Résumé

Dans ce mémoire nous proposons une combinaison entre deux disciplines du domaine informatique, celle du workflow interorganisationnel lâche qui a pour but de faire collaborer des processus workflow répartis, autonomes, hétérogènes, issues et s'exécutant dans différentes organisations, et la technologie des agents qui connait une projection sur divers domaines comme la recherche d'information, la simulation, les systèmes d'aide à la décision,…etc. en raison de leurs adaptation aux différents contextes.

La flexibilité, l'adaptation et la distribution sont vues comme les majeurs défis pour le workflow interorganisationnel. Dans ce travail nous proposons une architecture pour le workflow interorganisationnel lâche à base des agents.

De ce qui précède, on peut dire que l'exécution entière du modèle implémenté est divisée en deux parties : l'intra-exécution, qui exprime l'exécution au sein d'une organisation, et l'inter-exécution, qui représente l'interaction entre les organisations. Un agent joue le rôle du moteur de workflow.il assure la supervision de l'exécution de processus workflow au sein de l'organisation et l'interaction avec l'environnement externe de l'organisation.

Mots clés : *workflow, agents, interaction, ontologie.*

Abstract

In this work we propose a combination between two disciplines of the data-processing field, the loosely coupled interorganisationnel workflow aiming to make collaborating processes workflow distributed, autonomous and heterogeneous resulting and being executed in various organizations and the agents which are autonomous, co-operative and adaptive entities to various contexts.

Flexibility, adaptation and distribution are the major challenges for the interorganizational workflow.

In this work we propose an agent based architecture for the loosely coupled interorganizational workflow.

By what precedes, we can say that the whole execution of the implemented model is divided into two parts: the intra-execution, which expresses the execution within an organization, and the inter-execution, which represents the interaction between organizations. An agent plays the role of a workflow engine; it ensures the supervision of the execution of the process workflow in the organization and the interaction with its external environment.

Key words: *workflow, agents, discovery, negotiation, contract.*

Sommaire

Chapitre II :Les workflow interorganisationnels; Définitions, modèles et formes d'interopérabilité

Chapitre IV : Etude de cas et perspectives d'implémentation

Conclusion générale

Table Des Figures

Chapitre I :

Chapitre II :

Chapitre III :

Chapitre IV :

Introduction

Introduction

1. Contexte du travail

A l'heure actuelle, les Workflows (WFs) émergent comme une nouvelle technologie dans la gestion des opérations d'affaires des organisations.

Un WF est une spécification des opérations de réalisation qui existent dans une organisation. Cette spécification doit tenir en compte plusieurs aspects, tels que la chronologie des opérations, la complexité et la nature de ces opérations, la disponibilité et la localisation des ressources nécessaires à ces opérations, la nature des participants à ces opérations, etc.

L'évolution économique et industrielle et la propagation des nouvelles technologies de l'information et de la communication ont permis l'apparition plusieurs modes d'interopérabilité dans le domaine des workflow interorganisationnels tels que : partage de capacité, exécution chaînée, sous-traitance, transfert de cas, transfert de cas étendu, public-à-privé et faiblement couplé (lâche).

2. Problèmatique est objectifs

Notre contribution est orientée vers le workflow interorganisationnel lâche (WIOL). Cette forme d'intropérabilité est caractérisée par la présence de *n* partenaires, chacun d'eux dispose d'un workflow local privé et une structure d'interaction qui lui permet d'interagir avec les autres partenaires en utilisant des messages asynchrones.

L'objectif de ce travail est d'aboutir à un modèle à base d'agents pour les workflows interorganisationnels faiblement couplés pour :

- Permettre à une organisation ayant des services à fournir de les publier pour qu'ils soient perceptibles par son environnement , mettre à jour des services déjà publiés ou supprimer un service s'il n'est plus disponible au niveau de l'organisation
- Permettre à une organisation cherchant des services de découvrir les offres existantes et de négocier pour choisir le partenaire répondants aux critères qu'elle exige
- Trouver un mécanisme de négociation pour établir un contrat déterminant les droits et les obligations de chaque partenaire pour garantir l'exécution parfaite du service.

Le choix du paradigme agent pour l'implémentation de ce modèle est dû à sa capacité de fournir des réponses aux tendances actuelles de développement d'applications par intégration d'éléments logiciels distribués sur de nombreux systèmes. La principale

caractéristique qui distingue les agents est leur autonomie. Un agent a la capacité de décider spontanément de réaliser une activité afin de remplir des objectifs individuels. Les agents s'exécutent donc concurremment dans des threads séparés. Ils ne sont cependant pas isolés : ils interagissent en échangeant des messages grâce à des mécanismes de messagerie asynchrones. Ainsi, les agents peuvent collaborer tout en préservant leur autonomie.

Envoyer un message à un autre agent pour lui demander un service est une action non bloquante : l'agent client (l'expéditeur de la requête) n'attend pas la réponse de l'agent serveur et peut poursuivre son activité courante. Le client récupère la réponse plus tard, dans un message envoyé en retour par le serveur. Réciproquement, recevoir un message n'est pas un événement préemptif : l'agent récepteur peur choisir de reporter le traitement de ce message afin de terminer des activités plus prioritaires.

L'asynchronisme des communications donne aux agents les moyens de gérer des schémas d'exécution avancés comme requérir de manière redondante le même service à différents agents pour obtenir de meilleures garanties de performance et de fiabilité ce qui les rend l'outil logiciel le plus adéquat pour la modélisation des workflows interorganisationnels faiblement couplés.

3. Organisation du mémoire

Ce mémoire est structuré en deux grandes parties. La première partie est consacrée à un état de l'art de la technologie agent et du domaine du Workflow. Elle comporte deux chapitres.

Dans le premier chapitre nous allons parler du concept qui va constituer l'élément de base de notre architecture, il s'agit du paradigme agent.

Le deuxième chapitre sera consacré au domaine des workflow ; nous aborderons les différents concepts liés à ce domaine, les différents types de workflows existants, les aspects relatifs à leur modélisation ainsi que les formes d'interopérabilité dans les workflow interorganisationnels.

La deuxième partie est consacrée à la proposition d'une architecture à base d'agents pour les workflow interorganisationnels faiblement couplés. Elle comporte deux chapitres.

Le troisième chapitre qui représente une mise en œuvre des acquis théoriques des deux chapitres précédents, nous allons proposer un modèle à base d'agents pour les workflow interorganisationnels faiblement couplés permettant aux différentes organisations

d'interagir avec leur environnement à travers des publications ,des recherche et des négociations pour l'établissement des contrats d'exécution des services.

Le dernier chapitre sera concrétisation du modèle proposé par une projection sur un cas réel en utilisant les outils logiciels adéquats.

Nous clôturons ce mémoire par un bilan du travail effectué en donnant des conclusions sur les objectifs atteints, les difficultés rencontrées et les perspectives ouvertes pour le développement de ce travail.

Chapitre 1 : Agents et systèmes multi agents - présentation, définitions et concepts

1. Introduction

Les développements récents de l'informatique, notamment la puissance des machines et l'informatique distribuée ont donné une importance fondamentale au problème de la communication et de la coordination entre entités autonomes. La notion d'Agent et de systèmes Multi-Agents a fait son apparition non seulement dans des domaines théoriques comme la théorie des jeux et l'intelligence artificielle, mais aussi dans des domaines appliqués comme le commerce électronique ou les plateformes de simulations.

Ce chapitre présente les concepts principaux de la technologie d'agent. Nous n'avons pas l'intention d'effectuer une étude détaillée, mais la définition des concepts nécessaires pour la lisibilité de ce travail.

2. Définition de l'agent

De diverses définitions ont été données au concept de l'agent. Nous énumérerons dans ce qui suit quelque unes ;

- Ferber[1] définit un agent comme étant une entité physique ou virtuelle évoluant dans un environnement dont il n'a qu'une représentation partielle et sur lequel il peut agir. Il est capable de communiquer avec d'autres agents et est doté d'un comportement autonome.
- Yves Demazeau [Demazeau and Costa, 1996], un agent est une entité réelle ou virtuelle dont le comportement est autonome, évoluant dans un environnement qu'il est capable de percevoir et sur lequel il est capable d'agir, et d'interagir avec les autres agents.
- Mickael Wooldridge (WOOLDRIDGE, 2002), un agent est un système informatique capable d'agir de manière autonome et flexible dans un environnement pour atteindre leur objectif. Par flexibilité on entend par:

– Réactivité : un système réactif maintient un lien constant avec son environnement et répond aux changements qui y surviennent.
– Pro-activité : un système proactif génère et satisfait des buts. Son comportement n'est donc pas uniquement dirigé par des événements.
– Capacités sociales : un système social est capable d'interagir ou coopérer avec d'autres systèmes.

3. Les modèles d'agents

Les agents peuvent être classés selon leurs degrés d'autonomie, de coopération et d'adaptation ; caractéristiques généralement considérées comme principales.

- Quand l'agent a un but à atteindre, il doit être suffisamment autonome pour pouvoir prendre des initiatives permettant atteindre ce but.
- Pour que l'ensemble des agents constitue un système cohérent chacun d'entre eux doit avoir un certain degré de coopération.
- L'agent doit agir en fonction de son environnement ; c'est à dire qu'il doit s'adapter à celui-ci.

Le plus haut niveau d'autonomie permet à l'agent de planifier ses actions, le plus haut niveau de coopération accorde à l'agent des capacités de négociation, le plus haut degré d'adaptation permet à l'agent d'adapter et d'acquérir des connaissances.

3.1) Modèle d'agents réactifs

Ce sont des agents qui :

- fonctionnent selon un mode stimuli/réponse (basé sur la règle perception→ action). Dès qu'ils perçoivent une modification de leur environnement, ils répondent par une action programmée.
- L'agent réactif ne possède pas une représentation complète de son environnement et n'est pas capable de tenir compte de ses actions passées.
- De ce fait, il ne peut avoir des capacités d'apprentissage.

L'architecture interne de l'agent est la suivante:

- les capteurs : qui servent à détecter un évènement survenu dans l'environnement,
- les tâches : qui décrivent une partie du comportement de l'agent,
- les effecteurs : qui permettent de réaliser les tâches.

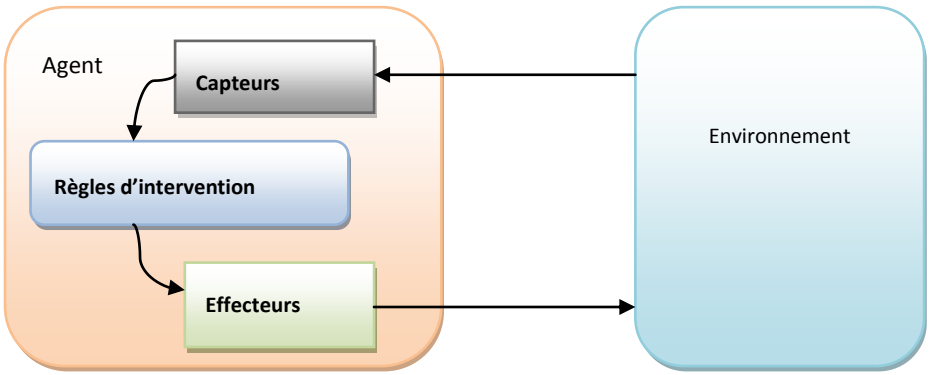

Figure I-1 : Architecture interne de l'agent réactif

3.2) Modèle d'agents cognitifs

Ils se basent sur un modèle qui provient d'une métaphore du modèle humain. Ces agents possèdent une représentation explicite de leur environnement, des autres agents et d'eux-mêmes. Ils sont aussi dotés de capacités de raisonnement, planification et de communication. Ce modèle est appelé aussi modèle BDI (Believe, Desire, Intention). Il est fondé sur trois attitudes qui définissent la rationalité d'un agent intelligent :

B = Believes = Croyances : Il s'agit de l'ensemble d'informations que possède l'agent sur son environnement et sur les autres agents agissant dans le même environnement. Ces informations constituent les connaissances supposées être vraies pour l'agent.

D = Desires = Désirs : Ils constituent les différents états de l'environnement et parfois de l'agent lui-même, on peut dire que ce sont les objectifs que se fixe un agent.

