'un test de durée courte (test à attirer l'attention de l'apprenant), l'agent *Testeur* ne dépasse pas cinq questions.

- **Calculer le résultat total d'un module dans un instant**

L'agent *Testeur* calcule le résultat final de l'apprenant dans un module *m*. le résultat obtenu est la moyenne des résultats des différents tests passés par l'apprenant dans ce module.

- **Calculer le résultat total de la formation dans une instant**

L'agent *Testeur* calcule la moyenne (résultat) générale de l'apprenant dans la formation. La moyenne générale est calculée en fonction des moyennes des différents modules et leurs coefficients.

- **Corriger les réponses de l'apprenant dans un test**

L'agent *Testeur* joue le rôle d'un correcteur des tests (examens). Il compare les réponses de l'apprenant avec les réponses associées aux questions qui composent le test. La comparaison touche le champs de la vérité de chaque réponse proposée pour une question.

Cas 3 : Agents associés à chaque tuteur

Cas3: (1) ***Agent Organisateur***

Il construit l'interface entre le système et le tuteur.

Cas3: (1).a **Architecture structurelle**

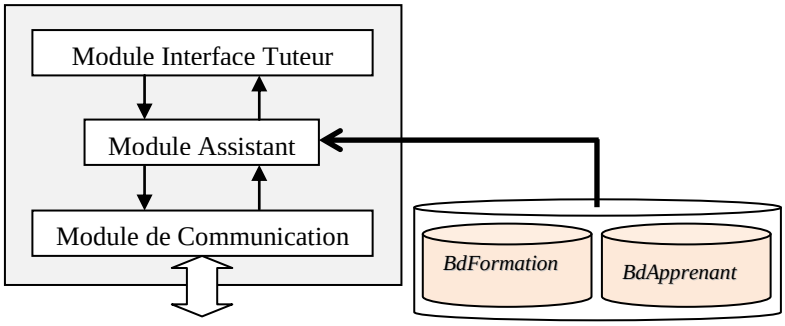


Figure (26) : Architecture structurelle d'agent

Module Interface Tuteur : fournit une interface graphique des différentes fonctionnalités du système présentés dans le cas d'un utilisateur de type tuteur.

Module Assistant : ce module sert à aider et suivre le tuteur pendant sa connexion.

Module de communication : c'est l'interface de l'agent avec les autres agents.

Cas3: (1).b **Architecture fonctionnelle**

- **Vérification d'organisation proposée**

L'agent *Organisateur* vérifie l'organisation proposée pour le module dans les points suivants :

1. chaque élément est associé par sa durée partielle.
2. chaque élément a un titre.
3. chaque élément est associé par son niveau (25%, 50%, 75%, 100%).
4. la somme des durées partielles des différents éléments (et les concepts) de cours ne dépasse pas la durée autorisée pour prendre le module.

S'il existe un problème à ces points, l'agent *Organisateur* informe le tuteur. Il lui suit jusqu'au ce qu'il n'y a aucun problème et l'organisation sera bien faite.

- **Vérification des propositions concernant les examens (tests)**

Le tuteur doit proposer le nombre minimum d'examens à faire par l'apprenant. Ainsi, il doit donner leurs meilleurs temps selon lui. L'agent *Organisateur* insiste sur le tuteur de respecter cette engagement.

Donc, l'agent *Organisateur* suit le tuteur jusqu'à ce que ses engagements seront bien respectés sans aucun problème ou un manque. A ce moment, l'agent *Organisateur* informe l'agent *Hôte* que le tuteur suit bien les renseignements et qu'il peut être inscrit.

Nous avons procédé de cette manière, car si l'agent *Hôte* crée un compte pour ce tuteur, le nombre de tuteurs sera incrémenté sans existence d'organisation. Dans ce cas, nous ne pouvons pas connaître quand le tuteur veut poser son organisation de cours et par conséquent quand il sera exclu du système éducatif. Comme il peut ne pas déposer l'organisation entièrement, pour éviter ce problème, le système oblige le tuteur de poser l'organisation sinon il ne sera pas inscrit.

- **Evaluation des questions d'un auteur**

Le tuteur a la possibilité d'évaluer une ou un ensemble de question de l'un de ses auteurs. Donc, l'agent *Organisateur* lui donne un bilan de résultats obtenus par les apprenants autours des questions posées par l'auteur considéré. Le bilan est présenté par le pourcentage des réponses justes à chaque question.

- **Evaluation des résultats obtenus par les apprenants d'un auteur**

L'agent *Organisateur* présente au tuteur le niveau général d'un groupe d'apprenants suivant les cours d'un auteur. Cette tache est considérée comme une autre norme aidant le tuteur d'évaluer l'auteur.

L'agent prend en considération les apprenants en cours de formation et ceux qui ont terminé.

- **Evaluation de la satisfaction des apprenants par les explications des concepts données par un auteur**

Cette évaluation contient le pourcentage général des réactions positives vers les explications données par un auteur. Ainsi, elle contient le pourcentage de chaque concept. Pour plus de précision, l'agent *Organisateur* donne au tuteur le taux des cas où les apprenants d'un auteur sont satisfaits par les explications d'un autre auteur.

- **Prendre des décisions à un auteur**

Si le tuteur veut envoyer des décisions à un auteur ou un ensemble des auteurs, l'agent *Organisateur* prend en charge ce désire. Les décisions ont le format des messages. Donc, l'agent *Organisateur* les envoi aux différents e_mails des concernés ou par les sauvegardes dans les structures de stockages. Dans ce dernier cas, l'agent *Auteur* compagnon à chaque compte d'un des auteurs concerné est responsable de transformer ces décisions à l'auteur.

Remarque : Le tuteur a le droit de demander les résultats concernant seulement les auteurs qui le suivent (leurs questions, leurs explications de concepts, leurs apprenants).

Cas 4 : Agents associés à chaque auteur

Cas 4: (1) ***Agent Auteur***

Cas 4: (1).a **Architecture structurelle**

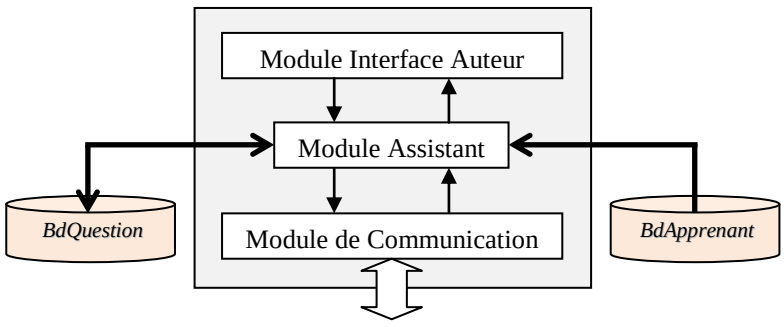


Figure (27) : Architecture structurelle d'agent

Module Interface Auteur : fournit une interface graphique des différentes fonctionnalités du système présentés dans le cas d'un utilisateur de type auteur.

Module Assistant : ce module sert à aider et suivre l'auteur pendant sa connexion.

Module de communication : c'est l'interface de l'agent avec les autres agents.

Cas 4: (1).b **Architecture fonctionnelle**

- **Suivi d'auteur pendant la rédaction de cours**

L'agent *Auteur* suivi l'auteur pour faire ses engagements qui sont :

- Rédaction du cours. L'agent renforce l'auteur de prendre en compte tous les éléments cités dans l'organisation suivie.
- L'explication des concepts supplémentaires. L'agent *Auteur* confirme que l'auteur donne au moins une explication pour chaque concept proposé dans l'organisation.

Donc, l'agent *Auteur* suit l'auteur jusqu'au la rédaction sera posée sans aucun problème ou un manque quelconque. A ce moment, l'agent *Auteur* informe l'agent *Hôte* que l'auteur suit bien les renseignements et qu'il peut être inscrit.

Nous avons procédé de cette manière, car si l'agent *Hôte* crée un compte pour cet auteur, le nombre d'auteur sera incrémenté sans existence de la rédaction. Dans ce cas, nous ne pouvons pas connaître quand l'auteur veut poser sa rédaction de cours et par conséquent quand il sera exclu du système éducatif. Comme il peut ne pas déposer de la rédaction entièrement, pour éviter ce problème, le système oblige l'auteur de poser la rédaction sinon il ne sera pas inscrit.

- **Suivi de l'auteur pendant la proposition des questions**

L'auteur pose des questions pour faire les différents tests des apprenants. L'agent *Auteur* vérifie la structure des questions posées par l'auteur. Il lui donne une liberté de poser n'importe quelle question à n'importe quel temps.

- **Evaluation des questions**

L'auteur a la possibilité d'évaluer une ou un ensemble de ses questions. Donc, l'agent *Auteur* lui donne un bilan de résultats obtenus par les apprenants autours de ces questions. Le bilan est présenté par le pourcentage des réponses justes à chaque question.

- **Suppression d'une question**

Si l'auteur voit qu'une question est inconvenable, il demande de l'exclure de l'utilisation. L'agent *Auteur* la supprime immédiatement dans le cas où la question est non utilisée, sinon, il associe à celle-ci une information d'interdire son utilisation.

La suppression dans ce cas est faite ultérieurement par le même agent (l'auteur redemande de supprimer la question) ou d'autres agents (Agent *Organisateur* compagnon le tuteur suivi par cet auteur, Agent *Administrateur* compagnon l'administrateur du système).

- **Ajout / suppression d'une explication à un concept**

L'agent *Auteur* prend la responsabilité d'ajouter des explications à un concept donné. Il donne à chaque explication un identificateur pour la différencie aux autres explications.

Il supprime aussi une explication d'un concept en prenant en compte que le concept a au moins une explication suivant l'organisation. Donc, si le concept a une seule explication, l'agent renforce l'auteur d'ajouter une autre pour supprimer l'ancienne.

L'ajout et la suppression d'une explication d'un concept sont effectués suivant une demande de l'auteur.

Cas 5 : Agents associés à l'administration du système

Cas 5: (1) Agent Administrateur

Cas 5: (1).a Architecture structurelle

Module Interface Administrateur : fournit une interface graphique des différentes fonctionnalités du système présentés dans le cas d'administrateur du système.

Module De Décisions: ce module aider l'administrateur de prendre et appliquer ses différentes décisions (ajouter/ supprimer une formation, évaluer/ exclure un auteur/tuteur…).

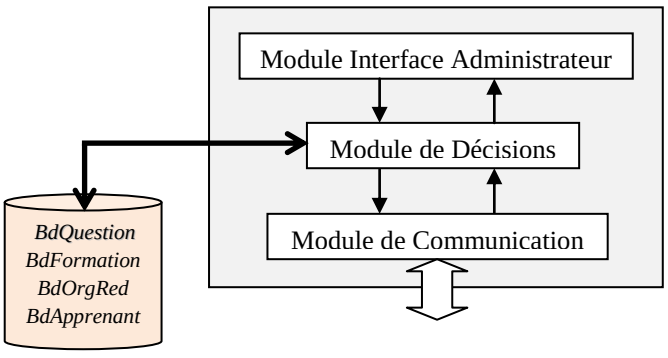


Figure (28) : Architecture structurelle d'agent

Module de communication : représente l'interface de l'agent avec les autres agents.

Cas 5: (1).b **Architecture fonctionnelle**

- **Suivi l'administrateur pendant l'Ajout d'une nouvelle formation**

L'agent *Administrateur* demande à l'administrateur de présenter les constituants de cette formation qui sont les modules. Chaque module est défini par :

- son nom
- son coefficient
- sa durée de formation
- le nombre maximum des organisations
- le nombre maximum des auteurs pour chaque organisation

L'agent *Administrateur* permet à l'administrateur de modifier ces différents attributs jusqu'à ce qu'il finisse la définition de la nouvelle formation.

- **Publication de la nouvelle formation**

Quand l'administrateur termine complètement de définir (son état est *Defini*) la nouvelle formation, l'agent *Administrateur* la publie sur le web. La publication d'une formation veut dire que son état sera autorisé.

La publication sur le web est pour informer les enseignants visitant le site par la nouvelle formation. Les enseignants qui sont intéressés par cette formation, se comporteront par des demandes d'inscription comme des tuteurs ou des auteurs.

- **Prendre la demande d'évaluation**

Nous laissons la suppression des tuteurs ou auteurs (ordre d'exclusion) à l'administrateur. Ce dernier, comme une personne, a les mécanismes d'évaluer l'échec ou la réussite d'organisations ou des cours des différents tuteurs ou auteurs.

Le système fournit à l'administrateur les différents résultats concernant chaque auteur ou tuteur. Le mécanisme d'évaluation se diffère suivant le point de vue de l'administrateur, de la formation, du nombre des tuteurs/auteurs, des résultats obtenus par les différents apprenants,…

Donc, l'agent Administrateur offre un relevé des résultats obtenus dans le système éducatif. Ces résultats concernent:

☒ Pour chaque auteur d'un module:

- Le pourcentage de réactions positives des apprenants vers chaque concept.
- Le pourcentage des réponses juste vers ses questions.
- La moyenne de ses apprenants qui terminent les cours du module.

- Pourcentage des abondons dans le module.
- Le pourcentage d'aide externes de ses apprenants (concernant les concepts). Càd, les cas où les apprenants demandent des explications alternatives aux celles données par leur enseignant auteur.

☒ Pour chaque tuteur d'un module:

- Le pourcentage général de réactions positives des apprenants vers chaque concept. Ce pourcentage est calculé en fonction des pourcentages des différents auteurs suivant l'organisation de ce tuteur.
- Pourcentage des abondons. Les abondons qui ne terminent pas leurs cours préparés par des auteurs suivant l'organisation de tuteur.
- La moyenne des apprenants qui ont terminé leurs formations soit par un échec ou un réussi. Ces apprenants ont suivi les cours de l'organisation du tuteur. Cette moyenne est calculée en fonction des moyennes obtenus par les auteurs qui lui suivent.

- **Suppression d'une formation**

Quand l'administrateur veut supprimer une formation, l'agent *Administrateur* vérifie qu'il n'existe pas des apprenants sous formation. Dans ce cas il supprime les ressources suivantes :

- la formation ,
- les tuteurs des différents modules constituent la formation
  - enlever l'organisations
  - supprimer les comptes des tuteurs (noms utilisateur et mots de passe pour interdire l'accès autre fois)
- les auteurs des cours des différents modules constituant la formation
  - enlever le cours
  - supprimer les comptes des auteurs (noms utilisateur et mots de passe pour interdire l'accès autre fois)

Dans le cas contraire, sinon il demande des agents *Hôte* de ne pas publier cette formation. En cours de temps, cette formation sera éliminée. La suppression sera effectuée par l'administrateur lui-même quand il se connecte autre fois.

- **Suppression d'un tuteur / auteur**

Quand l'administrateur veut exclure un tuteur / auteur, l'agent *Administrateur* vérifie qu'il n'existe pas des apprenants utilisant respectivement l'organisation / cours.

Dans le cas d'une exclusion d'un tuteur, il doit vérifier que toutes les rédactions suivant sa organisation ne sont pas en cours d'utilisation pour exclure les auteurs que le suivent.

Le cas d'un tuteur :

- exclure le tuteur
  - enlever l'organisation
  - supprimer le compte de tuteur (nom utilisateur et mot de passe pour interdire l'accès autre fois)
- exclure chaque auteur de cours suit l'organisation de ce tuteur :
  - enlever le cours
  - supprimer le compte de l'auteur (nom utilisateur et mot de passe pour interdire l'accès autre fois).

Le cas d'un auteur :

- enlever le cours
- supprimer le compte de l'auteur (nom utilisateur et mot de passe pour interdire l'accès autre fois)

Dans les cas contraires (organisation / rédaction en cours d'utilisation), l'agent *Administrateur* demande des agents *Hôte* de ne pas publier ce tuteur / auteur. En cours de temps, il sera éliminé. La suppression sera effectuée par l'administrateur lui-même quand il se connecte autre fois.

4.5 Conclusion

Dans ce chapitre, nous avons présenté notre architecture d'apprentissage en ligne basée agents. Nous avons utilisé les agents coopérants pour bénéficier de l'autonomie, la modularité, la distribution et l'intelligence des agents.

L'idée développée consiste à permettre à l'apprenant d'exploiter les ressources utilisées dans le cadre d'une formation de type e-Learning avec efficacité à travers la satisfaction de deux exigences a priori antinomiques, à savoir celle qui vise à lui accorder une certaine liberté de choix, et celle qui lui évite de se disperser eu égard à ses connaissances du moment. Nous n'avons pas voulu que l'apprenant soit soumis à des actions tutoriels directes, ni même qu'il les sollicite. S'il doit y avoir malgré tout intervention des formateurs, celle-ci doit être ponctuelle et limitée et d'une façon indirecte.