I = Intentions = Intentions: Ce sont les actions à faire par l'agent pour accomplir ses désirs. Ils se composent de l'ensemble des plans exécutés pour atteindre ses objectifs.

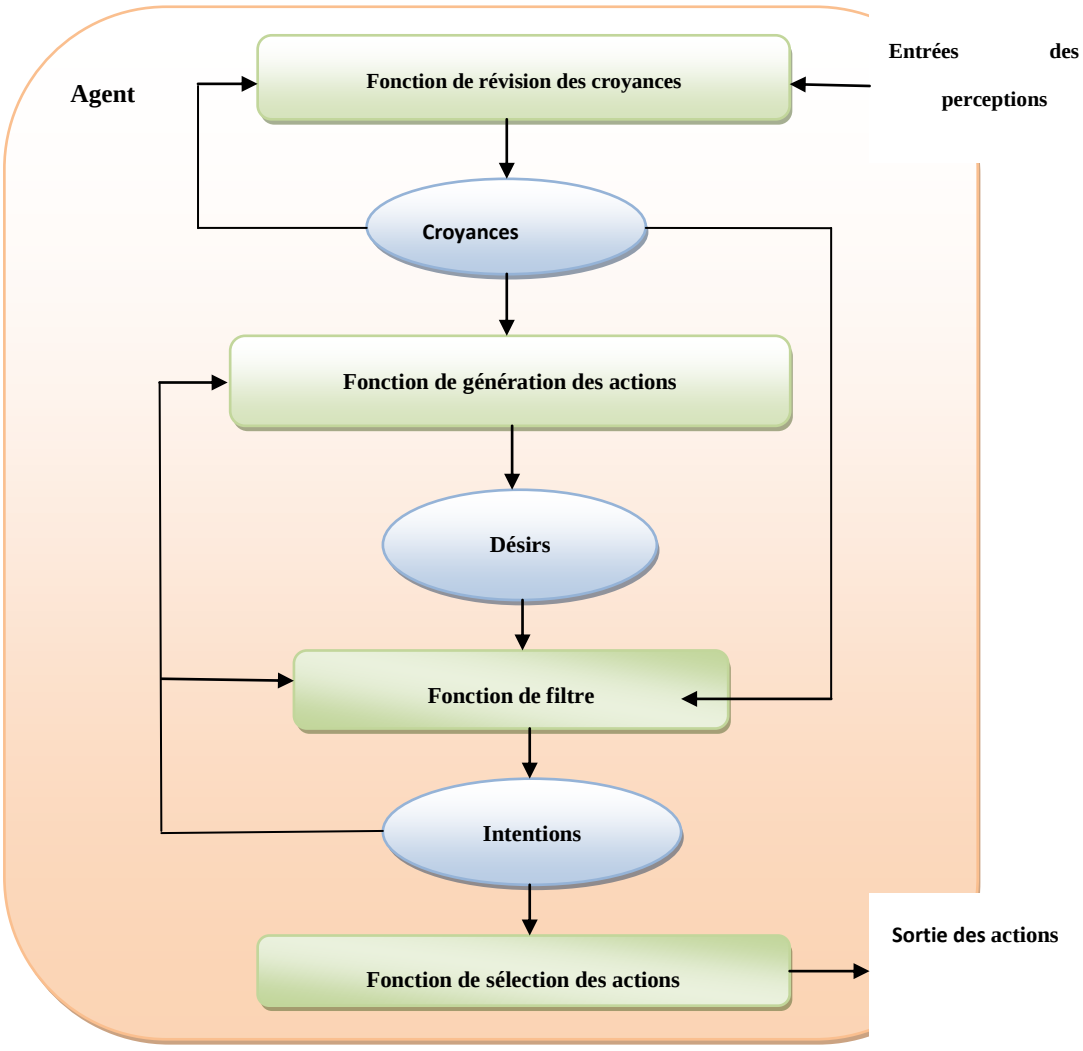

Figure I-2 : Architecture interne de l'agent cognitif

Un agent BDI doit donc mettre à jour ses croyances conformément aux informations qui proviennent de son environnement, décider quelles options lui sont offertes, filtrer ces options afin de déterminer des nouvelles intentions et poser ses actions en vue d'atteindre ses intentions.

3.3) Modèle d'agents hybrides

Ce sont des agents ayant à la fois des capacités cognitives et réactives. Ils conjuguent la rapidité de réponse des agents réactifs ainsi que les capacités de raisonnement des agents cognitifs. Dans ce modèle, les agents sont conçus comme étant composés de niveaux hiérarchiques qui interagissent entre eux. Chaque niveau gère un aspect du comportement de l'agent [OJ96] :

– Bas niveau : couche réactive qui est en relation directe avec l'environnement et qui raisonne suivant les informations brutes de ce dernier,
– Niveau intermédiaire : couche mentale qui fait abstraction des données brutes et qui se préoccupe d'un aspect de l'environnement plus évolué,
– Niveau supérieur : couche sociale qui tient compte de l'interaction avec les autres agents.

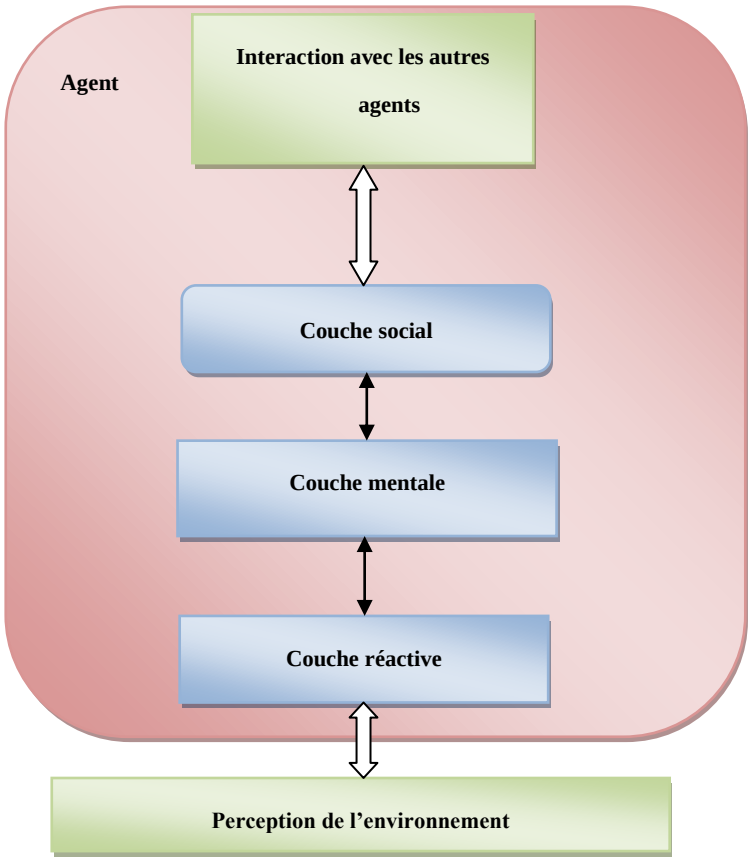

Figure I-3 : Architecture interne de l'agent hybride

3.4) Modèle d'agents mobiles

Un agent mobile est un agent capable de se déplacer d'un site à un autre. Ce type d'agent s'oppose aux agents statiques (ou stationnaires), qui s'exécutent seulement dans le système où ils ont commencé leur exécution. Par contre, un agent mobile n'est pas lié au système dans lequel il débute son exécution. Il peut transporter son état et son code d'un environnement vers un autre où il poursuit son exécution. Il s'agit donc de déplacer les traitements vers les données plutôt que les données vers les traitements. Un agent mobile peut avoir la même architecture qu'un agent réactif ou cognitif, bien qu'il y ait une nécessité d'avoir un module supplémentaire gérant la mobilité de l'agent.

4. Les Systèmes multiagent (SMA)

Ferber (Ferber, 1995) a donné la définition suivante :
On appelle système multiagent, un système composé des éléments suivants:

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

5. Problématique des SMA

L'étude d'un système multiagent est faite sur un ensemble d'axes, nous introduisons la vision donnée par Yves Demazeau appelée la décomposition Voyelles AEIO (**A**gent, **E**nvironnement, **I**nteractions, **O**rganisation)

5.1) Agent

Le concept d'agent a été défini plus haut.

5.2) Environnement

L'environnement est le milieu dans lequel les agents sont introduits. Son rôle est de permettre la communication entre agents, il support les actions des agents en définissant les règles et en renforçant ces actions, il est observable par les agents, et prend en charge l'activité propre des objets et des ressources présentes. Lorsque les agents sont réactifs, l'environnement détient une importance capitale car il est le médiateur de leurs interactions. En effet, comme ces agents ne peuvent communiquer directement entre eux, ils s'influencent mutuellement soit par leurs positions s'ils sont situés, soit par l'intermédiaire d'objets qu'ils perçoivent et modifient.

D'àprès Russel[1995] et Wooldridge et al [2000]Un environnement peut être :

- *Accessible* si un agent peut, à l'aide des primitives de perception, déterminer l'état de l'environnement et ainsi procéder, par exemple, à une action. Si l'environnement est *inaccessible* alors il faut que l'agent soit doté de moyens de mémorisation afin d'enregistrer les modifications qui sont intervenues.
- *Déterministe*, ou non, selon que l'état futur de l'environnement ne soit, ou non, fixé que par son état courant et les actions de l'agent.
- *Episodique* si le prochain état de l'environnement ne dépend pas des actions réalisées par les agents.
- *Statique* si l'état de l'environnement est stable (ne change pas) pendant que l'agent prend des décisions. Dans le cas contraire, il sera qualifié de *dynamique*.
- *Discret* si le nombre des actions faisables et des états de l'environnement est fini.

5.3) Interaction

L'interaction est une notion très complexe et variée dans le domaine des SMA, dans ce qui suit nous donnons quelques aspects concernant cette notion.

Une interaction est une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques (Ferber, 1995). Les interactions s'expriment ainsi à partir d'une série d'actions dont les conséquences exercent en retour une influence sur le comportement futur des agents. Les agents interagissent le long d'une suite d'événements pendant lesquels les agents sont d'une certaine manière en contact les uns avec les autres, que ce contact soit direct ou qu'il s'effectue par l'intermédiaire d'un autre agent ou de l'environnement.

L'interaction est caractérisée par ; les langages de communication et les protocoles d'interaction

On distingue différents types d'interaction que les agents peuvent adopter tels que : la coopération, la négociation et la coordination. Ces types sont à la base :

- Des buts des agents qui peuvent être compatible ou incompatible,
- Des ressources extérieures pour la réalisation des tâches des agents,
- Des capacités des agents par rapport aux tâches à accomplir et enfin
- De la situation d'interaction qui est la combinaison des trois facteurs précédents (but, ressource, compétence).

Le tableau suivant illustre ces types d'interactions

Buts	**Ressources**	**Compétences**	**Types de situation**	**Catégorie**
Compatibles	Suffisantes	Suffisantes	*Indépendance*	Indifférence
Compatibles	Suffisantes	Insuffisantes	*Collaboration simple*	Coopération
Compatibles	Insuffisantes	Suffisantes	*Encombrement*	
Compatibles	Insuffisantes	Insuffisantes	*Collaboration coordonnée*	
Incompatibles	Suffisantes	Suffisantes	*Compétition individuelle pure*	Antagonisme
Incompatibles	Suffisantes	Insuffisantes	*Compétition collective pure*	
Incompatibles	Insuffisantes	Suffisantes	*Conflits individuels pour des ressources*	
Incompatibles	Insuffisantes	Insuffisantes	*Conflits collectifs pour des ressources*	

5.4) Organisation

L'organisation désigne un ensemble d'agents travaillant ensemble pour la réalisation d'une ou plusieurs tâches. Le concept d'organisation peut être exprimé à partir des concepts plus élémentaires d'agent et de tâches. Par rapport au concept de tâche, l'organisation désigne les processus qui permettent la décomposition des tâches en sous-tâches, l'allocation des tâches aux agents et l'accomplissement des tâches dépendantes de façon cohérente. Par rapport au concept d'agent l'organisation détermine les statuts et les comportements sociaux des agents (les rôles) et les relations

qui permettent d'unir les agents au sein d'un groupe, que ce soit vis-à-vis de la décision (les liens d'autorité) ou vis-à-vis de la coordination (les liens d'engagement)[Bouron, 92].

La structure d'une organisation peut être simplement définie comme la somme totale des moyens employés pour diviser le travail en tâches distinctes et pour assurer la coordination nécessaire entre les tâches. (GUTKNECHT, 2001).

6. Infrastructure de l'interaction

L'implémentation de l'interaction exige une infrastructure incluant des langages de communication et des protocoles d'interaction.

6.1) La communication

La communication est un élément important dans le processus d'interaction des agents. Elle permet l'échange des informations. Le but de la communication est de produire un effet sur les destinataires pour exécuter une action demandée par l'émetteur ou répondre à une question. Elle peut être directe ou indirecte.

- ***La communication indirecte***, elle se fait à travers l'environnement ou bien par le biais d'un tableau noir.

Dans une communication par environnement, les agents laissent des traces ou des signaux qui seront perçus par les autres agents. Ce type de communication est adopté par les agents réactifs

Dans une communication via un tableau noir les agents utilisent une mémoire partagée pour déposer leurs messages. Dans ce genre de communication il n'y a pas de destinataire bien défini.

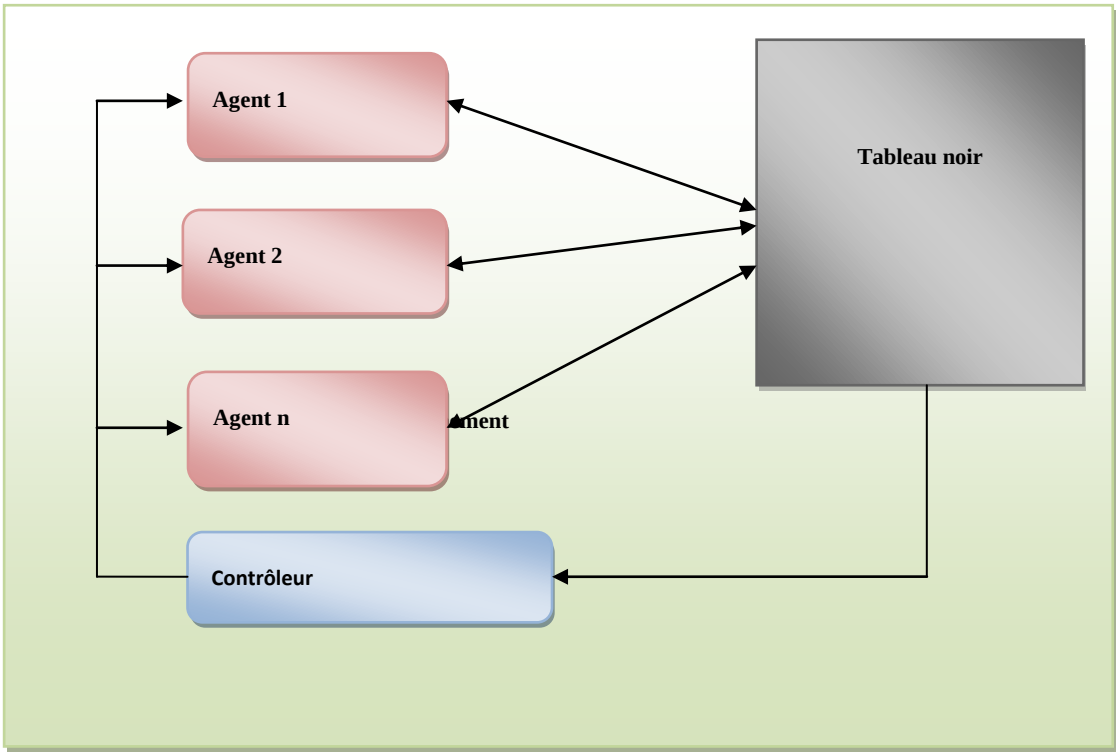

Figure I.4: Communication par tableau noir

L'inconvénient de ce type est qu'il centralise toutes les communications en un point (goulot d'étranglement), et donc amoindrit l'autonomie des agents.

– ***La communication directe***, est spécifique aux agents cognitifs. Elle est intentionnelle et se fait par l'envoi de messages à un (mode point-à-point) ou plusieurs destinataires (mode diffusion). Elle se base sur trois éléments essentiels :

- Le langage de communication : il permet de structurer les messages échangés entre les agents.
- L'ontologie : elle permet de rajouter un aspect sémantique pour les messages échangés entre les agents.

– Protocole d'interaction : il permet de définir un ordre sur les messages échangés entre les agents.

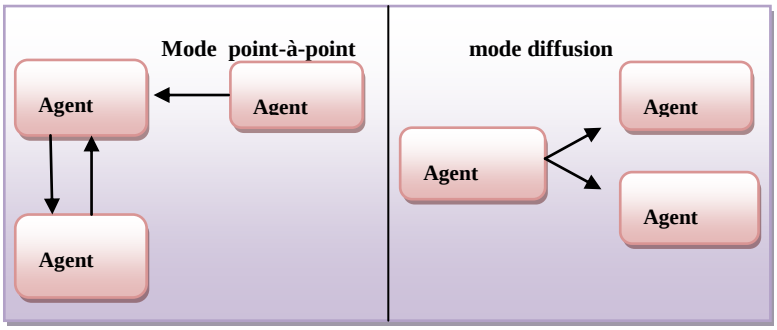

Figure I.5 : Communication Directe (envoie de message)

6.2) Les langages de communication entre agents

– ***Le langage KQML (Knowledge Query and Manipulation Language)***

Le langage KQML est un langage de communication conçu pour l'échange entre les agents, des informations, des connaissances ou des services. Les primitives du langage KQML sont appelées également des *performatifs* qui définissent les actions que les agents peuvent faire pour communiquer les uns avec les autres. Ainsi, un message KQML est constitué essentiellement d'un verbe performatif associé à un contenu dont le format de représentation n'est pas imposé par le langage. L'exemple qui suit représente un échange de message entre deux agents Agent1 et Agent2 le premier demande le prix d'un ordinateur de type HP-530 et l'autre lui répond.

```
(ask-one
         :sender Agent1
         :receiver Agent2
         :content (val (prix HP-530))
         :language KIF
         :ontology ordinateur
         )
(ask-one
         :sender Agent2
         :receiver Agent1
         :content(=(prix HP-530)(scalar54000 DA))
         :language KIF
         : ontology ordinateur
         )
```

– ***Le langage ACL (Agent Communication Langage)***

Le langage ACL est proposé dans le cadre d'un travail de standardisation mené au sein l'organisation *FIPA* (*Foundation of Intelligent Physical Agents*) qui a bénéficié grandement des résultats de recherche de KQML. La syntaxe du message est assez similaire à celle de KQML. *ACL* possède deux langages différents.

Le langage externe qui définit la signification intentionnelle du message.

Le langage interne (ou le contenu) décrit l'expression à laquelle s'appliquent les croyances, les désirs et les intentions des agents.

ACL est fondé sur vingt et un actes communicatifs, exprimés par des performatifs, qui peuvent être groupés selon leurs fonctionnalités de la façon suivante :

- Passage d'information: *Inform, Inform-if, Inform-ref, Confirm, Disconfirm.*
- Réquisition d'information : *Query-if, Query-ref, Subscribe,*
- Negotiation : *Accept-proposal, Cfp, Propose, Reject-proposal,*
- Distribution de tâches (ou exécution d'une action) : *Request, Request-when, Requestwhenever, Agree, Cancel, Refuse,*

- Manipulation des erreurs : *Failure, Not-understood.*

Voici un exemple de message *ACL* qui est envoyé par l'agent *A* à l'agent *B* :

```
(inform
:sender A
:receiver B
:reply-with devis12
:language Prolog
:ontology Ordinateur
:content prix(HP,54000 DA))
:conversation-id conv01
:reply-by 10 min)
```

L'agent *A* informe son interlocuteur que le prix d'un ordinateur *HP* est de 1500 euro. Le contenu du message est exprimé avec le langage *Prolog*. L'ontologie utilisée est celle des ordinateurs. Ce message fait partie d'une conversation ayant comme identifiant *conv01*. L'agent *B* est contraint par une durée de 10 minutes pour donner suite à ce message.

La différence entre *ACL* et *KQML,* réside dans les actes communicatifs. *ACL* contient un ensemble d'actes communicatifs normatifs, qui peuvent être primitifs ou composés. Les nouveaux actes communicatifs ne peuvent être définis qu'en combinant les actes existants et en utilisant les opérateurs prédéfinis. Ceci permet de maintenir l'intégrité sémantique du langage.

6.3) L'ontologie

Dans les systèmes multi agents une ontologie définit le vocabulaire dans un domaine donné pour que les agents puissent se comprendre. Une ontologie est composée de prédicats, d'actions et de concepts.

- Un concept permet de donner un sens à l'information. Par exemple, un ordinateur est une entité physique ayant un nom, une marque, etc.
- Une action est constituée d'une opération et de la définition de celle-ci. Par exemple, une action peut être achetée un ordinateur d'un acteur précis, dans une certaine tranche de prix et dans un magasin précis.
- Un prédicat est une opération qui nécessite une réponse positive ou négative. Par exemple l'ordinateur de tel marque est disponible ou non dans le stocke.

6.4) Les protocoles d'interaction

Afin de structurer la communication entre les agents, des protocoles d'interaction sont élaborés. Ils permettent de décrire explicitement les échanges de messages entre les agents. Ils représentent un schéma commun de conversation utilisé pour exécuter une tâche. Un protocole décrit les enchaînements possibles de messages entre les agents. Nous décrivons dans ce qui suit certains protocoles d'interaction de FIPA.

– **Le protocole FIPA Contract-Net**

Dans ce protocole, les agents peuvent prendre deux rôles: *manager* ou *contractant*.

Le manager est responsable de la surveillance de l'exécution d'une tâche et du traitement des résultats de cette exécution. Un contractant fait une proposition au manager pour réaliser la tâche. En cas d'acceptation, il est responsable de l'exécution concrète de cette tâche. Ce protocole facilite le contrôle distribué de l'exécution des tâches coopératives.

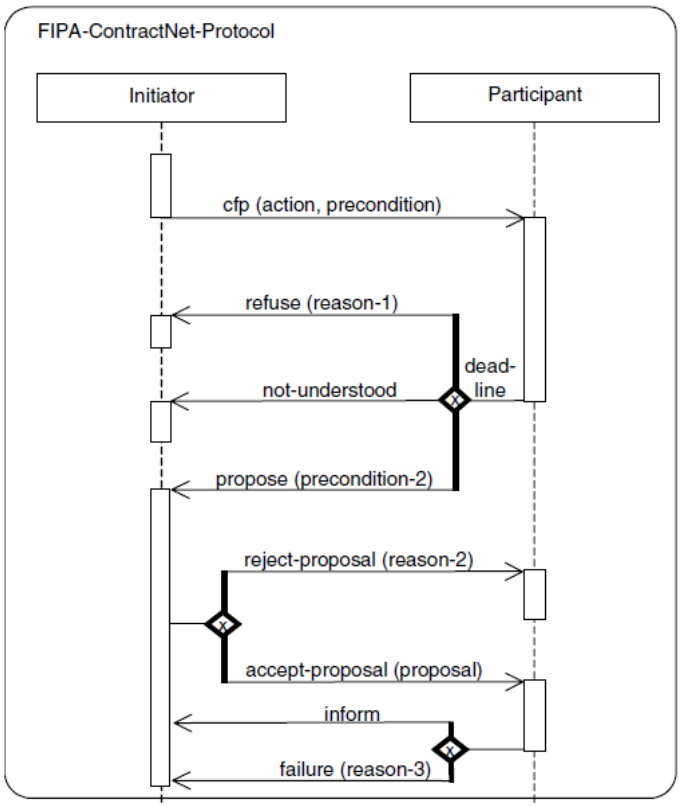

Figure I-6 : Diagramme d'interaction de protocole FIPA ContractNet (FIPA, 2003)

- **Le protocole FIPA-Propos**

Un Initiateur envoie un message à un Participant lui indiquant qu'il effectuera une action si le participant est d'accord. Le Participant répond par un refus ou un accord. Lorsque l'accord est reçu, l'Initiateur doit effectuer l'action et retourner un résultat.