Les autres acteurs du système éducatif eux aussi sont pris en considération dans notre approche. Les tuteurs / les auteurs peuvent gérer facilement leurs organisations / rédactions des cours respectivement. L'administrateur du système gère le système éducatif avec souplesse.

Cette phase d'analyse et de conception nous a permis d'identifier et décomposer notre système en terme d'agents. Nous avons attribué un modèle à chacun de ses utilisateurs (acteurs). Notre modélisation a pour but de réaliser un apprentissage adaptatif avec la possibilité de changer la structure du système éducatif.

Dans le chapitre suivant, nous abordons la phase de la réalisation pour valider notre approche.

Chapitre 5

Implémentation

5.1 Introduction

Dans ce chapitre nous allons décrire les grands axes de la réalisation de notre système et les outils utilisés. Le meilleur moyen pour construire un système multi-agent (SMA) est d'utiliser une plate-forme multi-agent. Une plate-forme multi-agent est un ensemble d'outils nécessaire à la construction et à la mise en service d'agents au sein d'un environnement spécifique. Ces outils peuvent servir également à l'analyse et au test du SMA ainsi créé. Ces outils peuvent être sous la forme d'environnement de programmation (API) et d'applications permettant d'aider le développeur. Nous allons utiliser dans notre implémentation la plate-forme JADE(Java Agent DEvelopment framework).

Dans la section suivante, nous argumentons nos choix techniques en définissant la plateforme JADE suivie par une description du FIPA ACL. Les section 3,4 et 5 présentent les différentes classes et structures de données utilisées. Ensuite, une description des différents agents écrits en JADE avec les formes des messages envoyés entre eux. Nous finalisons par une conclusion du chapitre.

5.2 Choix techniques

- **Java comme langage de programmation**

Le choix du langage de programmation Java nous est en quelque sorte dicté par les contraintes d'exécution de l'application. En effet, Java permet l'écriture d'applications un peu particulières que l'on appelle appliquettes ou plus communément *applets*, qui s'exécutent dans un navigateur supportant Java après avoir été chargées à partir d'un serveur *Web.*

Les *applets* sont écrites dans un vrai langage de programmation et possèdent un potentiel bien plus important que toute combinaison de HTML (*HyperText Markup Language*), XML *(eXtensible Markup Language*) ou tout autre forme de scriptage. Cela

donne la possibilité d'exécuter de véritables applications sur tout ordinateur disposant d'un navigateur supportant Java, et ce, sans installation préalable de logiciel sur cette même machine.

Java permet aussi de développer des applications qui sont comparables à tout autre programme écrit dans un autre langage, au détail près qu'elles nécessitent comme pour les *applets*, une « machine virtuelle » pour leur exécution. La machine virtuelle Java interprète des classes Java exécutables qui sont générées par la compilation du code source. L'intérêt d'une telle démarche est de permettre l'exécution d'un même programme (sans recompilation) sur n'importe quel système informatique qui possède une machine virtuelle Java, et donc, de façon indépendante du système d'exploitation.

Les programmes sont développés avec la version 1.6 du JDK (*Java Development Kit*). Les tests sont réalisés sur Windows Xp Service Pack 2.

o **Choix d'une plate-forme SMA : JADE**

Nous cherchons parmi l'ensemble des réalisations existantes, une plate-forme SMA qui nous fournisse les outils nécessaires pour supporter la couche de communication et celle d'exécution et de suivi des protocoles d'interaction de notre architecture d'agents.

Les plates-formes SMA existantes se composent soit de produits issus de la recherche, soit de produits issus de firmes commerciales. Nous nous sommes appuyés sur les études de [25] qui présentent un comparatif des plates-formes SMA existantes ainsi que leurs caractéristiques générales, pour sélectionner celle qui correspondait le mieux à nos critères.

Nous rappelons ci-dessous les critères de sélection :

· Langage de programmation Java

· Possibilité d'écrire des agents de type application et *applet.*

· Communication supportant les actes de langage

· Prise en main rapide

Au vu de ces critères, notre choix s'est porté sur la plate-forme JADE. Notons que nous avons utilisé JADE 3[1].4.1.

- *JADE*

JADE (Java Agent DEvelopement framework) est une plate-forme multi-agent créé par le laboratoire TILAB [1] . JADE permet le développement de systèmes multi-agents et d'applications conformes aux normes FIPA [2]. Elle est implémentée en JAVA et fourni

des classes qui implémentent « JESS » pour la définition du comportement des agents. Toute la communication entre agents est exécutée par messages FIPA ACL.

JADE possède trois modules principaux (nécessaire aux normes FIPA).

- DF « Director Facilitor » fournit un service de « pages jaunes» à la plate-forme ;
- ACC «Agent Communication Channel » gère la communication entre les agents ;
- AMS « Agent Management System » supervise l'enregistrement des agents, leur authentification, leur accès et l'utilisation du système.

Ces trois modules sont activés à chaque démarrage de la plate-forme.

La plate-forme d'agent peut être répartie sur plusieurs serveurs. Une seule application Java, et donc une seule machine virtuelle de Java (JVM), est exécutée sur chaque serveur. Chaque JVM est un conteneur d'agents qui fournit un environnement complet pour l'exécution d'agent et permet à plusieurs agents de s'exécuter en parallèle sur le même serveur.

L'architecture de communication offre la transmission de messages flexibles et efficaces. JADE crée et contrôle une file d'attente des messages entrants pour chaque agent. Le modèle global de communication FIPA a été mis en application. Ses composants ont été distingués clairement et ont été entièrement intégrés: protocoles d'interaction, ACL, langues, schémas de codage, protocoles de transport...

Concrètement, un thread est lancé pour chaque agent, mais ces derniers doivent souvent exécuter des tâches parallèles. Avec la solution du multithreading offerte directement par Java, Jade supporte également la gestion des comportements coopératifs. Le run-time inclut également quelques fonctions complexes prêtes l'emploi pour les tâches les plus communes dans la programmation agent, comme des protocoles d'interaction de FIPA.

- *La norme FIPA*

La FIPA (Foundation for Intelligent Physical Agents) [2] est une organisation à but non lucratif fondée en 1996 dont l'objectif est de produire des standards pour l'interopération d'agents logiciels hétérogènes. Par la combinaison d'actes de langages, de logique des prédicats et d'ontologies publiques, la FIPA cherche à offrir des moyens

standardisés permettant d'interpréter les communications entre agents de manière à respecter leur sens initial, ce qui est bien plus ambitieux que XML, qui ne standardise que la structure syntaxique des documents. Afin d'atteindre ce but, le FIPA émet des standards couvrant :

- Les applications (applications nomades, agent de voyage personnel, applications de diffusion audiovisuelles, gestion de réseaux, assistant personnel…) ;
- Les architectures abstraites, définissant d'une manière générale les architectures d'agents ;
- Les langages d'interaction (ACL), les langages de contenu (comme SL, CCL, KIF ou RDF) et les protocoles d'interaction ;
- La gestion des agents (nommage, cycle de vie, description, mobilité, configuration);
- Le transport des messages : représentation (textuelle, binaire ou XML) des messages ACL, transport (par IIOP, WAP ou HTTP) de ces messages.

Ces standards évoluent, et sont régulièrement mis à jour, ainsi que de nouveaux standards qui sont nouvellement proposés. Les standards qu'édicte la FIPA ne constituent pas vraiment une plate-forme de construction multi-agents. Ce n'est pas non plus l'objectif que s'est fixé la FIPA. Tout au plus, la FIPA normalise une plate-forme d'exécution standardisée dans un but d'interopérabilité. Ces normes s'appliquent donc pour la plupart en phase de déploiement. Elles n'abordent pas les phases d'analyse ni de conception. Elles peuvent cependant guider certains choix d'implémentation.

5.3 Les structures de données utilisées pour le stockages

Nous utilisons le modèle conceptuel de données MCD de la méthode MERISE [28] pour présenter les différentes données et informations constituant les bases de données de notre système et les relations entre elles.

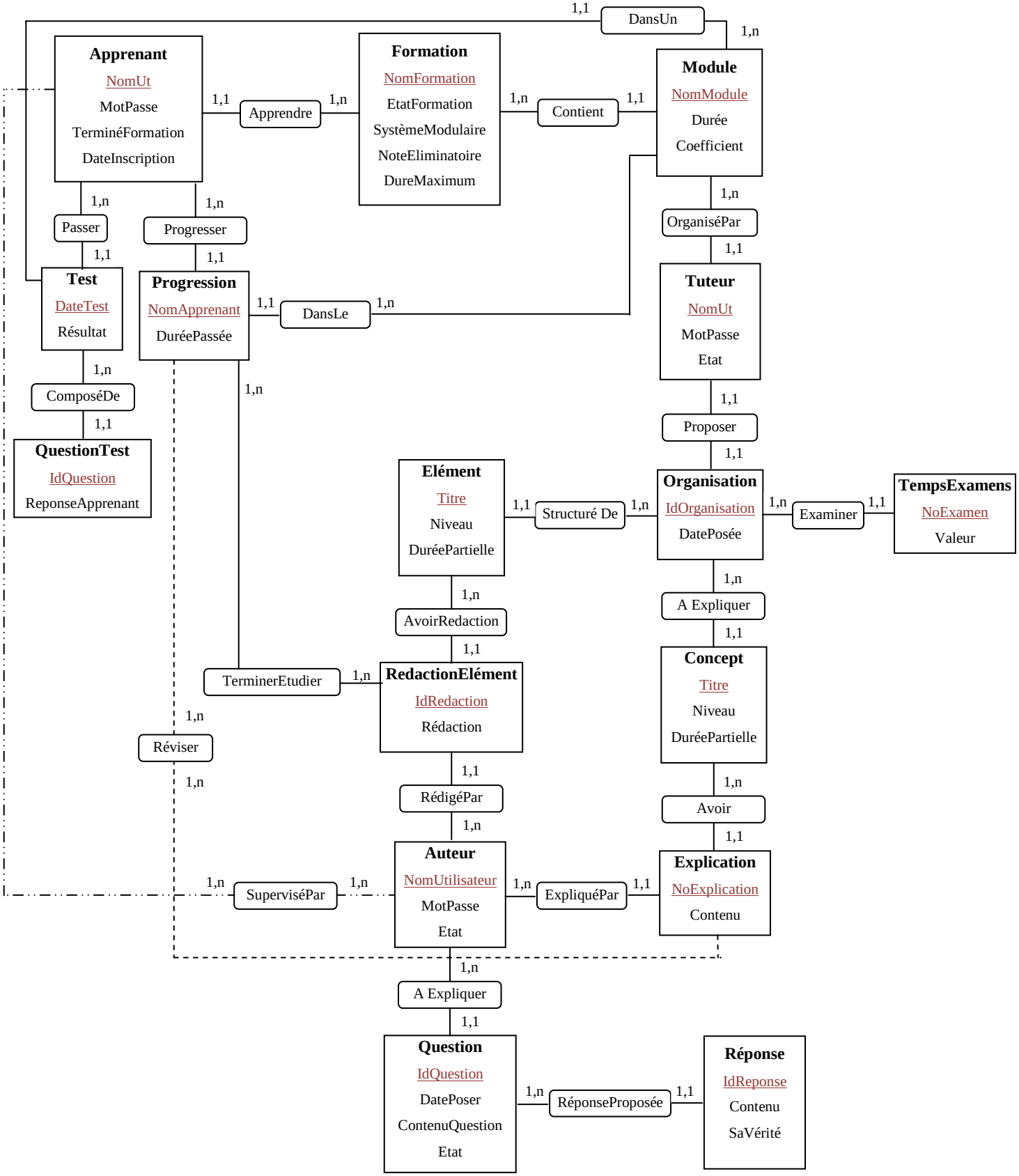


Figure (29) : Le MCD des données de stockages

5.4 Les classes JADE exploitées

Dans cette section nous allons citer et expliquer les classes JADE que nous avons utilisées dans notre implémentation.

Concernant la déclaration des différents types d'agents constituant notre système, deux classes JADE abstraites ont été utilisées pour déclarer les agents graphiques (ayant des interfaces utilisateurs) et ceux non graphiques :

1. La classe **jade.gui.GuiAgent** pour déclarer les agents d'interfaces.
2. La classe **jade.core.Agent** pour déclarer les agents non graphiques.

Selon nos besoins, nous avons choisi d'utiliser quatre types de comportements JADE à savoir :

♦ Le comportement cyclique : Un comportement s'exécute d'une manière contenue sans cesse durant toute la vie de l'agent.

♦ Le comportement OneShot**:** Un comportement s'exécute une seule fois.

♦ Le comportement périodique**:** Un comportement s'exécute périodiquement durant toute la vie de l'agent.

♦ Le comportement séquentiel**:** Un comportement composé d'un ensemble des sous comportements s'exécutent l'un après l'autres.

JADE fournit ces différents types de comportements sous forme des classes abstraites (voir le **Tableau (2)**) ce qui donne au programmeur la liberté de définir dans-t-elles les programmes qu'il veut.

Type de comportement	Son nom en JADE	Sa classe JADE
Cyclique	CyclicBehaviour	jade.core.behaviours.CyclicBehaviour;
OneShot	OneShotBehaviour	jade.core.behaviours.OneShotBehaviour;
Périodique	TickerBehaviour	jade.core.behaviours.TickerBehaviour;
Séquentiel	SequentialBehaviour	jade.core.behaviours.SequentialBehaviour;

Tableau (2) : Types de comportements utilisés

Concernant la communication entre agents, nous avons utilisé deux manières fournies par JADE [31]:

♦ La première consiste à utiliser les simples chaines de caractères pour représenter les contenus des messages. Donc, elle est commode dans le cas où le contenu du message est une donnée atomique.

♦ La deuxième façon exploite la technologie Java pour transmettre directement les objets Java de type *Serialized object* comme des contenus des messages. Les objets utilisés dans ce cas ont des classes java implémentée sous l'extension java.io.Serializable.

Les méthodes utilisées pour manipuler le contenu d'un message sont *set* et *get* contenu. Elles sont différées selon la manière utilisée pour définir le message comme il est montré dans le tableau suivant :

Type de contenu	Lire (Prendre) contenu	Ecrire (Poser) contenu
String	getContent()	SetContent()
Java Objects	getContentObject()	SetContentObject()

Tableau(3) : Communications entre notre agents

5.5 Les structures de données de communication entre agents et de représentation de leurs connaissances

Comme nous avons indiqué dans la section précédente, nous avons déclaré les structures de données représentant les contenus non atomiques des messages sous forme des classes Java implémentées dans l'interface *Serializable*. Ce type d'interface utilise les méthodes *get* et *set* pour la lecture et l'écriture des différents attributs de la classe.