- **Le protocole FIPA-Subscribe**

Un Initiateur envoie un message à un Participant lui demande s'il souhaite souscrire. Le participant répond par un accord ou un refus. En cas d'accord, le Participant envoie les informations répondant à la souscription jusqu'à ce que l'Initiateur annule la souscription ou que le Participant émette un message de type échec.

- **L'enchère anglaise**

Dans ce type d'enchère le commissaire-priseur initie l'enchère en annonçant un premier prix initialement inférieur à la valeur estimée du produit. Après chaque annonce, le commissaire-priseur attend pendant un certain temps pour voir s'il existe des participants acceptant cette offre. Dès qu'un participant se manifeste, le commissaire-priseur annonce une nouvelle proposition de prix égale au dernier prix incrémenté d'une valeur appelée *pas d'incrémentation*.

L'enchère se termine quand aucun des participants ne se manifeste. A ce moment là, le commissaire-priseur compare le prix de la dernière offre acceptée avec la valeur estimée du produit. Si le prix de l'offre est supérieur à la valeur estimée alors l'enchère est accordée au participant qui s'est prononcé pour cette offre. Dans le cas contraire, les participants sont informés de l'annulation de la vente. Le manager peut réexaminer au cours de l'enchère la valeur estimée du produit, pour éviter toute annulation de la vente.

- **L'enchère hollandaise (ou allemande)**

Dans ce mode d'enchère le commissaire-priseur annonce au départ une valeur très élevée par rapport à la valeur estimée du produit. Puis il commence par baisser le prix jusqu'à ce qu'un participant signale son acceptation du prix proposé. Le vendeur possède une estimation de la valeur du produit (*prix de réserve*) au-dessous de laquelle il ne vend pas le produit. Si l'enchère atteint le prix de réserve sans qu'il y ait d'acheteurs, l'enchère se termine.

7. Modes d'interaction

7.1) La coopération

La coopération se traduit par le fait qu'un ensemble d'agents travaillent ensemble pour satisfaire un but commun ou individuel. L'ajout ou la suppression d'un agent influe considérablement sur la performance du groupe. Le besoin de faire coopérer des agents, vient essentiellement du fait qu'un agent ne peut atteindre son objectif individuellement et a besoin de l'aide des autres agents du système. La coopération a comme devise « diviser pour régner », c'est-à-dire que la tâche va être décomposée entre plusieurs agents et ceci peut être réalisé par plusieurs mécanismes comme le Contract Net ou encore la planification multi-agents qui consiste en une technique adoptant un organe centralisé qui se charge de la réalisation des plans et de la gestion des conflits.

7.2) La coordination

On parle d'une coordination entre les agents lorsqu'il y a une interdépendance dans leurs buts, leurs actions ou les ressources qu'ils utilisent.

A ce stade, coordonner les activités des agents devient un aspect essentiel afin d'éviter des problèmes dans le comportement global du système. En effet, la coordination met de l'ordre dans le processus global effectué par des agents. Elle consiste à synchroniser leurs activités ou à régler les conflits qui existent entre eux.

La coordination d'actions est nécessaire pour les raisons principales suivantes :

- Les agents ont besoin d'informations et de résultats que seuls d'autres agents peuvent fournir.
- Les ressources sont limitées. On ne fait parfois attention aux actions des autres que parce que les ressources dont on dispose sont réduites et que d'autres utilisent ces mêmes ressources.
- Optimiser les coûts. Coordonner des actions permet aussi de diminuer les coûts en éliminant les actions inutiles et en évitant les redondances d'action.
- Permettre à des agents ayant des objectifs distincts mais dépendants les uns des autres de satisfaire ces objectifs et d'accomplir leur travail en tirant éventuellement parti de cette dépendance.

7.3) La négociation

Un système multi-agents est composé de plusieurs agents. Ces derniers peuvent entrer en conflit pour plusieurs raisons : conflits d'intérêts ou de buts, accès à des ressources ou proposition de plusieurs solutions différentes à un seul problème. Afin de résoudre ces conflits et de trouver une situation convenable, les agents négocient entre eux en faisant des concessions ou en cherchant des alternatives. Les deux grands piliers de la négociation sont la communication et la prise de décision. Afin de résoudre les conflits, les systèmes multi-agents peuvent adopter plusieurs méthodes que nous pouvons classer en deux grandes familles :

- Les méthodes centralisées : elles reposent sur la négociation via un médiateur ou sur des protocoles tels que le Contract Net Protocol,
- Les méthodes décentralisées : elles sont plus utilisées par les agents BDI telles que la recherche négociée, la méthode heuristique ou encore la négociation basée sur l'argumentation.

8. Méthodologies de conception des SMA

8.1) La méthodologie voyelles « AEIO »

Cette méthodologie repose sur quatre concepts, identifiés par les quatre voyelles (Jarraya, 2006): **A** pour Agents, **E** pour Environnements, **I** pour Interactions et **O** pour Organisations. Chaque voyelle correspond à un ensemble de modèles combinés pour aboutir à la réalisation du système multiagents :

- **Agents** : concernent les modèles (ou les architectures) utilisés pour la partie active de l'agent.
- **Environnement :** et le milieu dans lequel sont plongés les agents, ils permettent de définir l'ensemble des capacités de perception et d'actions des agents.
- **Interaction :** concernent les infrastructures, les langages et les protocoles d'interactions entre agents.
- **Organisation** : structure les agents en groupes, hiérarchies, relations, etc.

La méthodologie « AEIO » se décompose en trois phases qui sont :

- **La phase d'analyse,** qui permet d'identifier et de décomposer le problème en quatre composantes fondamentales : **A**gents, **E**nvironnement, **I**nteractions et **O**rganisation, pour dégager l'architecture générale du système.
- **La phase de conception,** qui permet de choisir les modèles opérationnels des composantes pour aboutir à la spécification du fonctionnement global du système exprimé sous forme de diagrammes de comportement ; Les modèles d'agents ou architectures d'agents vont de modèles simples comme les automates à états finis

(agents réactifs) aux modèles plus complexes comme les systèmes à base de connaissances (agents cognitifs) ; Les modèles d'environnement dépendent du domaine d'application, ils sont généralement spatialisés et dotés d'une métrique ; Les structures et langages d'interaction vont des modèles issus de la physique, comme les modèles à base de forces, à des interactions de haut niveau basées sur la théorie des actes de langages ; Enfin, les modèles d'organisations vont des modèles biologiques jusqu'aux modèles inspirés par les lois sociales.

- **La phase de programmation :** (ou implémentation) consiste en l'instanciation des modèles, en utilisant des plates-formes et des langages choisis ; Le résultat, le système implémenté, peut alors être exécuté, évalué (test et validation) et repensé en cas d'inadéquation avec les besoins exprimés par le type de problème et le domaine d'application **[PICA, 04].**

8.2) La Méthodologie GAIA

Cette méthodologie est basée sur l'idée qu'un système multiagent doit être vu comme une organisation "computationnelle" reposant sur les interactions entre les différents rôles présents dans le système. Son processus de développement comprend deux importantes phases, l'analyse et la conception.

Selon (Dignum, 2009), GAIA applique un modèle de développement en cascade.

- **La phase d'analyse** : qui produit deux modèles : le modèle de rôles et le modèle d'interactions.

- **Le modèle de rôles :** un rôle dans Gaia est définit par quatre attributs :

i. **Les responsabilités** : ce sont "les invariants" du rôle, qui se décomposent en deux catégories :

1. **L'animation,** un comportement à déclencher selon certaines conditions.
2. **La sécurité,** qui consiste à s'assurer que certaines conditions restent valides.

ii. **Les permissions** : Ce sont les droits accordés à ce rôle ; Ces droits concernent principalement les droits d'accès à l'information.

iii. **Les activités** : qui correspondent aux comportements "privés" de l'agent, c'est-à-dire des comportements pouvant être déclenchés sans que l'agent soit en interaction ;

iv. **Les protocoles**: Ils définissent la structure des interactions entre les rôles.

- **Le modèle d'interaction : U**ne interaction est représentée comme un protocole entre différents rôles, mais il n'y a pas de description précise des messages échangés ou encore de leur flux.

- **La phase de conception** : Elle transforme les modèles abstraits de la phase d'analyse en modèles plus spécifique, qui sont :

- **Le modèle d'agent :** Il est représenté par une arborescence, dont les racines correspondent aux différents types d'agents qui existent dans le système et les feuilles correspondent aux rôles des ses agents ;
- **Le modèle de service :** Il énumère les différents services (fonctions) assurés par chaque rôle;
- **Le modèle de relation:** Il représente les liens de communication entre les différents types d'agents, il est représenté par un graphe dont les nœuds sont les types d'agents et les arcs sont les liens de communication.

Cette méthodologie est intéressante sur bien des points, mais reste trop générale pour pouvoir facilement passer de la phase de conception du système à sa réalisation**[**YANN**,** 03].

8.3) La méthodologie AGR (Agent/Groupe/Rôle)

Cette méthodologie proposée par Jaques Ferber est basée sur un concept simple : On suppose qu'un **agent** possède un ou plusieurs **rôles**, l'agent appartient à un ou plusieurs **groupes,** et un groupe contient un ou plusieurs rôles (Garneau, 2003).

- Un agent est une entité communicante qui joue un ou plusieurs rôles dans des groupes (un ensemble atomique d'agents). Chaque agent peut appartenir à différents groupes et les groupes peuvent se chevaucher.
- Un rôle dans AGR est une représentation abstraite de la fonction, du service ou tout simplement d'identificateur d'un agent au sein d'un groupe. Chaque agent peut tenir plusieurs rôles, mais chaque rôle est spécifique à un groupe. La communication entre les agents est possible grâce aux rôles qu'ils assument et le contrôle sur les communications est effectué par le groupe.

Dans AGR, la notion d'organisation correspond à une relation structurelle entre les rôles définis au sein des groupes, et la notion d'interaction n'est pas explicitement représentée ni manipulable dans le système.

Les principales étapes de cette méthode sont :

- **l'analyse**, qui permet d'identifier les fonctions du système et les dépendances au sein de communautés identifiées ; Il convient de définir quels sont les mécanismes de coordination et d'interaction entre les entités d'analyse ;
- **la conception**, qui contient l'identification des groupes et des rôles dans des diagrammes de structures organisationnelles ;

- **la réalisation**, qui commence par le choix de l'architecture d'agents suivie de la gestion des entités du domaine qui permet d'implanter le système à partir d'organisations concrètes ; Le mieux étant d'utiliser MadKit comme plate-forme de prototypage et de simulation**.**

9. Les plates-formes orientées Agent

Il existe une multitude de plates-formes multi-agents dédiées à différents modèles d'agent. Les plates-formes fournissent une couche d'abstraction permettant de facilement implémenter les concepts des systèmes multi-agents. D'un autre coté, elle permet aussi le déploiement de ces systèmes. Ainsi, elles constituent un réceptacle au sein duquel les agents peuvent s'exécuter et évoluer. En effet, les plates-formes sont un environnement permettant de gérer le cycle de vie des agents et dans lequel les agents ont accès à certains services.

Comme le choix d'une plate-forme d'agent a une grande influence sur la conception et la mise en œuvre des agents, FIPA a produit les normes qui décrivent comment une plate-forme d'agent devrait être. Ces normes existent pour assurer une conception uniforme des agents indépendamment de la plate-forme.

Pour comparer les différentes plateformes, nous appliquons un cadre de comparaison qui se base sur les éléments suivants :

- les types de systèmes visés : certaines plates-formes sont conçues pour développer des applications dans des domaines spécifiques tel que la simulation, les systèmes distribués, etc.
- les modèles d'agents supportés : certaines plates-formes imposent un modèle d'agent spécifique (agent BDI, agent collaboratif, etc.).
- les méthodologies : certaines plates-formes proposent une méthodologie de développement, d'autres plates-formes sont utilisées par certaines méthodologies orientées agent.
- le langage d'implémentation appliqué : généralement, le langage d'implémentation qui est adopté par les plates-formes est Java. Nous étudierons les classes abstraites qui sont fournies par la plate-forme, qui aident à architecturer les applications.

9.1) La plate-forme ZEUS

Zeus est une plate-forme dédiée pour la construction rapide d'applications à base d'agents collaboratifs. Elle se prête bien aux systèmes économiques qui utilisent des applications de planification ou d'ordonnancement. Pour implémenter les agents collaboratifs, Zeus se base principalement sur les concepts agents, buts, tâches (que les

agents doivent réaliser pour atteindre leurs buts) et faits (qui représentent les croyances des agents). Un agent dans Zeus est constitué en trois couches : la couche de définition, qui contient les capacités de raisonnement et des algorithmes d'apprentissage, la couche organisationnelle, qui contient la base de connaissances et des accointances de l'agent, et la couche de coordination, qui définit les interactions avec les autres agents. Zeus propose aussi un ensemble d'agents utilitaires (serveur de nommage et facilitateur) pour faciliter la recherche d'agents. Zeus fournit aussi une méthodologie qui se base sur quatre phases pour la construction d'agents :

- l'analyse du domaine : consiste à modéliser des rôles. A ce stade aucun outil logiciel n'est fournit et le langage UML est utilisé.
- la conception : consiste à spécifier des buts et des tâches qui permettent de les réaliser. Cette spécification se fait par le langage naturel et n'est supportée par aucun outil.
- le développement : Cette phase est réalisée en quatre étapes : création de l'ontologie, création des agents, configuration de l'agent utilitaire, configuration de l'agent tâche et implémentation des agents. Des outils supportant des notations graphiques sont fournies afin d'aider à l'élaboration de ces étapes.
- le déploiement : dans cette phase, le système multi-agents est lancé. Un outil de visualisation permet le suivi de l'exécution. Cet outil permet de visualiser l'organisation, les interactions ayant lieu dans la société d'agents, la décomposition des tâches et l'état interne des agents.

Zeus fournit un environnement de développement d'agents grâce à un ensemble de bibliothèques Java que les développeurs peuvent réutiliser pour créer leurs agents.

9.2) La plate-forme JADE

La plate-forme JADE (Java Agent DEvelopment framework) [BPR99] est certainement celle qui est la plus utilisée par la communauté des systèmes multi-agents.

JADE permet de développer et d'exécuter des applications distribuées basées sur le concept d'agents et d'agents mobiles. Elle est compatible à la plate-forme FIPA. Les agents dans JADE sont implémentés selon 6 propriétés :

- Autonomie : les agents ont leurs propres threads de contrôle qui leur permet de contrôler leurs actions, de prendre leurs propres décisions afin de réaliser leurs buts mais aussi de contrôler leur cycle de vie.
- Réactivité : les agents peuvent percevoir les évènements de leur environnement et réagissent en fonction de ces évènements,

- Aspects sociaux : les agents exhibent des aspects sociaux qui leur permettent de communiquer et d'interagir entre eux. La communication se fait à travers le passage de messages asynchrones. La communication est considérée comme un type d'actions et peuvent de ce fait intégrer un plan d'actions. Les messages ont une sémantique et une structure définis par le standard FIPA.
- Dynamicité : les agents ont la possibilité de découvrir dynamiquement d'autres agents et de communiquer avec eux.
- Offre de service : chaque agent offre un ensemble de services, il peut enregistrer ses services et les modifier. Il a aussi la possibilité de chercher des agents qui offrent les services dont il a besoin.
- Mobilité : les agents dans JADE ont la possibilité de se déplacer. Les agents sont implémentés dans des conteneurs et ils peuvent se déplacer.

Un composant complémentaire est ajouté à JADE pour permettre d'intégrer l'implémentation de l'architecture interne des agents ; il s'agit de Jadex [PBL05] [SBPL06] qui prend en compte l'architecture des agents hybrides (c'est-à-dire des agents qui sont à la fois réactifs et proactifs). Pour les agents proactifs, JADEX se base sur le modèle BDI. JADE assure la sécurité en offrant aux applications des systèmes d'authentification qui vérifient les droits d'accès des agents.

JADE n'offre pas de méthodologie, par contre plusieurs méthodologies la prennent comme plate-forme cible lors de la génération de code tel que Gaia et PASSI.

L'implémentation de JADE est basée sur Java. La plate-forme peut être répartie sur un ensemble de machines et configurée à distance. La configuration du système peut évoluer dynamiquement puisque la plate-forme supporte la mobilité des agents.

9.3) La plate-forme MADKIT

La plate-forme MadKit (Multi-Agents Developement kit) [FG00] est développée à l'Université de Montpellier II. Bien qu'elle puisse supporter le développement de divers systèmes, elle semble bien adaptée pour les applications de simulation. La plate forme MADKIT est basée sur les concepts agent, groupe et rôle. Ces concepts sont appliqués de la manière suivante :

- L'agent : il décrit comme une entité autonome communicante qui joue des rôles au sein de différents groupes. La faible sémantique associée à l'agent est volontaire l'objectif étant de laisser le choix au concepteur pour choisir l'architecture interne appropriée à son domaine applicatif.

- Le groupe : chaque agent peut être membre d'un ou plusieurs groupes. Il sert à identifier la structure organisationnelle d'un système multiagent usuel.
- Le rôle : il est considéré comme une représentation abstraite d'une fonction ou d'un service. Chaque agent peut avoir plusieurs rôles et un même rôle peut être tenu par plusieurs agents. Les rôles sont locaux aux groupes.

Cette implémentation correspond à la conception réalisée au niveau de la méthodologie AALAADIN. En effet, à partir des concepts AGR, cette méthodologie définit une démarche de développement axée sur la spécification du cadre organisationnel des applications multi-agents. Cette démarche définit l'ensemble des rôles possibles, spécifie les interactions, et décrit les structures abstraites de groupe.

MadKit fournit une API permettant la construction d'agent en spécialisant une classe d'agent abstraite. Les agents sont lancés par le noyau de MadKit, qui propose notamment les services de gestion des groupes et de communication.

L'échange des messages se fait à travers le rôle que l'agent est en train de jouer.

9.4) La plate-forme AgentBuilder

AgentBuilder est une suite intégrée d'outils permettant de construire des agents intelligents. Cette plate-forme est adaptée pour tous types de systèmes. L'élaboration du comportement des agents se fait à partir du modèle BDI et du langage AGENT-0. KQML est utilisé comme langage de communication entre les agents. AgentBuilder est composé d'une interface graphique et d'un langage orienté agent permettant de définir des croyances, des engagements et des actions. Il permet également de définir des ontologies et des protocoles de communications inter-agents. Les agents sont décrits avec le langage Radl (Reticular Agent Definition Language), qui permet de définir les règles du comportement de l'agent. Les règles se déclenchent en fonction de certaines conditions et sont associées à des actions. Les conditions portent sur les messages reçus par l'agent tandis que les actions correspondent à l'invocation de méthodes Java. Il est aussi possible de décrire des protocoles définissant les messages acceptés et émis par l'agent.

AgentBuilder propose l'utilisation de la méthode OMT durant la phase d'analyse. OMT permet de spécifier des objets du domaine et les opérations qu'ils peuvent effectuer. A partir de cette spécification une ontologie du domaine est créée. Des outils graphiques sont fournis pour effectuer ces tâches. Durant la phase de conception, les agents sont identifiés et des fonctionnalités leur sont assignées. Leurs rôles et leurs caractéristiques sont définis ainsi que les protocoles d'interaction auxquels ils

participent. Durant la phase de développement, le comportement de l'agent est défini ainsi que ses croyances, intentions et capacités. Durant le déploiement, le code de l'agent est interprété par le *Run-Time Agent Engine*.

10. Conclusion :

Ce chapitre nous a permis de comprendre les concepts et les notions essentiels du paradigme agent ainsi que les différents outils logiciels disponibles pour l'implémentation d'un modèle à base d'agents.

Nous n'avons pas l'intention de faire une étude comparative mais seulement avoir une réflexion sur l'outil et la démarche convenables pour atteindre nos objectifs.

La réalisation des objectifs fixés dans ce travail est basée sur l'utilisation de la technologie agent pour traiter les problèmes liés aux processus dans le contexte du workflow interorganisationnel lâche ce qui exige une connaissance profonde de cette discipline : c'est l'objet du chapitre suivant.

Chapitre II : Les workflows interorganisationnels; Définitions, modèles et formes d'interopérabilité

1. Introduction

Dans ce chapitre nous allons parler des workflow. Cette technologie pluridisciplinaire introduite dans le domaine des organisations résulte de la combinaison de plusieurs domaines tels que le génie logiciel, la modélisation de l'entreprise, la composition des services web, etc.

Pour clarifier ce concept nous allons commencer par une définition donnée par WfMC[1] une des organisations internationales de normalisation dans ce domaine nous présentons aussi le modèle de référence le plus dominant et les types des workflow existants.

Nous terminons ce chapitre par l'exposé d'un ensemble d'éléments importants pour une modélisation réussite d'un workflow tels que les différents aspects de modélisation, les formes d'interopérabilité, les langages de spécification et les patrons de contrôle de flux des workflow.

2. Définitions de workflow

Selon [23],

- le workflow est l'automatisation totale ou partielle de l'exécution de processus métier, exécution au cours de laquelle des documents, des informations des tâches passant d'un participant à un autre pour exécuter des actions précises selon des règles prédéfinies.
- *Processus* est un ensemble coordonné d'actions ou d'opérations reliées, en série ou en parallèle, dans le but d'atteindre un objectif commun.
- Processus métier est un processus qui systématise l'organisation et la politique d'une entreprise dans le but d'atteindre certains des objectifs de cette entreprise.
- Un workflow est un ensemble d'actions (étapes) s'enchainant dans un ordre prédéfini. Ces actions peuvent s'enchaîner en fonction des conditions, d'interactions avec d'autres workflow ou en fonction d'interactions humaines. Les actions sont appelées également des activités. Une activité est un composant réutilisable représentant une étape d'un workflow.
- Un workflow est une représentation d'un procédé métier dans un format interprétable par la machine. Il est constitué d'un réseau d'activités et de dépendances entre elles, des critères pour spécifier le démarrage et la terminaison d'un procédé et des informations sur les activités individuelles (participants, applications, données informatiques associées, etc.). Les activités et les procédés ont des données en entrée et

[1] WfMC(Workflow Management Coalition)

en sortie. Ces données sont représentées comme des ensembles d'éléments de données, appelés conteneurs.

3. Système de gestion de workflows

Un système de gestion de Workflow est un système complet qui sert à définir, gérer et exécuter des procédures en exécutant des programmes dont l'ordre d'exécution est pré défini dans une représentation informatique de la logique de ces procédures.

4. Le modèle de référence pour le workflow

Le modèle de référence identifie les composants de bases, ainsi que les interfaces qui permettent l'interopérabilité entre les différents produits workflow. Ces composants sont illustrés dans la figure ci-dessous.