Les classes utilisées dans notre implémentation sont les suivantes :

- Classe d'une réponse d'une question proposée par un auteur

```
public class Reponse_Proposee_Question implements Serializable
{
  String Contenu_Reponse;    // La réponse proposée
  boolean verite;            // Vraie ou Fausse
  public Reponse_Proposee_Question() {   }
  public void setContenu_Reponse(String Contenu_Reponse){ this.Contenu_Reponse = Contenu_Reponse;}
  public String getContenu_Reponse(){ return this.Contenu_Reponse;}
  public void setVerite(boolean verite){ this.verite = verite;}
  public boolean getVerite(){ return verite;}
}
```

- Classe d'une question d'un test

```
public class QuestionTest implements Serializable
{
  String Contenu;                  // Le contenu de la question
  String Id_Quest;                 // Identificateur de la question
  List Reponses = new ArrayList();     // Liste des réponses proposées
  public QuestionTest(){}
```

```
    public void setContenu(String Contenu) {   this.Contenu = Contenu;   }
    public String getContenu() {   return this.Contenu;   }
    public void setId(String Id_Quest)  {     this.Id_Quest = Id_Quest;   }
    public String getId()   {    return this.Id_Quest;   }
    public void setReponses( Reponse_Proposee_Question Reponses) {  this.Reponses.add(Reponses);  }
    public List getReponses() {   return Reponses;  }
  }
```

- Classe d'un test

```
public class Test implements Serializable
{   List Question = new ArrayList();                // Liste des questions constituent le test (l'examen)
    public Test(){}
    public void setQuestion(QuestionTest q)  {  this.Question.add(q);  }
    public List getQuestion()  {    return this.Question;  }
}
```

- Classe résultat d'un test dans un module / résultat obtenu dans un module au moyen des résultats des différents tests

```
public class ResultatModule implements Serializable
{  String Module;          // Le module considéré
   double Resultat;        // Le résultat obtenu au module courant
   boolean PasseTest;      // = vrai si l'apprenant a passé ou moins un test dans le module Module, = faux sinon
   public ResultatModule (){}
   public void setModule(String Module){ this.Module = Module;}
   public String getModule(){ return this.Module;}
   public void setResultat(double Resultat){ this.Resultat = Resultat;}
   public double getResultat(){ return this.Resultat;}
   public void setPasseTest(boolean PasseTest){ this. PasseTest = PasseTest;}
   public boolean getPasseTest(){ return this.PasseTest;}
}
```

- Classe bilan des résultats obtenus dans les différents modules

```
public class BilanResultats implements Serializable
{   List ResultatsModules = new ArrayList();        // Un bilan des résultats obtenus dans différents modules
    public BilanResultats (){}
    public void setResultatsModules (ResultatModule ResultatModule){ this.ResultatsModules.add(ResultatModule);}
    public List getResultatsModules (){ return this.ResultatsModules;}
}
```

- Classe d'un élément ou d'un concept constituant un cours

```
public class Element_Concept implements Serializable
```

```
{  String Titre;                  // Le titre de l'élément ou concept de cours
   String Type;                   // Son type: un élément ou un concept
   String Redaction ="";          // Sa rédaction
   double Taux_Temps;             // Son temps par apport
   int Niveau ;                   // Le niveau correspond
   public Element_Concept(){}
   public void setTitre(String Titre){ this.Titre = Titre;}
   public String getTitre (){ return this.Titre;}
   public void setRedaction (String Redaction){ this.Redaction = Redaction;}
   public String getRedaction (){ return this.Redaction;}
   public void setTaux_Temps (double Taux_Temps){ this.Taux_Temps = Taux_Temps;}
   public double getTaux_Temps (){ return this.Taux_Temps;}
   public void setNiveau (int Niveau){ this.Niveau = Niveau;}
   public int getNiveau (){ return this.Niveau;}
}
```

- Classe d'une organisation de cours

```
public class Organisation implements Serializable
{ String Id_Org;                                  // Identificateur de l'organisation
   List Element_Concept_Cours = new ArrayList();  // Liste des constituants de cours
   List Examens = new ArrayList();                // Liste des temps des examens
   public Organisation (){}
   public void setId_Org (String Id_Org){ this.Id_Org = Id_Org;}
   public String getId_Org (){ return this.Id_Org;}
   public void setElement_Concept_Cours (ElementConcept ec){ this.Element_Concept_Cours.add(ec);}
   public List getElement_Concept_Cours (){ return this.Element_Concept_Cours;}
   public void setExamens (double Temps_Examen){ this.Examens.add(Temps_Examen);}
   public List getExamens (){ return this.Examens;}
}
```

- Classe Tuteur / Auteur d'un module

```
public class Enseignant implements Serializable
{ String NomUt;                  // Nom de l'enseignant
   String Operation;             // Inscription d'un nouveau enseignant ou ouvrir une session d'un enseignant déjà inscrit
   String Formation;             // La formation choisie
   String Module;                // Le module choisi
   Organisation Org;             // L'organisation proposée (cas d'un tuteur) ou l'organisation suivie (un auteur)
   Time DureMod;                 // La durée spécialisée au module
   public void setNomUt(String NomUt){ this.NomUt = NomUt;}
   public String getNomUt(){ return this.NomUt;}
   public void setOperation (String Operation){ this.Operation = Operation;}
   public String getOperation(){ return this.Operation;}
```

```
    public void setFormation(String Formation){ this.Formation = Formation;}
    public String getFormation(){ return this.Formation;}
    public void setModule(String Module){ this.Module = Module;}
    public String getModule(){ return this.Module;}
    public void setDureMod (String DureMod){ this.DureMod = DureMod;}
    public String getDureMod (){ return this.DureMod;}
    public void setOrg (Organisation Org){ this.Org = Org;}
    public Organisation getOrg (){ return this.Org;}
}
```

- Classe définie un Apprenant

```
public class Apprenant implements Serializable
{ String NomUt;                          // Nom utilisateur de l'apprenant
   String Formation;                     // la formation qu'il apprend
   Date DateInscription, DateFinFormation;     // La date d'inscription et la date de la fin de formation
   public void setNomUt(String NomUt){ this.NomUt = NomUt;}
   public String getNomUt(){ return this.NomUt;}
   public void setFormation(String Formation){ this.Formation = Formation;}
   public String getFormation(){ return this.Formation;}
   public void setDateInscription (Date DateInscription){ this.DateInscription = DateInscription;}
   public Date getDateInscription (){ return this.DateInscription;}
   public void setDateFinFormation (Date DateFinFormation){ this.DateFinFormation = DateFinFormation;}
   public Date getDateFinFormation (){ return this.DateFinFormation;}
}
```

- Classe un point de cours terminé (Etudié par l'apprenant)

```
public class PointEtudie implements Serializable
{ String Type;                    // Un élément ou un concept
  String Titre;                   // son titre
  String Source;                  // la source (l'auteur)
  int NoExplication;              // Numéro d'explication pour les concepts
  public void setType(String Type){ this.Type = Type;}
  public String getType(){ return this.Type;}
  public void setTitre(String Titre){ this.Titre = Titre;}
  public String getTitre(){ return this.Titre;}
  public void setSource(String Source){ this.Source = Source;}
  public String getSource(){ return this.Source;}
  public void setNoExplication(int NoExplication){ this.NoExplication = NoExplication;}
  public int getNoExplication(){ return this.NoExplication;}
}
```

- Classe Historique d'un Test passé l'Apprenant

```
public class Hist_Test implements Serializable
{ double Resultat;                                  // Le résultat obtenu dans un test
  List Question_Reponse_Ap = new ArrayList();       // La liste des questions constituant ce test
  public void setQuestion_Reponse_Ap (String IdQuestion, boolean ReponseAp)
  {  struct Q{ String IdQuestion ; boolean ReponseAp ; } Q_T ;
     Q_T.IdQuestion = IdQuestion;    Q_T.ReponseAp = ReponseAp;
     this.Question_Reponse_Ap.add((Object) Q_T);}
  public List getQuestion_Reponse_Ap (){ return this.Question_Reponse_Ap;}
  public void setResultat(double Resultat){ this.Resultat = Resultat;}
  public double getResultat(){ return this.Resultat;}
}
```

- Classe résume les informations sur l'apprenant dans un module

```
public class InfMod implements Serializable
{
  String NomModule, NomTuteur, NomAuteur;  // Le nom de module, le tuteur et l'auteur que l'apprenant suit
  String TypeTest;                         // Demande d'un Court test ou un test Complet
  Time DurePasse, DureMax;                 // La durée que l'apprenant a passée dans ce module, et la durée maximum
  List Prog = new ArrayList();             // La liste des points de cours que l'apprenant a terminé de les étudier
  List HistTests = new ArrayList();        // Liste de ses différents tests dans le module
  public void setNomModule(String NomModule){ this.NomModule = NomModule;}
  public String getNomModule(){ return this.NomModule;}
  public void setNomTuteur(String NomTuteur){ this.NomTuteur = NomTuteur;}
  public String getNomTuteur(){ return this.NomTuteur;}
  public void setNomAuteur(String NomAuteur){ this.NomAuteur = NomAuteur;}
  public String getNomAuteur(){ return this.NomAuteur;}
  public void setTypeTest(String TypeTest){ this.TypeTest = TypeTest;}
  public String getTypeTest(){ return this.TypeTest;}
  public void setDureMax(Time DureMax){ this.DureMax = DureMax;}
  public Time getDureMax(){ return this.DureMax;}
  public void setDurePasse(Time DurePasse){ this.DurePasse = DurePasse;}
  public Time getDurePasse(){ return this.DurePasse;}
  public void setProg (PointEtudie Point )  { this.Prog.add(Point);}
  public List getProg(){ return this.Prog;}
  public void setHistTests (Hist_Test Test)   { this.HistTests.add(Test);}
  public List getHistTests(){ return this.HistTests;}
}
```

5.6 Les agents

5.6.1 L'*Agent Hôte*

Il est un agent d'interface. Donc, il est considéré selon JADE un agent graphique de type **GuiAgent**.

```
public class AgentHote extends GuiAgent {

protected void onGuiEvent(GuiEvent ev)   // Méthode abstraite de l'agent graphique s'exécute à chaque
                                         //   réception d'un événement graphique.

{  String s = (String) ev.getParameter(0);

   ACLMessage toSend = new ACLMessage(ACLMessage.INFORM);        // Transformer les événements à
                                                                 // des messages et les envoyer à
   toSend.setContent(s);                                         // l'Agent Hôte lui même

   toSend.setPerformative(ACLMessage.INFORM);

   toSend.addReceiver(new AID(this.getName(), AID.ISLOCALNAME));

   send(toSend);  }

public void setup()   {            // Méthode abstraite de l'agent JADE s'exécute à la création de l'agent

  CyclicBehaviour cb = new PrincipalBehaviour(this);     // L'Agent Hôte contient un comportement cyclique pour
                                                         // contrôler tous les événements de l'utilisateur à n'importe

   addBehaviour(cb);     // Ajouter le comportement à la base des comportements de
                         // l'agent.
} }
```

Le comportement principal de l'*Agent Hôte* rassemble tous ses actions et ses autres comportements. Ce comportement s'exécute d'une manière continue durant toute la vie de l'agent. Il est de type **CyclicBehaviour** de la classe abstraite **jade.core.behaviours.CyclicBehaviour**. La classe suivante lui présente.

```
class PrincipalBehaviour extends CyclicBehaviour

{   // Le constructeur a comme paramètre un agent graphique

  public PrincipalBehaviour(GuiAgent ga)  {      super(ga);  }
```

```
public void action()    // Méthode abstraite s'exécute à la création de comportement
{  ACLMessage reply = receive(); // Le message reçu par l'agent
   if(reply != null)  {   // Le message non vide, càd ; l'agent reçoit un message
   AID Emeteur = reply.getSender();  // L'émetteur du message
   if(Emeteur.equals(this.myAgent.getName()))          { // L'Agent lui-même. Dans ce cas il traite les  événements graphiques de l'utilisateur
    if(reply.getContent().equals("To Recognize What The System Provides."))
      addBehaviour(new Afficher(this.myAgent,"Visit"));    // Ajouter le comportement Afficher(…,…) (1)
   else {
     if(reply.getContent().equals("Informations Du Module Choisi"))
        addBehaviour(new InfModuleChoisi(this.myAgent));  // Ajouter le comportement InfModuleChoisi (…,…) (2)
     else {
       if(reply.getContent().equals("Show Available Organisations"))
          addBehaviour(new AfficherOrganisationsModule(this.myAgent, "Show The Following Organisation","Visit")); (3)
       else {
        if(reply.getContent().equals("Show The Following Organisation"))
           addBehaviour(new AfficherOrganisationsModule(this.myAgent, "Show The Following Organisation","Visit"));
        else {
          if(reply.getContent().equals("Exit.")) this.myAgent.doDelete();    // détruire l'agent
         else {
           if(reply.getContent().equals("To Log In (Old User).")) … des operations graphiques…
            else {
              if(reply.getContent().equals("Log In"))
                 addBehaviour(new Identification(this.myAgent,IdOldUser.Etat.getSelectedItem(),IdOldUser.NomUt.getText())); (4)
              else {
                if(reply.getContent().equals("To Sign In (New User Of Learner Kind)."))
                   addBehaviour(new Afficher(this.myAgent,"Inscription Apprenant"));
                else {
```

```
if(reply.getContent().equals("Module Suivant"))
    addBehaviour(new AfficherOrganisationsModule(this.myAgent, "Show The Following Organisation",
                                    "Inscription Apprenant"));
else {
  if(reply.getContent().equals("Following Organization"))
    addBehaviour(new AfficherOrganisationsModule(this.myAgent, "Show The Following Organisation",
                    "Inscription Apprenant"));
  else {
    if(reply.getContent().equals("Choose The Current Organization"))
        addBehaviour(new AjoutOrgChoixNouvAp(this.myAgent)); (5)
    else {
      if(reply.getContent().equals("Module Choix Suivant"))
        addBehaviour(new AfficherOrganisationChoix(this.myAgent)); (6)
      else {
        if(reply.getContent().equals("Submit"))
          addBehaviour(new Inscription_Ap_Mod(this.myAgent)); (7)
        else {
          if(reply.getContent().equals("To Sign In (New User Of Author Kind)."))
            addBehaviour(new Afficher(this.myAgent,"Inscription Author"));
          else {
            if(reply.getContent().equals("View Following Organization"))
              addBehaviour(new AfficherOrganisationsModule(this.myAgent, "Show The Following
Organisation","Inscription Author"));
            else {
              if(reply.getContent().equals("Next Module In The Author Case"))
addBehaviour(new AfficherOrganisationsModule(this.myAgent, "Show The Following Organisation","Inscription Author"));
              else {
                if(reply.getContent().equals("Submit In The Current Organization"))
```

```
{ SequentialBehaviour seq = new SequentialBehaviour(this.myAgent);
  seq.addSubBehaviour(new Accept_NAuteur(this.myAgent));
  seq.addSubBehaviour(new OneShotBehaviour(this.myAgent)
  { public void action()  {
      … // des opérations et interactions graphiques avec l'utilisateur…
        // Cas d'un nouveau auteur ➔ création d'un Agent Auteur
   String name = "Auteur" ;
    AgentContainer c = getContainerController();
    try {
    AgentController AgAut = c.createNewAgent( name, "Projet.AgentAuteur", null );
     AgAut.start();
     Enseignant Auteur;  // … Collecter les informations de l'auteur dans Auteur
                // Envoi un message (Structure de données) à l'Agent Auteur
     ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
     msg.setLanguage("JavaSerialization");
     msg.setContentObject(Auteur);
     msg.setPerformative(ACLMessage.INFORM);
     msg.addReceiver(new AID( name, AID.ISLOCALNAME ));
     send(msg); }
    catch (Exception e){e.printStackTrace();}}});
   addBehaviour(seq); // Ajouter ce comportement cyclique (8) }
 else {
  if(reply.getContent().equals("To Sign In (New User Of Tutor Kind)."))
   addBehaviour(new Afficher(this.myAgent,"Inscription Tutor"));
  else {
   if(reply.getContent().equals("Submit In The Current Module"))
   { SequentialBehaviour seq = new SequentialBehaviour(this.myAgent);
```

```
                                    seq.addSubBehaviour(new Accept_NTuteur(this.myAgent));
                                    seq.addSubBehaviour(new OneShotBehaviour(this.myAgent) {
                                    public void action() { // Cas d'un nouveau tuteur
                                        // … Idem que un nouveau auteur (9) }
                                    addBehaviour(seq); }
        }}}}}}}}}}}}}}}}}
else      // Cas d'un autre émetteur de message  {
 if(Emeteur.equals(Auteur)) {      // L'Emetteur est un Agent Auteur. Il est créé par l'Agent Hôte
  if(reply.getLanguage().equals("JavaSerialization")){      // Le message reçu est une structure de données complexe de
                                                            // type java.io.Serializable
    Enseigant Auteur ;
    try{
     if(reply.getContentObject().getClass().equals(Auteur.getClass())){   // Le message reçu est un objet de type
                                                                           // Enseigant
        Auteur  = (Enseigant)reply.getContentObject();
        addBehaviour(new OrgChoisieAuteur(this.myAgent, Auteur)); // (10)  }
    catch (Exception e){e.printStackTrace();}}
 else{ // Le message est une simple chaine de caractères
  if(reply.getContent().equals("Author Don't Respect Inscription Procedures"))      //… Informer l'auteur … }}
else {     // L'émetteur n'est pas un Agent Auteur
 if(Emeteur.equals(Tuteur)) {      // L'Emetteur est un Agent Organisateur. Il est créé par l'Agent Hôte
  if(reply.getLanguage().equals("JavaSerialization")){      // Le message reçu est une structure de données complexe de
                                                            // type java.io.Serializable
    Enseigant Tuteur ;
    try{
     if(reply.getContentObject().getClass().equals(Tuteur.getClass())){   // Le message reçu est un objet de type
                                                                           // Enseigant()
        Tuteur  = (Enseigant)reply.getContentObject();
        addBehaviour(new OrgProposeeTuteur(this.myAgent, Tuteur)); // (11)  }
    catch (Exception e){e.printStackTrace();}}
 else{ // Le message est une simple chaine de caractères
```

```
if(reply.getContent().equals("Tutor Don't Respect Inscription Procedures"))       //… Informer le tuteur … }}
} } }
else   block();    // Pas de message   } // Fin de l'action }  // Fin de comportement
```