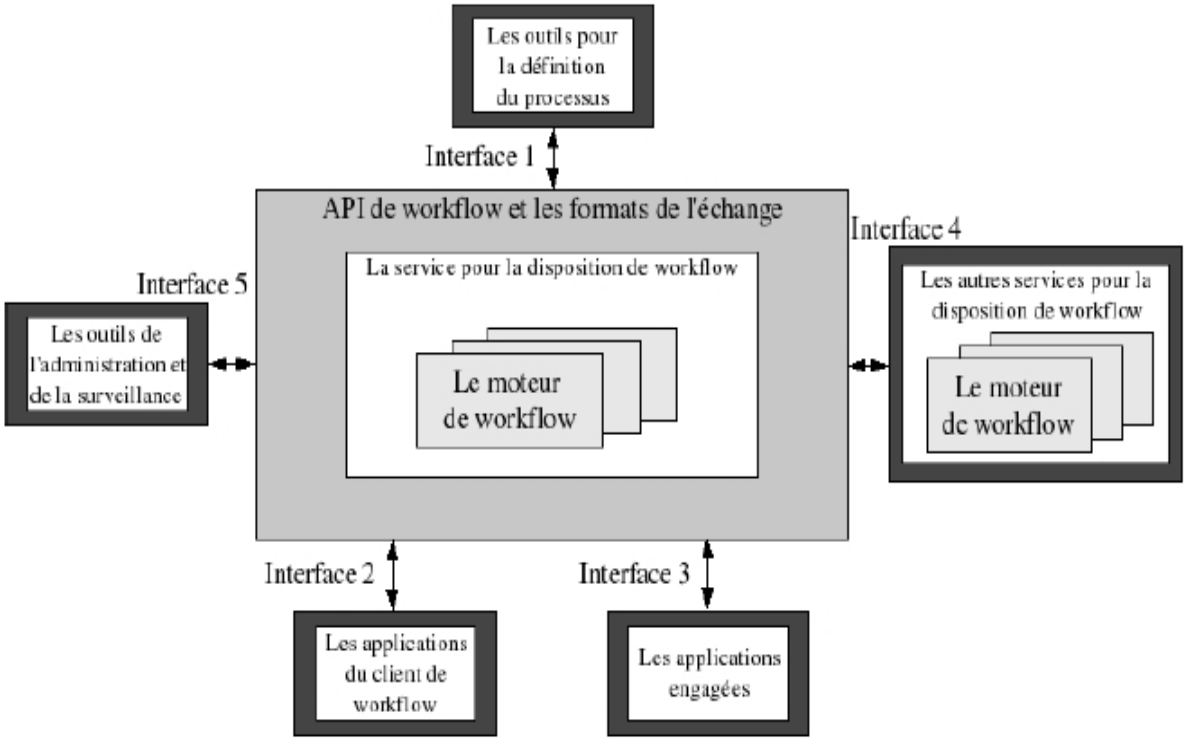

Figure 4.1 : le modèle de référence du workflow

a. Le service pour la disposition de workflow est un service logiciel composé d'un ou plusieurs moteurs Workflow de même type qui servent à définir, gérer et exécuter des procédures Workflow.

b. Le moteur de workflow est un service logiciel qui fournit tout ou partie de l'environnement d'exécution d'un workflow.

c. **L'Interface 1** fournit le lien entre les outils de création et de modification de workflows et le moteur de workflow.
d. **L'Interface 2** permet la communication entre les applications du client de workflow et le moteur de workflow.
e. **L'Interface 3** permet d'invoquer des applications bien déterminées d'une activité donnée afin d'exécuter des tâches spécifiques.
f. **L'Interface 4** permet l'interopérabilité et l'échange de travail entre plusieurs systèmes de gestion de workflows autonomes.
g. **L'Interface 5** permet l'interconnexion entre outils d'administration et de surveillance et le moteur de workflow. Elle est divisée en deux parties : les fonctions de système de gestion de workflow et les fonctions de cheminement de workflow. Les outils d'administration et de surveillance peuvent auditer par exemple les temps d'attente, de complétion et d'exécution ainsi que le routage.

5. Types de workflow

D'après [24, 25] il y a quatre catégories complémentaires de workflow, qui sont:

5.1) Workflow d'administration

Un workflow d'administration permet de gérer des procédures administratives dont les règles de déroulement sont bien établies et connues par chacun au sein de l'entreprise [26, 27]. Il est caractérisé par la circulation de documents/formulaires à travers l'entreprise (par exemple, demande de remboursement de frais de missions, demande d'inscription à une formation, etc.). De ce fait, il aide à transformer la circulation de documents papiers en circulation de documents électroniques. Les workflows d'administration ne sont pas d'une très grande valeur ajoutée pour une entreprise d'une part, et d'autre part, ils sont assez statiques si l'on considère leur degré de répétition élevé.

5.2) Workflow de production

Un workflow de production permet de gérer les processus de production dans l'entreprise. Il constitue généralement le cœur de métier de l'entreprise et la valeur ajoutée de l'entreprise dépend énormément de la qualité de ce workflow. C'est l'efficacité de ce type de workflows qui assure à l'entreprise des avantages compétitifs [28, 29, 30] (par exemple, workflow de livraison pour une entreprise de vente en ligne, procédé de gestion de prêts dans une banque, workflow de gestion des demandes de dommages et intérêts au sein d'une compagnie d'assurance, workflow de gestion de la chaîne de production chez un constructeur automobile, etc.).

5.3) Workflow de collaboration

Un workflow de collaboration [31, 32, 33, 34] permet de gérer la conscience et la collaboration de groupe lors d'un projet de travail créatif (par exemple, conception logicielle, réalisation d'un plan de bâtiment, montage d'un projet, etc.). Il est caractérisé par une forte valeur ajoutée au sein de l'entreprise, mais par un faible degré de répétition. Il se distingue des autres types de workflows par le grand nombre de participants qui interagissent en permanence pour la réalisation du projet en commun.

La complexité de ce type de procédé réside dans la difficulté de modélisation de la méthodologie de travail à suivre surtout qu'il faut prévoir, pendant le déroulement du workflow, l'arrivée de nouveaux participants, la création de nouvelles activités à intégrer et à gérer, etc.

5.4) Workflow ad-hoc

Un workflow ad-hoc [35] représente une classe de workflows destinés à des situations spécifiques où la logique de déroulement à suivre est définie durant l'exécution. Il forme une solution hybride rassemblant des caractéristiques d'administration, de production et de collaboration. Ce type de workflow gère des situations uniques ou causées par des exceptions. Généralement, ce type de workflow ne possède pas de structure prédéfinie, l'étape ultérieure à suivre est déterminée essentiellement par les participants au workflow. Par exemple, si un coordinateur envoie une note d'information aux membres de son équipe, ces derniers ont le choix de faire ce qu'ils veulent avec cette note. Ils peuvent éventuellement la renvoyer à d'autres personnes qu'elle peut intéresser.

5.5) Workflow interorganisationnel (WIO)

Le workflow interorganisationnel est considéré comme une extension du workflow classique puisqu'il envisage la coopération de plusieurs processus issus de plusieurs organisations réparties**,** autonomes et hétérogènes. Il se distingue du workflow traditionnel par trois aspects essentiels :

- La répartition des processus d'organisations.
- L'autonomie des organisations : chaque organisation prend individuellement les décisions concernant les conditions de coopération, c'est à dire quand, comment et avec qui elle doit coopérer.
- L'hétérogénéité des organisations à faire coopérer : cela concerne les différences en termes de modèles et de systèmes.

6. Modélisation des workflow

Un système workflow est modélisé selon plusieurs aspects, selon [4]

6.1) L'aspect fonctionnel

L'aspect fonctionnel concerne l'identification des activités des processus que l'on souhaite modéliser.il permet d'établir la hiérarchie des activités, c'est-à-dire d'exprimer de possibles décompositions en termes de sous-processus. Enfin, le modèle fonctionnel doit aussi représenter le flux de données associées aux activités et les interdépendances de données entre les activités (*data flow*).

6.2) L'aspect comportemental

Il correspond à la dynamique du processus. Le comportement s'exprime par la modélisation d'un contrôle de flux entre les activités. Ce dernier permet d'indiquer la chronologie de l'exécution des activités, leur flux (séquentiel ou parallèle), les points de synchronisation entre activités ou au contraire, les points de disjonction. De plus, il doit représenter les évènements qui permettent de déclencher les activités. Notons pour souligner l'importance de ce modèle, qui permet l'exécution du Workflow. L'aspect comportemental est également appelé aspect de coordination.

6.3) L'aspect informationnel (donnée)

Cet aspect concerne l'ensemble des informations et des données qui sont associées aux activités. Il décrit en détail les relations qui existent entre les données, leur type et leur structure.

6.4) L'aspect organisationnel

Il concerne la description de l'organisation des acteurs de l'entreprise. Le modèle organisationnel peut refléter fidèlement l'organigramme de l'entreprise, c'est-à-dire la décomposition hiérarchique de celle-ci en départements et services ou bien décrire des unités organisationnelles dans lesquelles on identifie des acteurs. Selon la méthode choisie, la description est plus ou moins détaillée et permet d'établir des liens hiérarchiques entre les acteurs ainsi que des relations entre unités organisationnelles ou départements. Toutefois, quelle que soit la méthode retenue, la description des rôles associés aux différentes activités reste invariante. Les rôles créent l'interface entre le modèle organisationnel et les modèles représentant les activités.

7. Formes d'interopérabilité dans les workflows intreorganisationnel

Dans le domaine du Workflow, la WfMC [36] a défini l'interopérabilité des modèles de Workflow comme étant la possibilité qu'ont plusieurs moteurs Workflow de communiquer pour exécuter de manière coordonnée des instances de procédures sur l'ensemble de ces différents moteurs. Dans ce qui suit, nous citons les différentes formes d'interopérabilité des workflows inter-organisationnels.

7.1) Partage de capacité

Dans cette forme d'interopérabilité [37] ; un ensemble de partenaires se mettent d'accord, dès la phase de conception, sur un workflow global mettant en service leurs capacités et savoir-faire mutuels. Le contrôle quant à lui est centralisé et le routage de workflows est assuré et géré par un seul système de gestion de workflow alors que l'exécution des activités est distribuée sur les différents participants. En effet, à chaque fois que le système de gestion de workflow gérant le workflow global rencontre une activité, il fait appel au partenaire adéquat ainsi qu'aux ressources appropriées permettant l'exécution de cette activité.

Le partage de capacité est simple et utilise une infrastructure commune ainsi qu'une gestion centralisée d'un workflow unifié.

Cependant, il présente plusieurs insuffisances vis-à-vis des besoins de coopération requis. Tout d'abord, cette forme d'interopérabilité se base sur un seul scénario de coopération que fixent les différents partenaires dès la phase de conception ce qui rend le système rigide.

Par conséquent, il est difficile de maintenir la cohérence du système ainsi que la mise en place des changements.

Ainsi, le partage de capacité n'est pas adéquat dans le contexte de coopérations spontanées où la connexion et l'interconnexion des partenaires est dynamique et où le scénario de coopération n'est pas défini à priori.

Sur le plan de l'interopérabilité des systèmes de gestion de workflow, le partage de capacité suppose l'utilisation d'une infrastructure commune. Ceci représente une contrainte rigide pour la coopération de workflow puisque les partenaires sont obligés d'adopter cette infrastructure commune.

En outre, cette forme d'interopérabilité, ne préserve pas le savoir-faire des partenaires puisque le workflow global ainsi que les ressources sont partagées et connus par les différents participants.

7.2) Exécution chaînée

L'exécution chaînée [37] consiste à modéliser un workflow global en plusieurs sous workflows disjoints exécutés séquentiellement par les différents partenaires. En effet, chaque partenaire se charge d'une partie de workflow. Une fois cette partie est exécutée, le partenaire transfère le flot au partenaire suivant. Cette forme d'interopérabilité ne requiert pas d'exécution parallèle et se base sur un contrôle distribué de workflow.

L'exécution chaînée est illustrée dans l'exemple de la figure suivante.

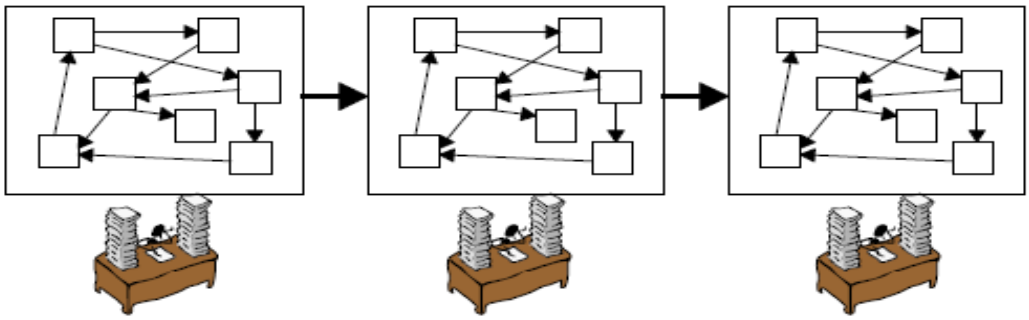

Figure II.2: architecture de l'exécution chainée

L'exemple est formé de trois partenaires chacun disposant de son propre workflow.

Lorsque *Partenaire 1* finit tout son travail, il transfère le flot au *Partenaire 2*. De même, une fois que *Partenaire 2* a fini l'exécution de son workflow, il transfère le flot au *Partenaire 3*.

L'exécution chaînée n'est pas dynamique et définit un seul scénario de coopération, spécifié dès le début, exigeant une exécution séquentielle du workflow par les différents partenaires. Ceci représente une contrainte rigide pour les applications dynamiques dont la coordination peut changer au court du temps en fonction des besoins requis et fournis par les partenaires.