Dans ce qui suit, nous expliquons la signification des comportements utilisés :

N°	Son type	Le constructeur	Signification
1	OneShotBehaviour	Afficher(Agent a,String Type_Utilisateur)	Afficher à l'utilisateur les informations du système selon son type
2	OneShotBehaviour	InfModuleChoisi(Agent a)	Afficher les informations du module courrant
3	OneShotBehaviour	AfficherOrganisationsModule(Agent a,String Type_Ut)	Afficher la prochaine organisation du module courant selon le type d'utilisateur
4	OneShotBehaviour	Identification(Agent a ,String NomUtilisateur)	Identification d'un utilisateur enrégistré
5	OneShotBehaviour	AjoutOrgChoixNouvAp(Agent a)	Ajouter/Effectuer l'organisation choisie à l'apprenant
6	OneShotBehaviour	AfficherOrganisationChoix(Agent a)	Afficher les organisations choisies
7	OneShotBehaviour	Inscription_Ap_Mod(Agent a)	Inscription effective de l'apprenant
8	SequentialBehaviour	Vérifier l'acceptation de l'auteur puis créer un Agent Auteur et l'envoi un message contient l'organisation suivie	
9	SequentialBehaviour	Vérifier l'acceptation du tuteur puis créer un Agent Organisateur et l'envoi un message contient le module choisi	
10	OneShotBehaviour	OrgChoisieAuteur(Agent a, Enseignant Auteur)	Inscription effective de l'auteur
11	OneShotBehaviour	OrgProposeeTuteur(Agent a, Enseignant Tuteur)	Inscription effective du tuteur

Tableau (4) : Comportements d'*Agent Hôte*

5.6.2 L'*Agent Apprenant*

Il est aussi un agent graphique. Il se compose de trois comportements principaux : un périodique et deux cycliques. Ses connaissances locales contiennent toutes informations sur l'apprenant et sa progression dans tous les modules.

```
public class Agent_Apprenant extends GuiAgent
{ Apprenant apprenant ;
  List Modules = new ArrayList() ;          // Chaque élément de cette liste est de type InfMod
 protected void onGuiEvent(GuiEvent ev)  // Idem que l'Agent Hôte
 {
   String s = (String) ev.getParameter(0);
   ACLMessage toSend = new ACLMessage(ACLMessage.INFORM);
   toSend.setContent(s);
   toSend.setPerformative(ACLMessage.INFORM);
   toSend.addReceiver(new AID(this.getName(), AID.ISLOCALNAME));
   send(toSend);
 }
  protected void setup()
```

```
  { CyclicBehaviour cb1 = new PrincipalBehaviour(this); // Un comportement cyclique traite les messages reçus
                    addBehaviour(cb1);
     addBehaviour( new TickerBehaviour( this, 10000 ){ // Exécuter chaque 10000 ms le comportement (a)
        protected void onTick()  {
           addBehaviour(new Verifier_Progression(this.myAgent)); // (a)
        } });
    CyclicBehaviour cb2 = new Controler_Temps(this); // (b)
                    addBehaviour(cb2);
  }
}
```

La classe suivante présente le comportement responsable de la communication et traitement des messages.

```
class PrincipalBehaviour extends CyclicBehaviour{
public PrincipalBehaviour(GuiAgent ga){ super(ga); }
  public void action() {
    ACLMessage reply = receive();
    if(reply != null) {
     AID Emetteur = reply.getSender() ; // L'Emetteur du message
     if( Emetteur.equal(this.myAgent.getName())) // L'agent lui même
     {  // Traitement des événements graphiques de l'interface apprenant … voir le Tableau (6)}
     else { // L'Emetteur du message est un autre agent
       if(Emetteur.equal(Hote)) { // L'Emetteur du message l'Agent Hôte
         SequentialBehaviour seq = new SequentialBehaviour(this.myAgent);
         seq.addSubBehaviour(new OneShotBehaviour(this.myAgent) {
            public void action() { // Réception d'un message contient les informations de l'apprenant ouvrant la session
              if( ("JavaSerialization").equals(reply.getLanguage())) {
                try {
                  if(reply.getContentObject().getClass().equals(apprenant.getClass())) {
                    apprenant = (Apprenant) reply.getContentObject(); } } }
                catch (Exception ex) { ex.printStackTrace(); } } });
         seq.addSubBehaviour(new ChargerConnaissances(this.myAgent)); (i).
         addBehaviour(seq);
       }
       else {    // L'Emetteur du message l'Agent Testeur
         if( ("JavaSerialization").equals(reply.getLanguage())) {
           Test TestRecu;
           try
           { if(reply.getContentObject().getClass().equals(TestRecu.getClass())) { // Le cas d'un message contient un test
              TestRecu = (Test)reply.getContentObject();
              addBehaviour(new Sauvegarde_Progression(this.myAgent)); (c)
              addBehaviour(new Afficher_Question(this.myAgent, "First Question")); (d) }
              else {
```

```
                ResultatModule ResultatRecu;
                if(reply.getContentObject().getClass().equals(ResultatRecu.getClass()))
                { // Le cas d'un message contient le résultat de dernier test
                 ResultatRecu = (ResultatModule) reply.getContentObject();
                 addBehaviour(new Affiche_Resultat_Correction_Test(this.myAgent)); (e) }
                else {   // Cas d'un message contient un bilan des résultats obtenus en différents modules
                    BilanResultats Resultats = (BilanResultats) reply.getContentObject();
                    addBehaviour(new AfficherBillan(this.myAgent)); (f)
                    }
        } } }
            catch (Exception ex)
            { System.err.println(getLocalName() + "       ");
              System.err.println( ex.getMessage());
            } }
           else          // Cas d'un message simple
             if(reply.getContent().equal("Impossible de preparer un test"))
                    // … Informer l'apprenant s'il a demandé personnellement de passer un test …
           } } }
    else block(); // Pas de message }}
```

Les comportements utilisés sont expliqué dans le tableau suivant (**Tableau (5)**)

N°	Son type	Le constructeur	Signification
a	OneShotBehaviour	Verifier_Progression(Agent a)	Verifier la progression de l'apprenant et lui donne des remarques. S'il est trop tard il va subir un court test (g)
b	CyclicBehaviour	Controler_Temps(Agent a)	* Afficher à l'apprenant sa progression de temps * Verifier s'il declanche l'un des temps des examens proposés (g) * Controler le temps restant dans le courant module
c	OneShotBehaviour	Sauvegarde_Progression(Agent a)	Sauvegarder la progression de l'apprenant dans le module
d	OneShotBehaviour	Afficher_Question(Agent a, String Q)	Affiche la question Q du test
e	OneShotBehaviour	AfficherResultatCorrectionTest(Agent a)	Afficher le resutat obtenu avec le corrigé type
f	OneShotBehaviour	AfficherBillan(Agent a)	Afficher un bilan des resultats et progression dans tous les modules
i	OneShotBehaviour	ChargerConnaissances(Agent a)	Charger ses connaissances locales (*apprenant, Modules*)

Tableau (5): Comportements de l'*Agent Apprenant*

Les autres comportements sont liés à l'interface de l'apprenant. Nous les citons dans le **Tableau (6)**. Tous ces comportements sont de type *OneShotBehaviour*.

L'événement	Le comportement	La signification
To Choose Module	AfficheModule(Agent a)	Afficher les modules de la formation suivie
Change Module	Inf_Ap_Mod(Agent a)	Afficher les informations de l'apprenant dans ce module

Show Course	Aff_Cours_Element(Agent a)	Afficher le cours de l'élément sélectionné
Finalize Element	(c)	
Demand Test	(g)	
Pass To Next Question Go Back To Previous Question Add The Current Response Delete The Current Response	(d)	- Afficher la question suivante du test - Afficher la question précédente du test - Ajouter la réponse sélectionnée aux réponses de l'apprenant - Supprimer la réponses sélectionnée
I Terminate The Test	(h)	
Show Responses Of Previous Question Show Responses Of Next Question	Affiche_Correction_Test(Agent a, String Question)	Afficher les réponses de la question suivante ou précédente
Show My Outcomes	(j)	Envoyer message à l'*Agent Testeur* pour construire un bilan des résultats obtenus par l'apprenant

Tableau (6): Suite de comportements de l'*Agent Apprenant*

(g): *L'Agent Apprenant* demande de l'*Agent Testeur* de préparer un test ;

```
try {  InfModule ProgApMod;   // Envoyer la progression de l'apprenant dans le module de test à l'Agent Testeur
       ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
       msg.addReceiver(Testeur);
       msg.setPerformative( ACLMessage.INFORM );
       msg.setLanguage("JavaSerialization");
       msg.setContentObject(ProgApMod);
       send(msg); }
       catch (Exception ex) { ex.printStackTrace(); }
```

(h) L'*Agent Apprenant* envoi les réponses de l'apprenant vers l'*Agent Testeur* ;

```
try { Test ReponseApprTestRecu;   // Envoyer la réponses de l'apprenant à l'Agent Testeur
      ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
      msg.addReceiver(Testeur);
      msg.setPerformative( ACLMessage.INFORM );
      msg.setLanguage("JavaSerialization");
      msg.setContentObject(ReponseApprTestRecu);
      send(msg); }
catch (Exception ex) { ex.printStackTrace(); }
```

(j) L'*Agent Apprenant* envoi une demande de préparation d'un bilan de résultats des différents modules vers l'*Agent Testeur* ;

```
try {  ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
```

```
            msg.addReceiver(Testeur);
            msg.setPerformative( ACLMessage.INFORM );
            msg.setContent("Demande Bilan Resultats");
            send(msg); }
        catch (Exception ex) { ex.printStackTrace(); }
```

5.6.3 L'*Agent Testeur*

Il est un agent non graphique puisqu'il a seulement des communications avec l'*Agent Apprenant*. Il a les informations de l'apprenant (envoyées par l'*Agent Hôte*) et son historique des tests et résultats dans tous les modules.

```
public class Agent_Testeur extends Agent {
Apprenant apprenant;
List HistTestsModules = new ArrayList();
                    // Les autres variables utilisées
InfModule ProgAp; Test TestEnv, RepTestRecu;   ResultatModule ResultatDernierTest; BilanResultats Bilan;
protected void setup() {
   addBehaviour(new CyclicBehaviour(this) {
    public void action() {
      ACLMessage msg = receive();
      if (msg != null) {
       if( ("JavaSerialization").equals(msg.getLanguage())) {
         if(reply.getContentObject().getClass().equals(ProgAp.getClass())) {
          SequentialBehaviour seq = new SequentialBehaviour();
          seq.addSubBehaviour( new Composer_Test(this.myAgent, ProgAp.NomModule, ProgAp.TypeTest) ); (α)
          seq.addSubBehaviour( new OneShotBehaviour(this.myAgent)
         {  public void action() {
             try {
                ACLMessage reply = new ACLMessage(ACLMessage.INFORM);
                reply.addReceiver(msg.getSender());
                reply.setPerformative( ACLMessage.INFORM );
                reply.setLanguage("JavaSerialization");
                reply.setContentObject(TestEnv);     // Envoyer le test à l'Agent Apprenant
                send(reply);    }
             catch (Exception ex) { ex.printStackTrace(); }  }  });
         addBehaviour( seq ); }
       else {
         if(reply.getContentObject().getClass().equals(TestEnv.getClass())) {
                                                          // Le cas d'un message contient les réponses de
                                                          // l'apprenant
          RepTestRecu = (Test)msg.getContentObject();
          SequentialBehaviour seq = new SequentialBehaviour();
          seq.addSubBehaviour( new  Correction_Test(this.myAgent));  (β)
```

```
seq.addSubBehaviour( new OneShotBehaviour(this.myAgent)
{ public void action()
  { try
    { ACLMessage reply = new ACLMessage(ACLMessage.INFORM);
      reply.addReceiver(msg.getSender());
      reply.setPerformative( ACLMessage.INFORM );
      reply.setLanguage("JavaSerialization");          // Envoyer le résultat de l'apprenant dans le dernier test
      reply.setContentObject(ResultatDernierTest);     // à l'Agent Apprenant
      send(reply);  }
     catch (Exception ex) { ex.printStackTrace(); } } });
    addBehaviour( seq );  }
  else {
    if(reply.getContentObject().getClass().equals(apprenant.getClass()))   // Message reçu de l'Agent Hote contient les
    {  apprenant = (Apprenant)msg.getContentObject();                       // informations de l'apprenant ouvrant la
      addBehaviour( new chargerConnaissancesLocales(this.myAgent));         // session
    } }      } }
 else { // Le cas d'un message simple
   if(msg.getContent().equals((String)"Demande Bilan Resultats")) {
    SequentialBehaviour seq = new SequentialBehaviour();
   seq.addSubBehaviour( new  Billan_Resultats(this.myAgent));  (γ)
   seq.addSubBehaviour( new OneShotBehaviour(this.myAgent)
   { public void action()
    { try
      { ACLMessage reply = new ACLMessage(ACLMessage.INFORM);
        reply.addReceiver(msg.getSender());
        reply.setPerformative( ACLMessage.INFORM );
        reply.setLanguage("JavaSerialization");
        reply.setContentObject(Bilan);                 // Envoyer le bilan des résultats de l'apprenant
        send(reply);  }
       catch (Exception ex) { ex.printStackTrace(); } } });
   addBehaviour( seq );   }  }   }
else block(); } }
```

Les comportements signés ci-dessus ((α), (β), (γ)) sont de type OneShotBehaviour (voir le **Tableau (7)**)

N°	Le constructeur	La signification
(α)	Composer_Test(Agent a,String Module, String Type)	Preparer un test de type *Type* ("court", "complet") dans le module *Module*
(β)	Correction_Test(Agent a)	Corriger les réponses de l'apprenant
(γ)	Billan_Resultats(Agent a)	Faire un bilan des résultats obtenus par l'apprenant

Tableau (7): comportements de l'*Agent Testeur*

5.6.4 L'*Agent Auteur*

Ce type d'agent est un agent graphique. Il se comporte dépendamment aux comportements de l'auteur qui lui associé. Il a aussi quelques communications avec l'*Agent Hôte*.

Sa base de comportements se compose essentiellement d'un comportement cyclique traite les différents messages reçus dus aux événements de l'interface auteur ou envoyés par l'*Agent Hôte*.

```
public class AgentAuteur extends GuiAgent
{  Enseignant Auteur ;
   protected void onGuiEvent(GuiEvent ev) { // Idem à celles de l'Agent Hôte et l'Agent Apprenant }
   public void setup()
   {  CyclicBehaviour cb = new PrincipalBehaviour(this);
      addBehaviour(cb);
   } }

class PrincipalBehaviour extends CyclicBehaviour
{
  public PrincipalBehaviour(GuiAgent ga) { super(ga); }
  public void action() {
    ACLMessage reply = receive();
   if(reply != null)
   {  AID emetteur = reply.getSender();// L'Emetteur du message
     if( Emetteur.equal(this.myAgent.getName())) // L'agent lui même
     {  // Traitement des événements graphiques de l'interface apprenant … voir le Tableau (8) }
     else { // L'Emetteur du message est un autre agent (l'Agent Hôte)
       if(reply.getLanguage().equals("JavaSerialization"))
       { try
         {  if(reply.getContentObject().getClass().equals(Auteur.getClass())) {
              Auteur = (Enseignant) reply.getContentObject();
              if( Auteur.getOperation.equal("Inscription New Author")) // Pour suivre le nouveau auteur
                addBehaviour(new Interface_Inscription(this.myAgent)); (A)
            else // Ouvrir la session d'un auteur ancien
              addBehaviour(new Initiation_Session(this.myAgent)); (B)  } }
        catch (Exception ex) { ex.printStackTrace(); } } } }
    else block() ; } }
```

Les deux comportements **(A)** et **(B)** sont de type OneShotBehaviour. Ils consistent de préparer l'interface de l'auteur selon son type (respectivement un nouveau auteur ou dans le cas où il a compte).

Le **Tableau (8)** contient les autres comportements de l'*Agent Auteur* dont ils dépendent aux événements graphiques.

L'événement	Le comportement	La signification
"Ok Current Element Redcation"	Add_Red_Element_Concept(Agent a, String Type)	Ajouter l'élément courant à la rédaction de l'auteur en l'effectuant son type Type ('Elément' ou 'Concept').
"Ok Current Concept Redcation"		
"Ok All"	**(C)**	
"==> Following Learner."	Resultat_App(Agent a)	Afficher le résultat de l'apprenant sélectionné.
" (*) Show The Following Question." 4	Gerer_Question(Agent a, int Operation)	Gestion et manipulation des questions posées par l'auteur. Le paramètre *Opération* signifie le type d'opération à effectuer sur la question. 1 : Ajouter la nouvelle question à *BDQuestion*. 2 : Supprimer la question sélectionnée de la *BDQuestion*. 3 : Calculer le pourcentage des réponses justes à la question selectionnée. 4 : Afficher la question suivante de l'auteur. 5 : Ajouter la nouvelle réponse proposée par l'auteur à la nouvelle question. 6: Ajouter l'élément courant à la liste des thèmes de la nouvelle question. 7: Ajouter le concept courant à la liste des thèmes de la nouvelle question. 8: supprimer le thème courant (sélectionné) 9: supprimer la réponse courante (sélectionnée)
"(*)Get The Current Question Percentage." 3		
" (*) Delete The Current Question." 2		
" Add The Current Response." 5		
" Add The Current Element As Question Theme." 6		
" Add The Current Concept As Question Theme." 7		
" Delete The Current Theme."8		
" Delete The Current Response."9		
" Add The Current Question."1		

Tableau (8) : comportements de l'*Agent Auteur*

(C): Un comportement séquentiel composé de trois autres comportements ;

```
SequentialBehaviour seq = new SequentialBehaviour(this.myAgent);
seq.addSubBehaviour(new Verifier_Redaction(this.myAgent)); (D)
seq.addSubBehaviour(new Verifier_Questions(this.myAgent)); (E)
```

```
seq.addSubBehaviour(new OneShotBehaviour(this.myAgent)
{ public void action() {
   try {
     ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
     String name = "Hote" ;
     AID Hote = new AID( name, AID.ISLOCALNAME );
     msg.addReceiver(Hote);
     msg.setLanguage("JavaSerialization");          // envoyer la rédaction de l'auteur à l'Agent Hôte
     msg.setContentObject(Auteur);
     send(msg); }
   catch (Exception ex) { ex.printStackTrace(); }   }  });
 addBehaviour(seq);
```

(D) et (E) sont deux comportements de type OneShotBehaviour. Le premier consiste à vérifier si l'auteur suit correctement l'organisation. Le deuxième insiste sur l'auteur de poser le nombre minimum des questions.

5.6.5 L'*Agent Organisateur*

Cet agent est le même que le précédant sauf pour ce qui concerne les comportements associés aux événements graphiques reçus de l'interface de tuteur (les opérations qu'il peut faire).

5.6.6 L'*Agent Administrateur*

Il est un agent graphique ayant une interface permet à l'administrateur (personne) de réaliser et de bien faire ses différentes opérations. Il est créé par l'*Agent Hôte* et n'a pas d'autres communications (messages) avec lui. Tous ses comportements sont liés aux activités de l'utilisateur (administrateur). Donc, sa méthode setup() est composée seulement d'un comportement cyclique traitre les différents messages réflexives (envoyés par l'agent lui-même) corresponds aux événements graphiques.

5.4 Conclusion

La meilleure manière pour valider une conception d'un système est de l'implémenter en utilisant les outils disponibles et adéquats au problème posé.

Dans cet chapitre, nous avons présenté les principaux points de l'implémentation. Nous avons commencé par l'argumentation de l'utilisation de java comme un langage de

programmation et JADE comme une plateforme Muti-Agent. Ensuite, les structures de données utilisées sont décrites en MERISE (MCD). Finalement, nous avons expliqué les classes d'agents composant notre système et leurs différents comportements ainsi leurs communications entre eux ou avec l'utilisateur dans le cas d'un agent graphique.

Après avoir clarifié la méthode d'implémentation de notre système, il nous reste à donner une idée de quoi il a l'aire c'est-à-dire sa réalisation, elle fera l'objet du chapitre suivant.

Chapitre 6

Résultats et tests

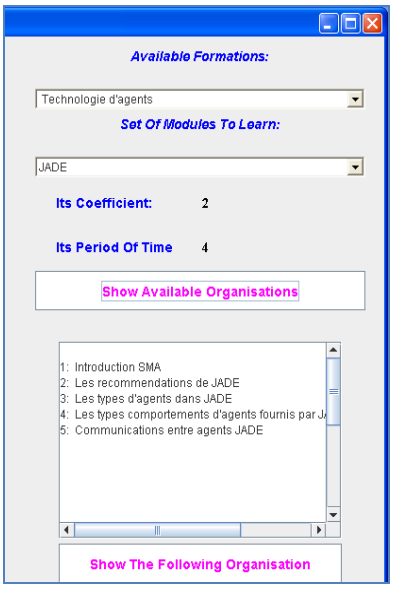
6.1 Introduction

Nous présentons dans ce chapitre, une vue de notre application au travers des interfaces utilisateurs qui ont été réalisées.

6.2 Les interfaces d'un visiteur de l'application (Celles de l'Agent Hôte)

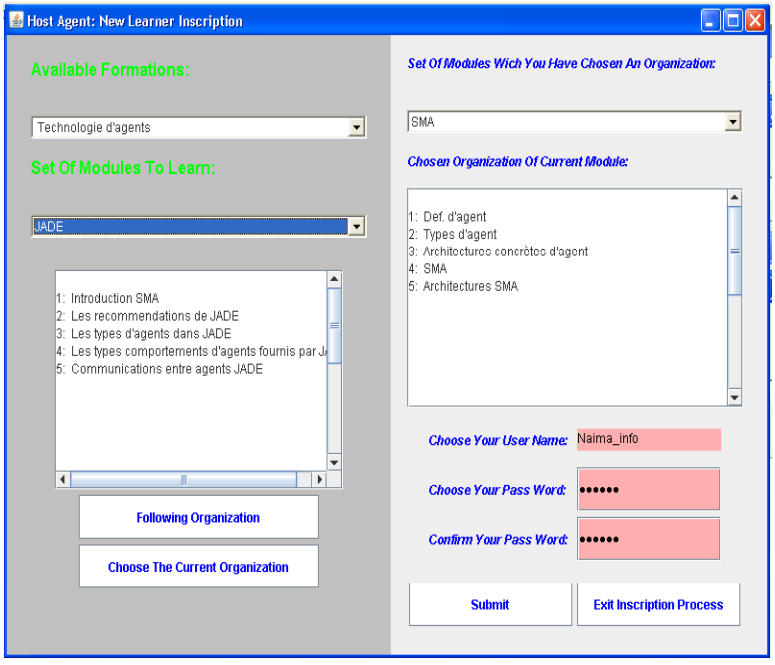
Lorsqu'un utilisateur se connecte, la fenêtre ci-dessous s'affiche. Cette interface permet de choisir le profil utilisateur.

(1) Le bouton : " ***To Recognize What The System Provides."*** est pour objectif d'expliquer les différents constituants du système éducatif (formations, modules, …). La figure suivante présente un exemple d'exécution de ce cas.



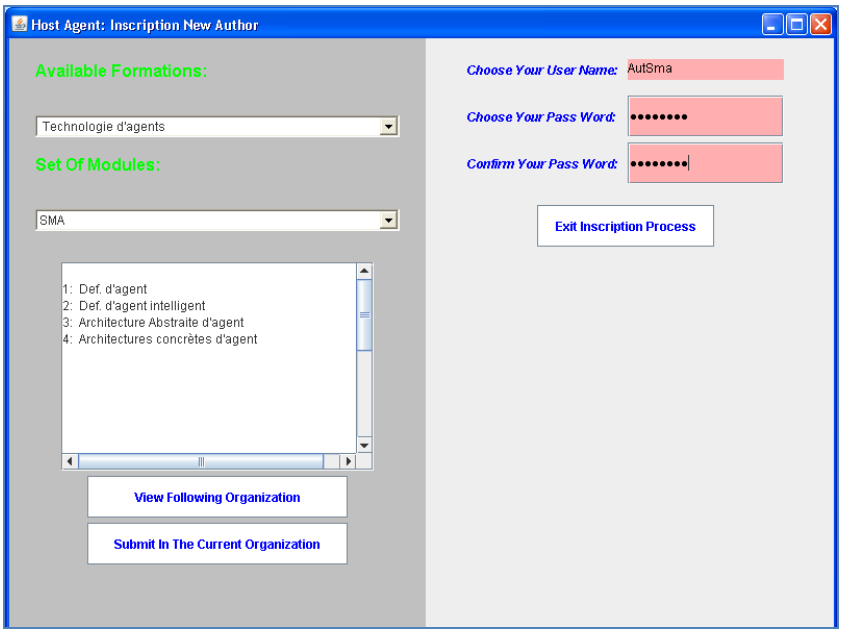
Les trois boutons qui se figurent après ((2) et (3) et (4)) sert à inscrire les nouveaux utilisateurs. Dans le cas d'un nouveau auteur et un nouveau tuteur l'*Agent Hôte* lance l'agent correspondant au choix de l'utilisateur (*Agent Auteur* et *Agent Organisateur* respectivement).

Pour un nouveau utilisateur de type apprenant (le bouton (2)), la fenêtre suivante s'affiche.

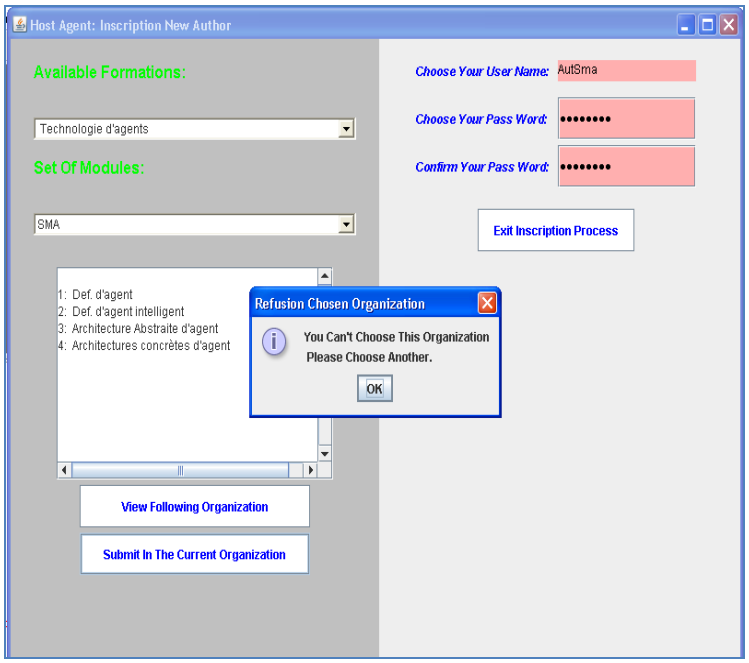


Cette interface permet à l'apprenant de choisir pour chaque module l'organisation qu'il veut suivre dans le cours. Il doit aussi introduire son nom d'utilisateur et le mot de passe.

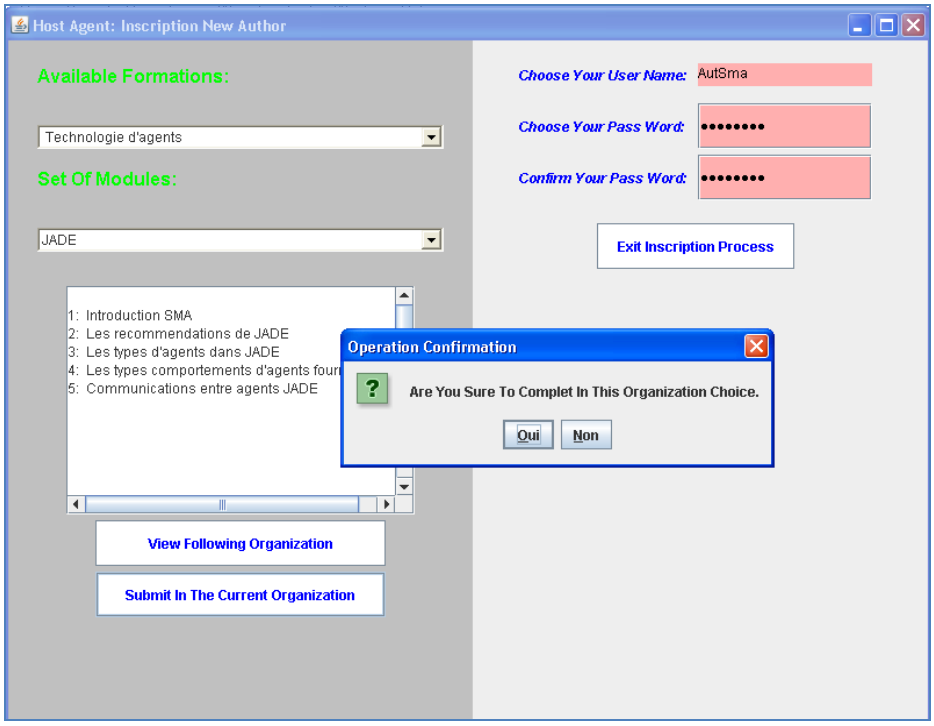
Lorsque le nouveau utilisateur est un auteur (le bouton (4)), l'interface correspondante est la suivante.



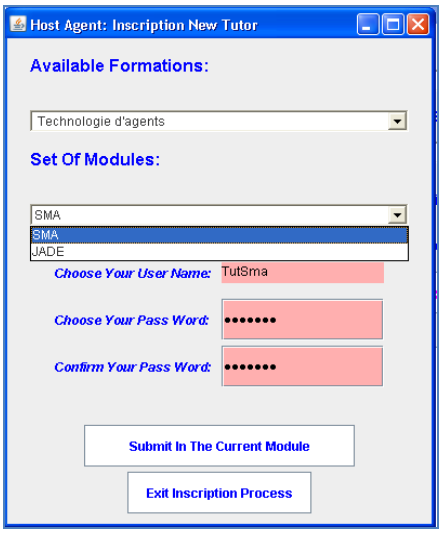
Cette interface permet au nouveau auteur de choisir l'organisation qu'il veut suivre pour rédiger les cours du module voulu. Si l'organisation choisie atteint le nombre maximum des auteurs autorisés, l'*Agent Hôte* répond par une impossibilité de continuer l'inscription comme il est montré dans l'interface suivante.



Dans le cas contraire, l'*Agent Hôte* prend la confirmation de l'auteur pour qu'il suit l'organisation choisie (voir la figure suivante) et il crée l'*Agent Auteur* pour assister et contrôler l'auteur pendant le processus de proposition d'une rédaction de cours (voir les interfaces de l'*Agent Auteur*).

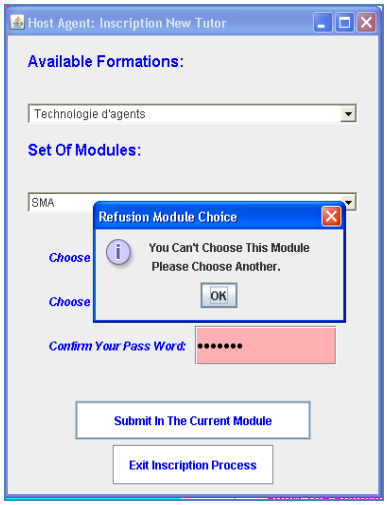


Concernant le cas d'un nouveau tuteur (bouton (3)), la fenêtre ci-dessous s'affiche.

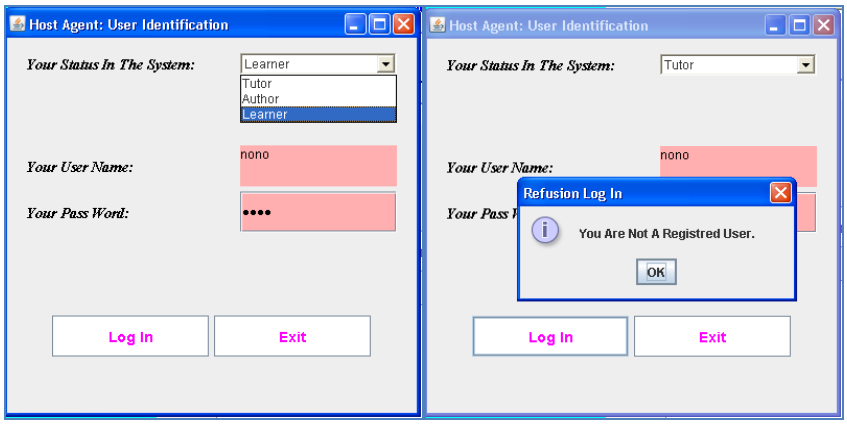


Donc, le nouveau tuteur peut choisir le module dont lequel il veut proposer une organisation. Comme il doit choisir un nom d'utilisateur et un mot de passe.