7.3) Sous-traitance

La sous-traitance [37] est une forme d'interopérabilité permettant à un partenaire principal de déléguer l'exécution et la coordination d'une partie de son workflow à

d'autres partenaires. Le contrôle des workflows étant hiérarchique, le partenaire principal voit les sous workflows sous traités comme atomiques alors que ceux-ci peuvent avoir des structures complexes au niveau des partenaires les exécutant.

La sous-traitance se base sur un seul scénario de coopération hiérarchique : un partenaire principal qui délègue un ensemble de ses activités à d'autres partenaires. Bien que cette forme d'interopérabilité préserve le savoir-faire des partenaires en sous-traitant les activités du workflow principal, elle ne supporte pas des coopérations plus complexes avec différents types d'interactions entre les partenaires.

7.4) Transfert de cas

Dans cette forme d'interopérabilité [37, 38] ; les différents partenaires impliqués dans l'interopérabilité partagent une description commune de la définition de workflow. La spécification de workflow est répliquée au niveau de chaque participant.

Chaque partenaire peut exécuter des pièces spécifiques du workflow. Cependant, une activité peut être exécutée par plusieurs partenaires. En outre, le workflow est partitionné verticalement (la même instance de workflow passe d'un partenaire à l'autre pour exécution) et par conséquent, à un instant donné, l'instance de workflow est possession d'exactement un seul partenaire. Cette instance peut passer ensuite d'un partenaire à un autre si l'une des conditions suivantes est satisfaite :

- le partenaire possédant l'instance n'est capable d'exécuter aucune activité de workflow, alors qu'il existe encore des activités pouvant être exécutées par d'autres partenaires.
- pour des raisons d'équilibre de charge ou de configurations proactives.

Le transfert de cas n'est pas dynamique et définit un seul scénario de coopération accepté par les différents partenaires dès la phase de conception du workflow global. De plus, il ne préserve pas le savoir-faire des partenaires puisque la même description du workflow global est répliquée et accessible à tous les participants. En outre, il ne préserve pas les workflows préétablis des partenaires. En effet, les partenaires doivent se conformer au workflow global. Enfin, le transfert de cas n'intègre pas les systèmes de gestion des partenaires. Ces derniers doivent, en effet, être augmentés de la capacité de réception/envoi des instances ainsi que des données d'états.

7.5) Transfert de cas étendu

Dans le transfert de cas [37] ; chaque partenaire utilise la même définition du workflow qui est répliquée au niveau de chaque participant. Ce qui n'est pas réaliste : dans le monde réel, un partenaire peut souhaiter coopérer avec une certaine autonomie interne.

Le transfert de cas étendu est conçu pour pallier ce problème et permet aux partenaires impliqués dans le workflow interentreprises de personnaliser leurs workflows et entrer des variations locales (ajout d'activités par exemple) selon leurs besoins avec la contrainte que le comportement des workflows personnalisés soit identique à celui du workflow global.

Le transfert de cas n'est pas dynamique. En effet, le choix des partenaires et l'implémentation des fonctions de correspondances entre les workflows locaux des partenaires et le workflow global partagé sont difficiles.

Bien que le transfert de cas étendu permette de personnaliser les workflows locaux des partenaires et entre quelques modifications dans leurs structures internes telles que l'ajout et/ou la suppression des activités et des flots de contrôle, cette forme d'interopérabilité ne préserve pas la totalité du savoir-faire des entreprises puisque les participants partagent le workflow global.

Enfin, le transfert de cas étendu n'intègre pas les systèmes de gestion de workflow existants des partenaires. En effet, il est indispensable d'augmenter ces systèmes avec des mécanismes permettant d'envoyer et de recevoir les instances et les données d'état de workflows entre les différents partenaires.

7.6) Workflow interorganisationnel faiblement couplé

Dans cette forme d'interopérabilité [39] ; le workflow est divisé en sous workflows disjoints. Typiquement, il existe *n* partenaires impliqués dans un seul workflow global chacun disposant de son propre workflow local privé. Les workflows doivent être augmentés d'une structure d'interactions afin de permettre la communication entre les différents partenaires et l'exécution correcte des instances. L'interaction est réalisée grâce à une communication asynchrone et se base sur la circulation de messages entre les workflows locaux des partenaires.

Dans cette forme d'interopérabilité, les partenaires commencent par se fixer un protocole d'interaction grâce à un diagramme de séquence spécifiant les différents messages circulant entre les partenaires et ordonnant les événements qui leurs sont associés. Ensuite, chaque partenaire conçoit un workflow local privé conforme avec le

protocole. Enfin, le workflow faiblement couplé est exécuté par les différents participants. Nous signalons ici que pour communiquer entre eux, les partenaires doivent disposer de gestionnaires de messages permettant d'interpréter les messages reçus et de les acheminer vers les activités appropriées au sein des systèmes de gestion de workflow.

La forme d'interopérabilité faiblement couplée n'est pas dynamique. En effet, les différents partenaires interagissent et communiquent via l'échange de messages entre leurs workflows privés et ce conformément à un diagramme de séquence sur lequel ils se sont mis d'accord à priori.

Par conséquent, tout changement de partenariat risque de mettre en cause le diagramme de séquence et amener les entreprises à se fixer un nouveau protocole de communication. Ceci peut prendre un temps important et demander un grand effort de coordination.

En outre, cette forme d'interopérabilité ne préserve pas les workflow préétablis. En effet, la conception du workflow local d'un partenaire dépend fortement du protocole de communication. Ainsi, en changeant de partenariat, un partenaire peut être amené à entrer des modifications dans son workflow afin d'être conforme avec le protocole de communication.

Enfin, le workflow interentreprises faiblement couplé n'intègre pas les systèmes de gestion de workflows existants. En effet, les partenaires impliqués sont amenés à étendre leurs systèmes de gestion de workflow avec des mécanismes permettant d'interpréter les messages reçus et d'exécuter les opérations correspondantes. Ces mécanismes risquent de changer d'une coopération à l'autre en fonction des protocoles de communication adoptés et des messages échangés.

7.7) Public-à-Privé

L'approche public à Privé [40] est basée sur la notion de l'héritage et consiste en trois phases : les entreprises impliquées dans la coopération se mettent d'accord sur un workflow public commun servant de contrat entre elles. Ensuite, chaque activité dans le workflow public est associée à un domaine qui sera responsable d'une partie du workflow public. Enfin, chaque partenaire crée un workflow privé sous classe de la partie du workflow public correspondant. Cette forme d'interopérabilité est basée sur trois phases qui sont :

Phase 1 : conception du workflow public

La première étape consiste à concevoir un workflow public commun qui peut être vu comme un contrat entre les entités organisationnelles participantes. Ce workflow ne montre nécessairement pas la façon dont les activités sont exécutées mais montre seulement les activités intéressant toutes les parties.

Phase 2 : partitionnement du workflow public

La seconde phase de l'approche Public à Privé consiste à partitionner le workflow public selon les partenaires et relier les parties publiques entre elles pour former un workflow interentreprises.

Celui-ci est formé d'un ensemble de workflows, de canaux pour acheminer les informations, des méthodes représentant la fonction invoquée lors de l'exécution de l'activité ainsi que des relations de flots de canaux spécifiant la relation entre les participants.

Phase 3 : conception de workflows privés

Après avoir partitionné le workflow public, chaque partenaire peut réaliser la partie publique du workflow interentreprises correspondant comme il veut à condition que le workflow privé constitue une sous classe de sa partie publique. La relation de sous classe est basée sur la notion d'héritage.

Cette forme ne préserve pas les Workflows préétablis des partenaires puisque les partenaires doivent déterminer l'ensemble des règles à appliquer, ainsi que leur nombre et ordre, afin de correspondre leurs Workflows à la partie publique résultant du partitionnement du Workflow global.

8. Approches d'interopérabilité de workflows inter-organisationnels

8.1) Cross Flow

L'approche CrossFlow [41] est basée sur deux concept externalisation de services et l'établissement des contrats afin de décrire les relations entre les différentes organisations et permet ainsi la définition et l'exécution des processus inter-organisationnels. L'externalisation de service est marquée par trois phases qui sont : établissement de contrat, installation de l'infrastructure et exécution de service.

La phase de l'établissement du contrat permet de déterminer les services à externalisés selon la démarche suivante :

- Les fournisseurs de services publient des modèles contenant les détails sur les services au sein de moteur de correspondance de services.
- Le client externalisant un service contacte le moteur de correspondance de service en lui fournissant les modèles de leurs services afin de récupérer la liste des modèles de services des fournisseurs qui correspond à sa requête. A la réception de la liste, le client choisi les fournisseurs avec lesquels établi les contrats.

La seconde phase consiste à construire une infrastructure pour l'exécution de services en se basant sur les détails extraits du contrat précédemment établi. Cette phase consiste en effet à sélectionner les modules adéquats et de les paramétrer afin d'obtenir le comportement recherché.

La dernière phase consiste alors à exécuter le service externalisé. Une fois l'exécution du service est terminée, l'infrastructure dynamiquement construite est abandonnée.

8.2) WISE (Workflow based Internet SErvice)

L'architecture WISE [41] est formée de quatre composants, qui sont :

- Définition de workflows, permet de définir les workflows en utilisant comme bloc de construction les entrées d'un catalogue où les organisations peuvent publier leurs services.
- Exécution et contrôle ; permet de compiler la description du workflow en une représentation appropriée à l'exécution et de contrôler l'exécution du workflow en invoquant les services correspondants.
- Surveillance et analyse ; permettant de garder la trace de l'évolution de l'exécution du workflow ainsi que des états de tous les composants actifs dans le système.
- Coordination et communication ; permet de véhiculer les informations appropriées entre tous les participants en utilisant les informations produites par les workflows comme la principale source de routage.

La plateforme WISE ne préserve pas l'autonomie des partenaires puisque tous les partenaires doivent se conformer à des interfaces bien définies.

8.3) CrossWork

L'approche CrossWork [43] est basée sur la technologie des agents afin de déterminer les partenaires et former le workflow coordonnant leurs activités respectives. Les agents sont utilisés pour représenter les membres d'un groupe de travail lors de la formation du groupe et du workflow coordonnant leurs activités. L'approche utilise la plateforme de modélisation XRL basée sur les réseaux de Pétri. Cette approche est basée sur la notion de boite noire pour représenter les workflows des organisations participantes, ce qui n'est pas adéquat dans le contexte de coopération où on a besoin d'un certain degré d'inter-visibilité des workflows des partenaires.

8.4) ebXML (Electronic Business using eXensible Markup Language)

L'approche ebXML [44] à pour objectif de fournir un cadre sémantique assurant l'interopérabilité des données inter-organisationnelles et de construire des partenariats pour la conduite des projets commun, elle est basée sur l'échange de messages XML, l'intégration des workflows à l'aide d'ebXML doit se faire en trois phases :

- *la phase d'implémentation de spécification ebXML :* le référentiel ebXML contient des spécifications des workflows des organisations et des scénarios qui sont souvent utilisés dans des transactions commerciales. Une organisation commence par demander des spécifications ebXML, puis elle décider d'implémenté une ou plusieurs spécification selon ses besoins, après elle doit de publier son profil de collaboration dans le référentiel ebXML pour être visible aux autres organisations. Ce profil permettra aux organisations d'apprendre les capacités de l'organisation qui l'a publié. A tout moment toute organisation peut mettre à jour son profil dans le référentiel.
- *La phase de recherche et de négociation des informations ebXML des partenaires :* pour qu'une organisation (A) puisse coopérer avec une organisation (B), l'organisation (A) récupère le profil de l'organisation (B) du référentiel ebXML, puis étudie les capacités de l'organisation (B). les organisations négocient les termes de contrats avant de commencer la coopération. En effet, l'organisation (A) envoie à l'organisation (B) un contrat de partenariat ebXML appelé CPA (Collaborative Partner Agreement). Finalement (A) et (B) se mettent d'accord sur un contrat de partenariat commun.
- *La phase d'exécution des transactions ebXML :* une fois que le contrat de partenariat a été négocié, les transactions peuvent commencer selon une manière prédéfinie où chaque organisation joue un rôle prédéterminé. Les transactions consistent en l'envoi de messages ebXML.