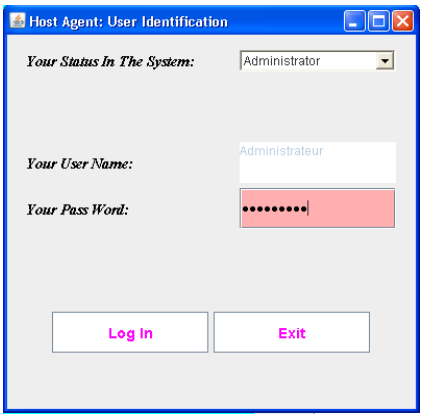
Si le nombre des tuteurs autorisés du module choisi atteint le nombre maximum, l'*Agent Hôte* informe le nouveau tuteur comme il est figuré dans la fenêtre ci-dessous. Sinon, un *Agent Organisateur* sera crée.



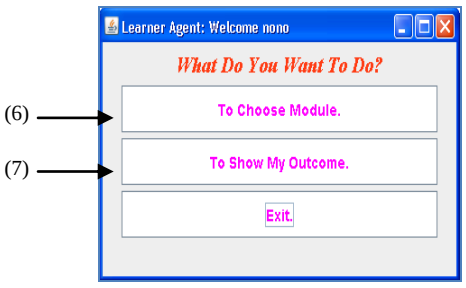
le bouton (5) est utilisé pour ouvrir une session ou un compte déjà existe. L'apprenant ou l'enseignant (tuteur ou auteur) saisit le nom d'utilisateur et le mot de passe qui lui ont été attribués lors de la création de son compte. Si l'identification est correcte, l'interface de l'agent(s) correspond (s) s'affiche (voir les interfaces des autres types agents). Dans le cas contraire, un message indique à l'utilisateur que le nom ou le mot de passe est incorrect.



Le choix du profil administrateur fait apparaître la fenêtre ci-dessous où seul le mot de passe doit être saisi. Ce choix est réservé à l'administrateur du système.

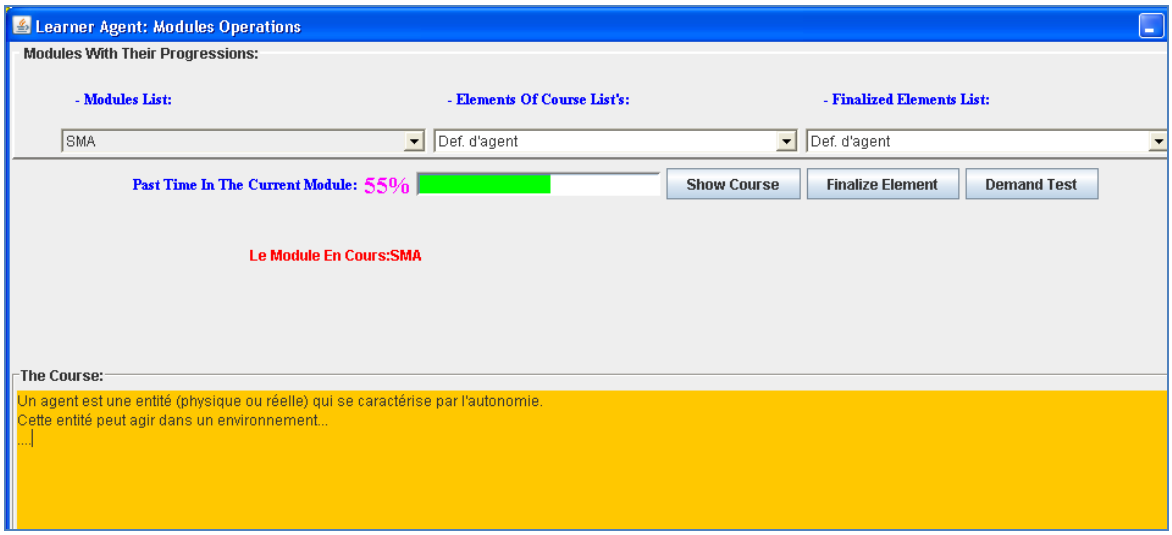


6.3 Les interfaces d'un enregistré apprenant (Celles de l'agent Apprenant)



Lorsqu'un apprenant se connecte, la fenêtre côtoyée s'affiche. Il peut choisir de faire ses différentes activités sur les modules (le bouton (6)) , ou de demander un bilan de ses divers progressions et résultats (le boutons (7)).

Le premier choix fait apparaître la fenêtre ci-dessous.

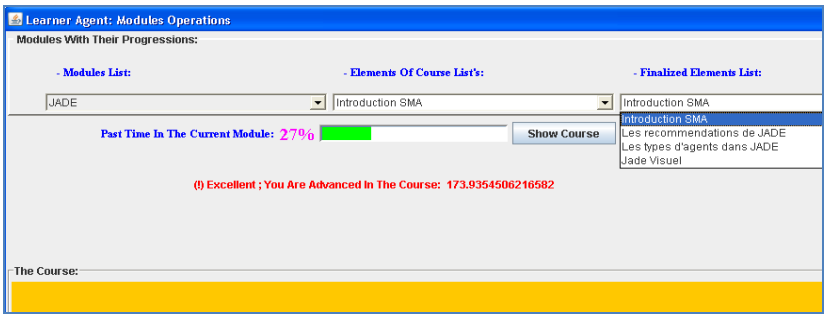


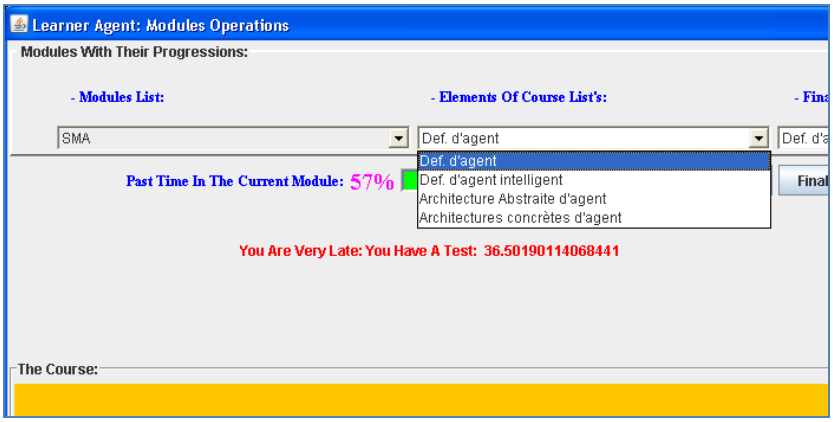
Elle se compose d'une partie supérieure comportant trois listes. La première contient les différents modules constituant la formation suivie par l'apprenant, la deuxième présente l'ensemble des éléments constituant le cours du module sélectionné et la dernière liste est constituée des éléments de cours que l'apprenant a terminé d'étudier. La partie inférieure est pour le dialogue entre l'apprenant et son *Agent Apprenant*.

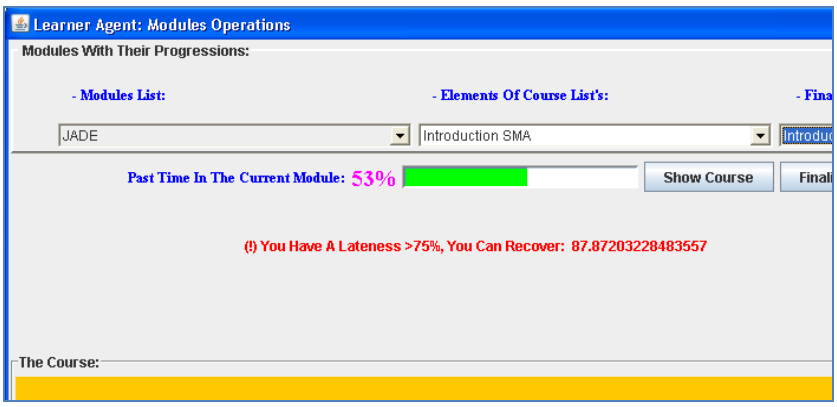
Le bouton « **Show Course** » permet d'afficher le cours (la rédaction) correspond à l'élément sélectionné dans la deuxième liste. Le bouton « **Finalize Element** » est utilisé par l'apprenant dans le cas où il veut informe l'*Agent Apprenant* qu'il termine d'étudier cet élément. L'élément finalisé sera ajouté à la troisième liste. L'apprenant peut demander de passer un examen (test) pour évaluer ses puissances et sa compréhension de cours. Le bouton « **Demand Test**» est pour cet objectif.

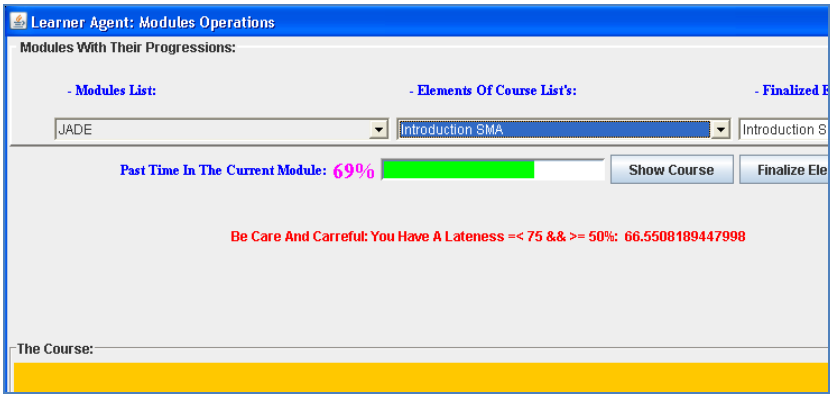
L'*Agent Apprenant* affiche périodiquement des remarques sur la progression de l'apprenant. Si l'apprenant est très retard, l'*Agent Apprenant* lui donne un court test (examen). En plus de ça, l'*Agent Apprenant* vérifie cycliquement (à tout moment où l'apprenant est en cours d'apprendre un module) le temps passé dans le module sélectionné et lui informe qu'il doit passer un examen lorsque l'un des temps des examens proposés par le tuteur est arrivé.

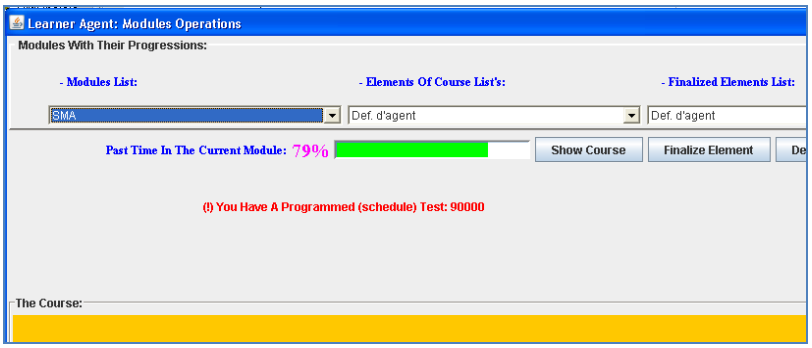
Ces points et d'autres sont révélées dans les figures ci-dessous

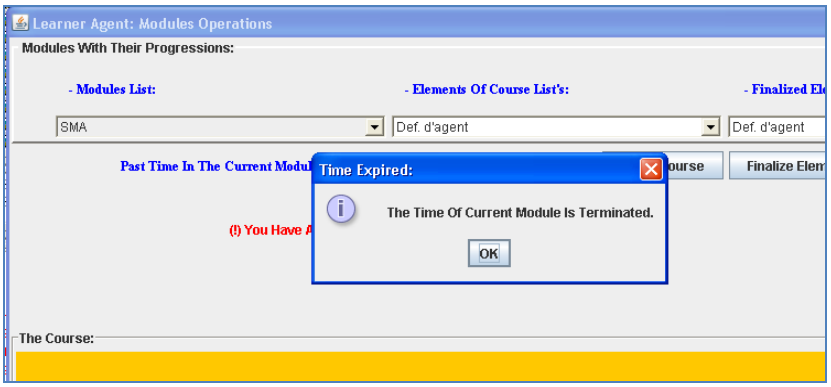

Learner Agent: Modules Operations
Modules With Their Progressions:
- Modules List:
- Elements Of Course List's:
- Finalized Elements List:
JADE
Introduction SMA
Introduction SMA
Introduction SMA
Les recommendations de JADE
Les types d'agents dans JADE
Jade Visuel
Past Time In The Current Module: 27%
Show Course
(!) Excellent ; You Are Advanced In The Course: 173.9354506216582
The Course:


Learner Agent: Modules Operations
Modules With Their Progressions:
- Modules List:
- Elements Of Course List's:
SMA
Def. d'agent
Def. d'agent
Def. d'agent intelligent
Architecture Abstraite d'agent
Architectures concrètes d'agent
Past Time In The Current Module: 57%
You Are Very Late: You Have A Test: 36.50190114068441
The Course:

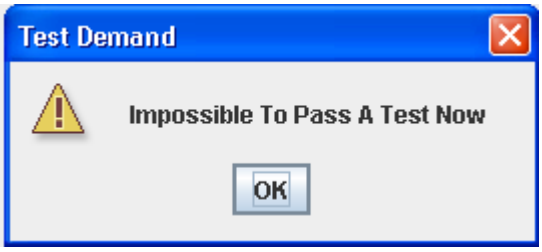

Learner Agent: Modules Operations
Modules With Their Progressions:
- Modules List:
- Elements Of Course List's:
JADE
Introduction SMA
Past Time In The Current Module: 53%
Show Course
(!) You Have A Lateness >75%, You Can Recover: 87.87203228483557
The Course:

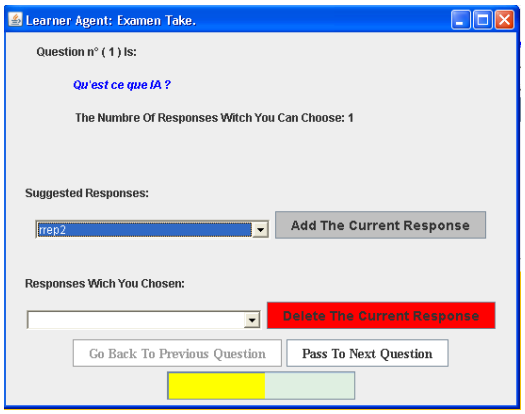

Learner Agent: Modules Operations
Modules With Their Progressions:
- Modules List:
- Elements Of Course List's:
JADE
Introduction SMA
Past Time In The Current Module: 69%
Show Course
Be Care And Carreful: You Have A Lateness =< 75 && >= 50%: 66.5508189447998
The Course:

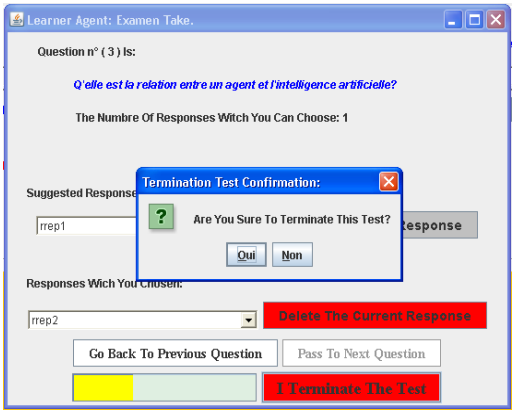


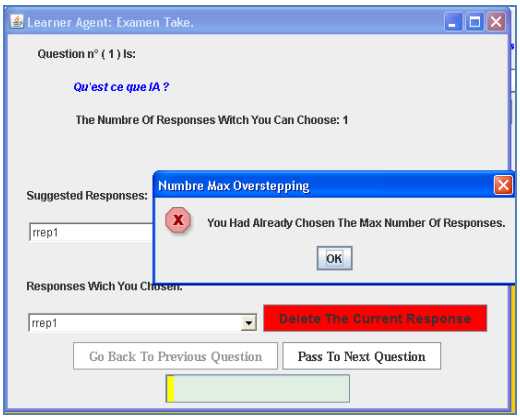


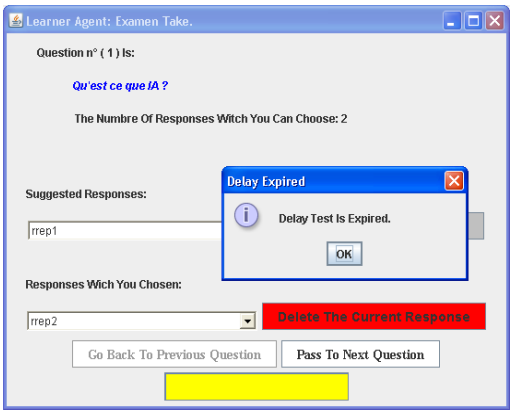
Nous avons distingué entre trois causes pour que l'apprenant passer un test : (1) l'apprenant demande d'une manière volitive de passer un test, (2) l'apprenant est très retard dans l'avancement de cours, (3) un temps d'examen proposé par le tuteur se déclenche. Dans n'importe de ces cas, l'*Agent Apprenant* demande de l'*Agent Testeur* de préparer un test à l'apprenant. Les fenêtres ci-dessous présentent les interfaces possibles concernant les tests (examens).

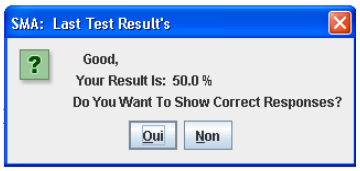


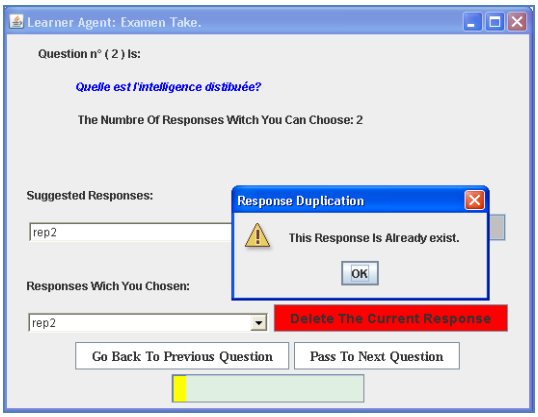

Learner Agent: Examen Take.
Question n° (1) Is:
Qu'est ce que IA ?
The Numbre Of Responses Witch You Can Choose: 1
Suggested Responses:
rrep2
Add The Current Response
Responses Wich You Chosen:
Delete The Current Response
Go Back To Previous Question
Pass To Next Question

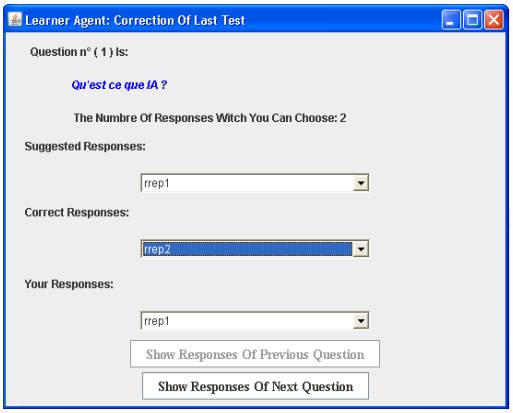

Learner Agent: Examen Take.
Question n° (3) Is:
Q'elle est la relation entre un agent et l'intelligence artificielle?
The Numbre Of Responses Witch You Can Choose: 1
Termination Test Confirmation:
Are You Sure To Terminate This Test?
Qui
Non
rrep1
rrep2
Delete The Current Response
Go Back To Previous Question
Pass To Next Question
I Terminate The Test

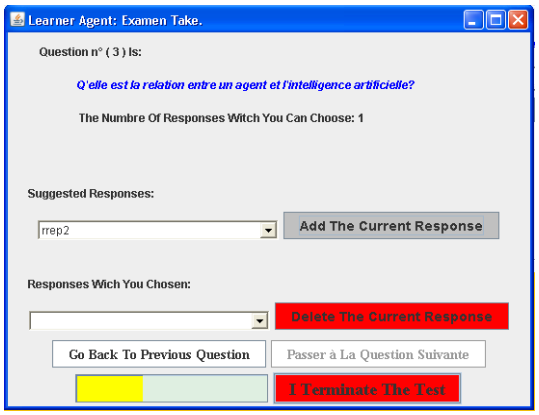

Learner Agent: Examen Take.
Question n° (1) Is:
Qu'est ce que IA ?
The Numbre Of Responses Witch You Can Choose: 1
Numbre Max Overstepping
You Had Already Chosen The Max Number Of Responses.
OK
Suggested Responses:
rrep1
Responses Wich You Chosen:
Delete The Current Response
Go Back To Previous Question
Pass To Next Question


Learner Agent: Examen Take.
Question n° (1) Is:
Qu'est ce que IA ?
The Numbre Of Responses Witch You Can Choose: 2
Delay Expired
Delay Test Is Expired.
OK
Suggested Responses:
rrep1
Responses Wich You Chosen:
rrep2
Delete The Current Response
Go Back To Previous Question
Pass To Next Question

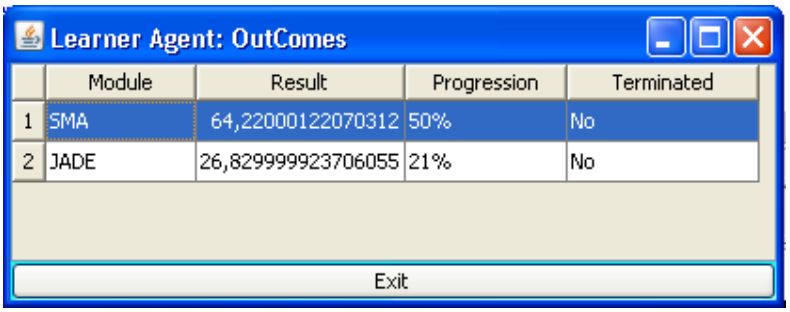

SMA: Last Test Result's
Good,
Your Result Is: 50.0 %
Do You Want To Show Correct Responses?
Qui
Non


Learner Agent: Examen Take.
Question n° (2) Is:
Quelle est l'intelligence distibuée?
The Numbre Of Responses Witch You Can Choose: 2
Response Duplication
This Response Is Already exist.
OK
Suggested Responses:
rep2
Responses Wich You Chosen:
Delete The Current Response
Go Back To Previous Question
Pass To Next Question


Learner Agent: Correction Of Last Test
Question n° (1) Is:
Qu'est ce que IA ?
The Numbre Of Responses Witch You Can Choose: 2
Suggested Responses:
rrep1
Correct Responses:
rrep2
Your Responses:
rrep1
Show Responses Of Previous Question
Show Responses Of Next Question

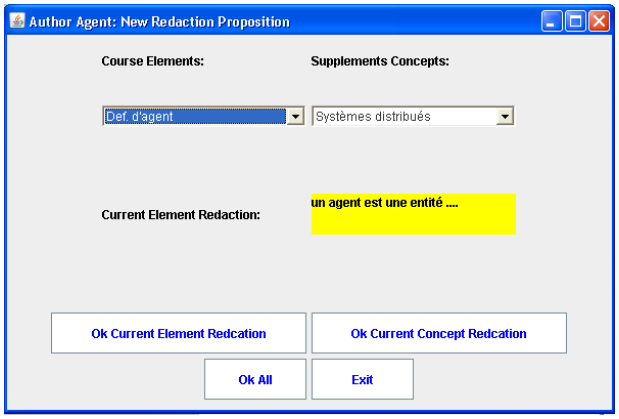

Learner Agent: Examen Take.
Question n° (3) Is:
Q'elle est la relation entre un agent et l'intelligence artificielle?
The Numbre Of Responses Witch You Can Choose: 1
Suggested Responses:
rrep2
Add The Current Response
Responses Wich You Chosen:
Delete The Current Response
Go Back To Previous Question
Passer à La Question Suivante
I Terminate The Test

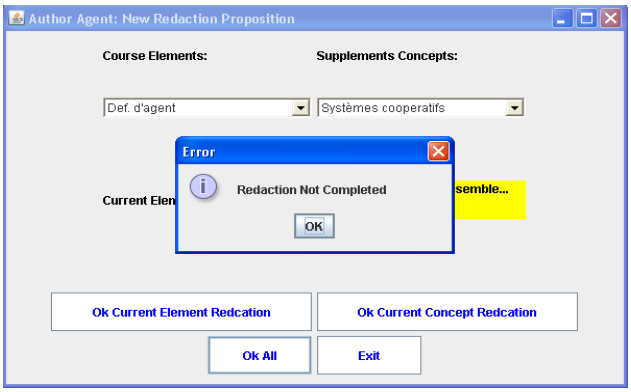
Le bouton (7) (To Show My Outcome) donne un bilan résume la progression de l'apprenant et ses résultats comme il est indiqué dans la fenêtre en face.

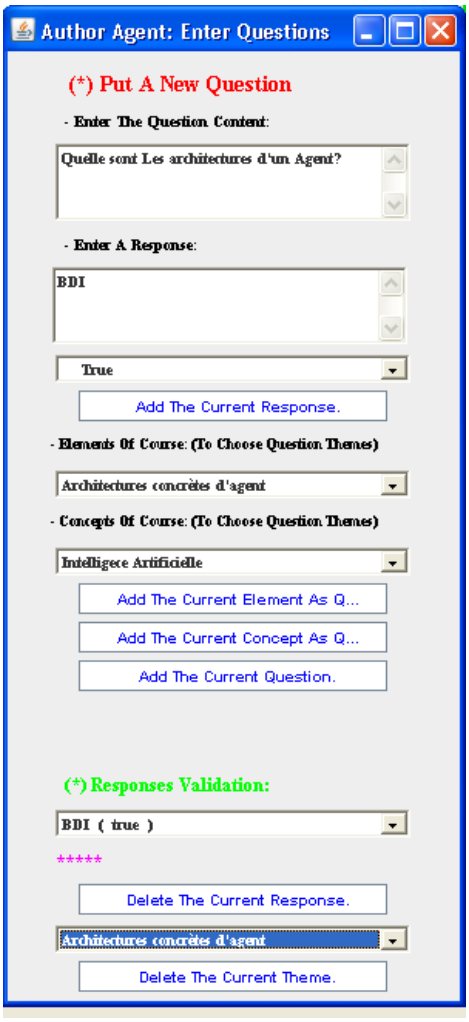


6.4 Les interfaces d'un auteur (Celles de l'*Agent Auteur*)

La première interface est celle qui s'affiche dans le cas d'un nouveau auteur où l'*Agent Auteur* est demandé de suivre l'auteur pendant la rédaction de cours. C'est ici par la suite que pourra par exemple s'affiche les fenêtres suivantes.

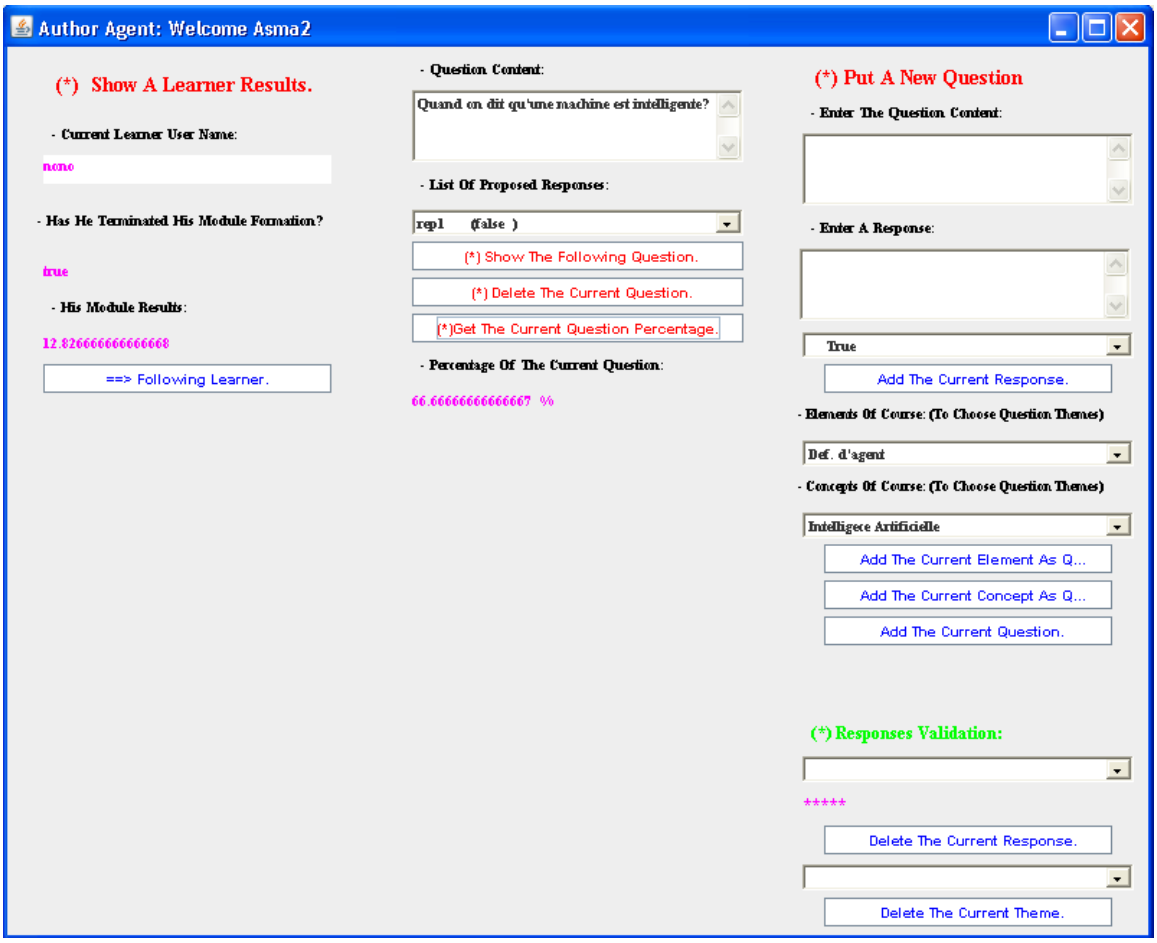






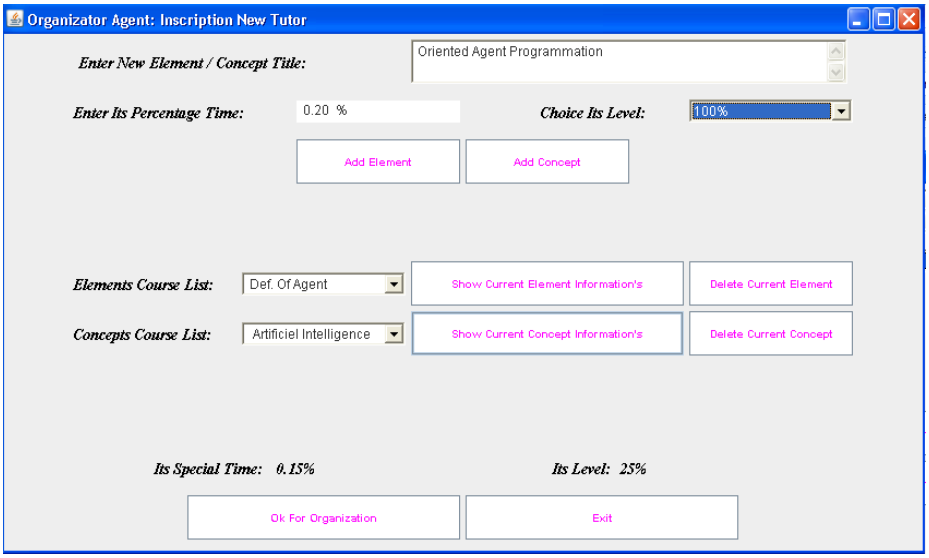
L'auteur doit poser au moins un nombre prédéfini des questions pour chaque élément et concept composant le cours. Cette condition est obligatoire pour que l'auteur sera accepté.

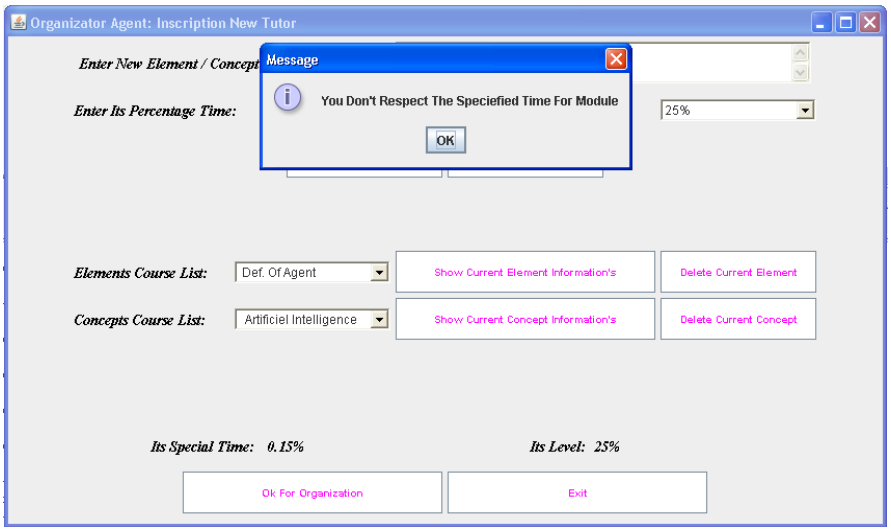
Pour un auteur ayant un compte, l'interface ci-dessous s'affiche. Elle lui permet de voir les différents résultats obtenus par ses apprenants dans son module, gérer ses questions (Ajouter/Supprimer/Prendre Pourcentage des réponses justes…), …



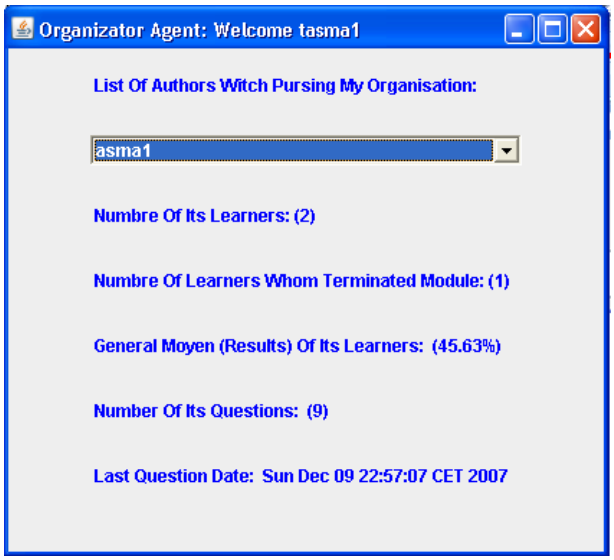
6.5 Les interfaces d'un tuteur (Celles de l'*Agent Organisateur*)

Les interfaces suivantes s'affichent dans le cas d'un nouveau tuteur. ce dernier est obligé de respecter la durée autorisée par l'administrateur.



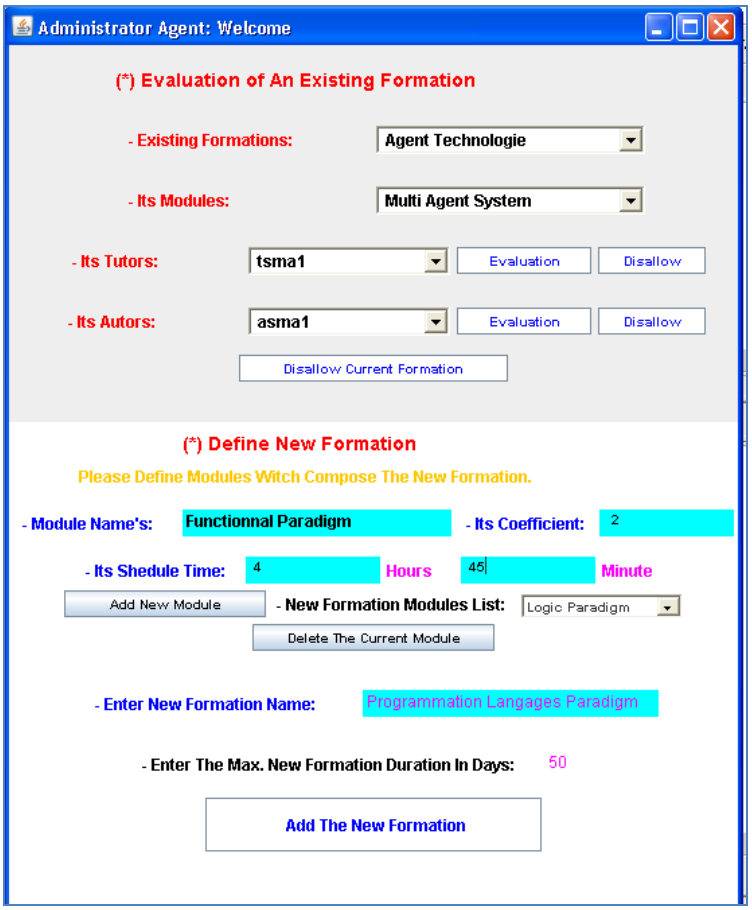


La fenêtre ci-dessous offre les informations des différents auteurs suivant l'organisation de tuteur connecté (cas d'un tuteur enregistré).



6.6 Les interfaces d'un tuteur (Celles de l'*Agent Organisateur*)

Quand l'utilisateur connecté est l'administrateur du système, la fenêtre suivante s'affiche. Elle résume tous ses activités possibles sur le système éducatif.



6.7 Conclusion

Dans ce chapitre nous avons exposé et discuté les interfaces possibles corresponds aux différents types d'utilisateurs et dépendamment à leurs comportements et besoins.

Chacune de ces interfaces ou fenêtres est liée (indexée) à son type d'agent car elle est considérée comme un résultat d'un comportement de cet agent selon des connaissances locales ou connaissances sur l'environnement (événements d'interface utilisateur, interaction avec les autre agents, …).

Chapitre 7

Conclusion générale

En conclusion, nous tenons à mentionner également que la façon d'apprendre change avec le e-Learning, mais aussi la manière de concevoir les produits de formation et de les diffuser. D'autres intervenants tels que les graphistes, les intégrateurs multimédia et les informaticiens sont maintenant présents dans le processus de conception de la formation. Le concepteur pédagogique doit désormais interagir avec un plus grand nombre d'intervenants qu'auparavant et il doit aussi élargir son champ d'expertise. Quant aux formateurs, ils doivent s'adapter à des rôles moins traditionnels qu'avant.

Ce travail utilise les technologies de l'information et de la communication et de l'intelligence artificielle pour le traitement des activités pédagogiques. De plus, nous soulignons qu'on doit se préoccuper de la conception des documents et la manière de les exposer et les utiliser avant leur présentation aux apprenants. Ainsi nous proposons une granularité assez fine du document afin qu'il puisse constituer une source effective de connaissances pour les diverses activités pédagogiques proposées aux apprenants (Consultation de cours, Obtention d'un examen,…). Par ailleurs, pour que l'adaptabilité soit gérée facilement, nous avons incorporé dans le document toutes les descriptions susceptibles de produire dynamiquement différentes présentations d'une même information renforçant ainsi son intérêt d'apprentissage.

Plus précisément, le but de ce travail était la conception d'un système d'enseignement sur le Web en utilisant la technologie d'agents comme un moyen d'adaptation de system éducatif selon les comportements de ses utilisateurs. Donc La plate-forme d'enseignement et d'apprentissage à distance est vu comme une communauté éducative composée d'agents qui coopèrent et travaillent ensemble pour l'éducation des apprenants.

Notons que cette approche développe un environnement d'apprentissage, dans une démarche mettant les besoins et les difficultés de l'apprenant au centre du processus de conception et de modélisation informatique. Nous avons proposé une architecture multi

agent capable d'initier l'apprentissage et de gérer particulièrement un enseignement et un suivi individualisé. En d'autres termes, nous avons présenté, les liens directs entre les technologies d'agents et celles du e-Learning.

Dans notre mémoire et après l'introduction générale, le deuxième chapitre a été consacré à présenter la technologie d'agent et les systèmes Multi-Agents. Le troisième chapitre s'accentue sur le e-Learning. La modélisation de notre système était le sujet du quatrième chapitre. Les grandes lignes de la validation de notre système était décrites dans le cinquième chapitre. Ensuite, nous avons présenté quelques cas de tests et résultats.

Les tests d'usage de la plate-forme ont montré son adéquation à la spécification initiale des comportements individuels et coopératifs des agents. Cependant il existe une difficulté concernant la communication entre les agents dont elle demande la prise en compte d'une grande variété de formes et de niveaux de représentation des connaissances directement liée à la multiplicité de leurs rôles. Cette gestion de la communication à niveau "connaissance" est un point de passage obligé pour aller vers le système dont nous cherchons à préciser les principes de conception et dont la fonctionnalité globale "éducante" sera le résultat des interactions et de la coopération d'une société d'agents spécialisés selon des types de compétences didactiques ou pédagogiques clairement identifiés.

Dans le cadre de notre application, l'utilisation des protocoles d'interaction et des actes de langage pour formaliser les interactions agent-agent ou agent-utilisateur contribue fortement à l'homogénéité globale du système. Un module d'interaction intégré à chaque agent prend en charge le suivi des messages et permet aisément de modifier les rôles et les comportements des agents. Cet aspect est primordial pour faire adapter simplement et rapidement le système.

Comme bilan de ce travail, on peut résumer ses caractéristiques en :

- Les S.M.A représentent une bénéfice pour le e-Learning à travers le web puisqu'ils sont adéquats dans les cas des environnements ouverts,
- Les différentes caractéristiques d'agents (autonomie, coopération, …) aident à l'adaptabilité du système éducatif aux divers comportements de ses acteurs (apprenants, tuteurs, auteurs, administrateur,…)
- Faire le système éducatif centré apprenant tend vers une formation constructive où l'apprenant a un degré de liberté pour choisir les temps d'examens, les éléments de

cours souhaités d'être des sujets d'un examen, réviser les cours des différents modules d'une manière libre et avec la séquence et l'ordre voulus, …. De l'autre coté, l'apprenant s'est posé toujours sous le contrôle du système, …

- Le système rend services aux autres acteurs du système éducatif (auteurs, tuteurs, …). Cela permet à chacun d'eux de réaliser ses différents rôles.
- JADE est un bon outil simplifiant le développement des systèmes multi-agents tout en fournissant un ensemble complet de services et d'agents conformes aux spécifications FIPA.

L'approche de conception itérative et incrémentale fait que notre travail ne s'arrête pas là et que des perspectives d'évolution s'offrent à nous. Nous cherchons à évaluer les impacts de l'utilisation de notre plateforme.

Nous envisageons d'enrichir et d'étendre la plateforme par d'autres fonctionnalités telles que l'intégration des formations ayants des projets de fin d'étude où l'apprenant aura un projet dans un domaine spécifique et il sera sous la supervision d'un encadreur à distance. Ainsi, nous voulons profiter des bénéfices du web sémantique dans la construction des examens présentés à l'apprenant et de la recherche des différents éléments de cours, …

Nous pensons également ajouter certaines fonctionnalités aux interfaces des apprenants en proposant à ceux-ci une vue sur leurs propres activités et également sur des activités collectives.

En addition, nous voulons de limiter le nombre des tuteurs et des auteurs dans chaque module à un nombre maximum. Cela nous renforce de traiter le problème de l'exclusion mutuelle dans le cas des demandes d'inscription simultanées.

Références

[1] [En ligne] www.jadeTilab.com.

[2] [En ligne] www.Fipa.org.

[3] **Alex, De Koning.** *Les Agents Intelligents.* s.l. : Publication technique de la SmalS-vM.

[4] **Aloys, Mbala, Reffay, Christophe et Chanier, Thierry. 2003.** *"SIGFAD: un système multi-agent pour soutenir les utilisateurs en formation à distance" Environnements Informatiques pour l'Apprentissage Humain.* Strasbourg : s.n., 2003.

[5] **2006.** Apprentissage en ligne. *http:// fr.wikipedia.org.* [En ligne] Novembre 2006.

[6] **Aubert, Sterenn. 2005.** *Le e-Learning adaptatif.* Master Recherche PMLT, Université de Nice Sophia-Antipolis : s.n., 2005.

[7] **BenAyache, Ahcene. 2005.** *Construction d'une mémoire organisationnelle de formation et evaluation dans un contexte eLearning: Le projet MEMORAE.* 2005. Vol. Thèse de docteur de l'UTC.

[8] **Bergia, Loris. 2001.** *Conception et réalisation d'une plate-forme multi-agents pour l'apprentissage et l'enseignant à distance.* Laboratoire Leibniz-IMAG, France : s.n., 2001.

[9] **Boissier, Olivier. 2004.** *Systèmes Multi-Agents.* s.l. : Master Web Intelligence, 2004.

[10] **Bourdon, François et al.** *Systemes multi-agents* : Cours exposé dans le cadre du DEA-IAA-(TRONC COMMUN), Université de CAEN.

[11] **Chaib-draa, B. 2001.** *Agents et Systèmes Multiagents: Thèmes, Approches et défis*: Cours ASMA, Département D'informatique, Université Laval Ste-Foy, 2001.

[12] **Chaib-draa, B, Jarras, I et Moulin, B. 2001.** Systèmes multiagents: Principes généraux et applications. [auteur du livre] J.P Briot et Y Demazeau. *Agent et systèmes multiagents* . Canada : s.n., 2001.

[13] **Chaib-draa, Brahim. 1999.** *Agents et systèmes multiagents*: Notes de cours, Département d'informatique, Faculté des sciences et de genie,Université Laval Quebec, 1999.

[14] **CHANG, Hervé et COLLET, Philippe. 2004.** *Négociations de contrats: des systèmes multiagents aux composants logiciels.* Vol. Journée Multi-Agents et Composants: JMAC 2004, Ecole des Mines de Paris : s.n., 2004..

[15] **Demazeau, Yves. 2000.** *Multi-Agent Systems Methodology.* Tandil : ASAI ' 2000, 2000.

[16] **Ferber, Jacques.** *Technologie multi-agent.*

[17] **Finin, Tim, Labrou, Yannis et Mayfield, James.** *KQML as an agent communication language.* Computer Science Departement, University of Maryland Baltimore Country, USA : s.n.

[18] Journal du net. [En ligne] www.journaldunet.com.

[19] **KHALFAOUI, S., CHAARI, W. L. et PINNA DERY, A. M. 2004.** *Interactions entre composants et environnements multi-agents.*Vol. Journée Multi-Agents et Composants: JMAC 2004. Ecole des Mines de Paris : s.n., 2004.

[20] **Kirnm, Stefan. 1999.** *Multi-Agent Organizations: Coordinating Individual and Global Goals.* TU Ilmenau : s.n., 1999.

[21] **2004.** La e-formation: un marché promoteur. *www.minefi.gouv.fr.* [En ligne] 13 05 2004.

[22] **Laurent, Vercouter. 2004.** *MAST : Un modèle de composant pour la conception de SMA.* Journée Multi-Agents et Composants: JMAC 2004, Ecole de Mines de Paris, 2004.

[23] **LEMLOUMA, Tayeb et BOUDINA, Abdelmadjid.** *L'intelligence artificielle distribuée et les systèmes multiagents.*

[24] **Müller, Jean Pierre. 2002.** *Des systèmes autonomes aux systèmes multi-agents: Interaction, émergence et systèmes complexes.* Université Montpellier II : s.n., 2002. Vol. Ecole doctorale (Information, Structures, Systèmes).

[25] **Pasquier, Philippe. 2001.** *Coomunication entre agents.* Université Laval, Canada : s.n., 2001.

[26] **Pesty, S, Webber, C et Balacheff, N. 2003.** *Baghera: une architecture multi-agents pour l'apprentissage humain.* Toulouse : Cognitique, (P. Aniorte, S. Gouarderes eds), 2003. Vol. Cepadeus Edition.

[27] **Richard, Nadine. 2002.** *Agents autonomes et systèmes multiagents.* s.l. : ENST, 2002.

[28] **Sayasenh, V.** Merise classique: L'essentiel. [En ligne] www.compucycles.com.

[29] **Schobens, P Y, Massonet, P et Ponsard, C. 2003.** *Les agents intélligents au service de la gestion électronique de l'entreprise ?* 2003.

[30] **VAN AEKEN, Francis. 1999.** *Les systèmes multi-agents minimaux.* 1999. Vol. thèse de doctorat.

[31] **Vaucher, Jean et Ncho, Ambroise. 2003.** JADE Tutoriel and Primer. [En ligne] Dep. d'informatique, Université de Montréal , Septembre 2003. http://jade.cslt.it.