L'ebXML ne préserve pas le savoir-faire des partenaires, puisque les organisations doivent publier leurs workflows. Ainsi, elle ne préserve pas les systèmes de gestion de workflows existants parce que tous les participants doivent se conformer à une infrastructure commune d'interopérabilité commerciale définie pour adapter et ajuster leurs interfaces ce qui réduit la flexibilité.

8.5) CoopFlow

L'approche CoopFlow [25] s'inspire de l'architecture orientée service qui est basée sur trois opérations : la publication, la recherche et la connexion. Les fournisseurs de services publient les différents services qu'ils offrent, les demandeurs de services cherchent les différents services répondant à leurs besoins, une fois que les services sont trouvés les demandeurs s'y connectent afin d'exécuter un ensemble d'opérations offertes par les services. Cette approche est considéré flexible et préserve l'autonomie et le savoir-faire des organisations.

9. Langages de spécification de workflow

Beaucoup de langages de spécification de processus métier sont apparues, chacun adopte sa terminologie particulière et définit ces concepts spécifiques en fonction des objectifs visés par le méta-modèle de workflow, du domaine d'intérêt ou du choix de représentation.

9.1) XPDL (XML Process Definition language)

XPDL [45, 46] est un langage proposé par la WfMC qui permet de définir des processus workflows et utilise le langage XML comme support, un processus workflow est représenté par (*Workflow process*), il est divisé en plusieurs activités qui sont (*Workflow process Activity*) qui peuvent être atomiques (tâches élémentaires) ou être des appels vers des sous-processus. Ces activités sont ordonnées grâce aux informations de transition *(Transition Information)* qui peuvent porter sur les données pertinentes pour le workflow (*Workflow Relevant Data*). Les activités sont réalisées par des ressources humaines ou des entités organisationnelles définies par l'intermédiaire des participants (*Workflow Application Specification*). Elles peuvent invoquer des applications externes via des déclarations d'applications (*Workflow Application Declaration*).

9.1) WSFL (Web Service Flow Language)

WSFL [45] est un langage proposé par IBM, destiné à la composition des services web, il offre deux types de composition :

- Modèle de flux : ce modèle correspond au processus, c'est-à-dire, à la composition explicite de la succession des étapes et de l'enchainement des appels aux opérations des services web. Il représente le processus métier proprement dit.
- Modèle global : ce modèle correspond à un modèle d'interactions de services web deux à deux. Il décrit une collection de liens entre les opérations duales de services web.

9.2) XLANG

XLANG [45] est un langage proposé par Microsoft. Il est utilisé pour l'orchestration des activités qui constituent un processus métier, il utilisé les concepts suivants pour décrire un processus :

- *All* : indique une exécution parallèle des sous-processus.
- *Compensate* : invoque une compensation d'une transaction.
- *Context* : un contexte forme un cadre pour les déclarations locales, la gestion des transactions et la gestion des exceptions
- *Empty* : un processus vide.
- *Pick* : attend l'arrivée d'événement pour déclencher des processus. L'événement peut être la levée d'un signal ou la fin d'une action.
- *Sequence* : une séquence de processus et actions.
- *Switch* : définit des branches conditionnelles, chacune reliée à un processus.
- *While*: définit une boucle sur un processus.

9.3) WSCL (Web Service Conversational Language)

WSCL [47] est proposé pour décrire à l'aide des documents XML les services web en se basant sur leurs conversations. Les éléments d'une spécification WSCL sont les suivant :

- Description des types de documents : elle référence les types des documents qui peuvent être échangés ou reçus. Les documents sont référencés par leurs URI respectifs.
- Interactions : elles modélisent les actions de conversations sous la forme d'échange de documents. Il existe cinq types d'interactions :

a) *Send* : le service envoie un document.
b) *Receive* : le service reçoit un document.
c) *SendReceive* : le service envoie un document et en attend un au retour.
d) *ReceiveSend* : le service reçoit un document et en renvoie un au retour.

e) *Empty* : aucun document n'est échangé, le rôle de ce type d'interaction est de modéliser le début et la fin d'une conversation.

- Transitions : elles spécifient la relation d'ordre entre les opérations. Pour cela, une transition possède une interaction source et une interaction de destination. Il est également possible de spécifier un type de document pour la transition source, ainsi qu'une condition sur la transition.
- Conversations : elles listent toutes les interactions et les transitions qui forment une conversation. Elles définissent également les propriétés supplémentaires telles que le nom de la conversation et les interactions de début et fin.

9.4) WSCI (Web Service Choreography Interface)

WSCI [48] est le langage de description d'interface qui décrit le flux de messages échangés par un service web qui interagit d'une manière ordonnée avec d'autres services. Il permet aussi, de décrire le comportement du service et d'exprimer les dépendances logiques et temporelles entre les messages échangés à l'aide de contrôle de séquences, corrélation, gestion de fautes et transactions.

WSCI définit les concepts suivants :

- Les interfaces : elles décrivent les scénarios offerts aux éléments. Il est ainsi possible d'offrir un scénario différent en fonction du client qui interroge les services. Une interface propose donc, le comportement externe observable du service considéré.
- Les activités et leurs chorégraphie : il existe deux types d'activités ; atomiques et complexes. Une activité complexe décrit la chorégraphie des activités atomiques qui la composent. Les chorégraphies supportées sont les exécutions séquentielles et parallèles, les boucles et les branchements conditionnels.
- Les processus : il s'agit de l'unité de réutilisation WSCI. Un processus peut être instancie par la réception d'un ou plusieurs messages.
- Les propriétés : elles sont employées pour référencer une définition de message.
- Les contextes : ils décrivent les environnements dans lesquels sont exécutes les activités. Un contexte contient :

a) Les déclarations disponibles pour les activités.
b) Les exceptions qui peuvent surgir à l'exécution des activités.
c) Les propriétés transactionnelles des activités.

- Les corrélations de messages : WSCI définit les conversations comme des échanges de messages. La corrélation de messages vise à associer à chaque message reçu la conversation qui lui est destinée. Pour cela, chaque message échangé fait référence à une instance de corrélation qui permet de le replacer dans sa conversation.

- Comportements exceptionnels : lorsqu'une exception survient, WSCI permet d'associer un ensemble d'actions à exécuter en réponse. Cette gestion des exceptions est systématiquement placée dans un point précis de la conversation.
- Les comportements transactionnels : un contexte peut être associé à une transaction. Une transaction est soit atomique ou composé d'autres transactions.
- Le modèle global : il décrit par la collection d'interfaces de services participants ainsi que par les liens statiques effectuées à l'aide de WSDL.

9.5) BPEL4WS (Business Process Execution Language For Web Services)

BPEL [49, 51] définit deux types de définition de processus : *Abstract* et *Process*. Un Abstract est un processus partiel et non exécutable. Un *Process* définit un processus exécutable, il peut implémenter plusieurs *Abstracts*.

Une définition de processus peut définir des messages (*Message*), des participants (*Participant*) et des ensembles de règles (*RuleSet*). Les messages définissent les informations qui transitent entre activités. Il peut être de type *data*, *request* ou *response*. Les participants recouvrent l'ensemble des entités qui interagissent avec le processus (Web Service, application, rôle organisationnel, etc.).

Enfin, un processus définit une activité de plus haut niveau. Cette activité est soit une activité simple (*SimpleActivity*) soit une activité complexe (*ComplexActivity*). Elle peut être des contraintes de temps concernant son démarrage (*schudle*) ou sa fin (*CompleteBy*). Une activité complexe peut définir une transaction (*Transaction*). Dans le cas où une transaction est abandonnée, des activités de compensation (*Compensate*) peuvent être lancées pour chacune des sous-activités de l'activité complexe.

Les activités complexes peuvent générer des exceptions (*Exception*). Une exception peut être adressée à un participant. Une activité peut être désignée pour traiter cette exception quand elle survient au cours de l'exécution d'une activité complexe (*OnException.target*).

BPEL définit trois types d'activités : complexes, simples et procédurales. Les activités complexes sont les suivantes :

- *All* : une activité qui est terminée lorsque toutes ses activités sont terminées. Ces activités sont effectuées en parallèles.
- *Choice* : une activité qui est terminée dés que l'une de ses sous-activités est terminée
- *Foreach* : une activité qui répète une séquence d'activités sur un ensemble de valeur.

- *Sequence* : une activité qui est terminée toutes ses sous-activités sont terminées. Ces sous-activités sont effectuées en séquence.
- *Switch* : une activité qui est terminée lorsque toutes ses sous-activités candidates à l'exécution sont terminées, si aucune activité n'est candidate, et si un lien '*otherwise*' est définit, l'activité reliée par le lien est exécutée.

Les activités simples identifiées sont les suivantes :

- *Empty* : ne fait rien.
- *ExceptionActivity* : génère une exception.
- *Consume* : attend et consomme un message d'un participant.
- *Operation* : invocation d'une opération sur un participant (*InvokingOperation*), c'est-à-dire, envoi du message sortant au participant et attente du message entrant, ou implémentation d'une opération (*ImplementingOperation*), c'est-à-dire, consommation du message entrant et production du message sortant.
- *Produce* : produit un message et le délivre à un participant.

Enfin, le dernier type d'activité concernant les activités procédurales qui sont :

- *Assign* : exécute une affectation.
- *Complete* : termine le processus.
- *Join* : attend la fin de l'exécution d'un sous-processus.
- *Release* : libère une donnée de la valeur qui lui a été affectée.
- *Repeat* : répète l'activité parente.
- *Spawn* : démarre un sous-processus.

L'élément atomique d'un processus BPLEL4WS est activité, qui peut être l'envoi d'un message, la réception d'un message, l'appel d'une opération (envoi d'un message, attente d'une réponse), ou une transformation de données. BPEL offre les fonctionnalités suivantes :

- Organisation des activités : des activités séquentielles, parallèles, conditionnel, etc.
- Gestion des transactions : il existe deux modes de transactions ; les transactions courtes et les transactions longues.
- Organisation du processus en sous processus ;
- Gestion des erreurs : il est possible de définir des gestionnaires d'erreurs pour détecter les erreurs, les traites ou les ignorées.

9.6) YAWL (Yet Another Workflow Language)

YAWL [52] est un langage de workflow basé sur les patrons de workflow [53]. Il utilise les réseaux de Pétri comme un point de départ. La spécification de workflow dans YAWL est un graphe orienté appelé WF-net. Dans ce graphe il ya des tâches atomiques et des tâches composées. Chaque tâche composée rapporte à un WF-net qui contient son expansion, une tâche atomique correspond à une action atomique, c'est-à-dire, une action qui exécuté par un utilisateur ou par une application logiciel [50].

10. Patrons de contrôle de flux workflow

Un aspect majeur pour modéliser un processus de workflow est son contrôle de flux. Il doit être clair dans quel ordre les activités sont exécuté et quand le fil de la commande est passé d'une activité à l'autre. Ces patrons sont divisés en six groupes [54] , qui sont :

10.1) Patrons de contrôle de flux de base (Basic Control Flow Patterns)

Ce sont 5 patrons qui capturent des enchaînements élémentaires d'activités dans un procédé :

- **Séquence**

Ce patron représente un enchaînement de deux activités dans un procédé où la deuxième activité est exécutée après l'achèvement de la première.

- **Parallélisme (AND-Split)**

Le parallélisme représente un point dans le procédé où un lien de contrôle simple est divisé en plusieurs liens de contrôle s'exécutant en parallèle.

- **Synchronisation (AND-Join)**

La synchronisation est un point dans le procédé où plusieurs flux de contrôle convergent et fusionnent dans un seul flux de contrôle synchronisant tous ces liens.

- **Choix Exclusif (XOR-Split)**

Ce patron représente un point dans un procédé où, basé sur une décision ou des paramètres, une activité parmi plusieurs est choisie pour être exécutée. Contrairement à Parallélisme, un seul lien dans le flux de contrôle est activé.

- **Fusion simple (XOR-Join)**

Ce patron représente un point dans un procédé où deux ou plusieurs branches alternatives se réunissent sans synchronisation. Ce patron suppose qu'une seule branche alternative est choisie pour être exécutée.

10.2) Patrons d'enchaînement avancés et de synchronisation (Advanced Branching and Synchronization Patterns)

Ces 4 patrons capturent des enchaînements plus complexes et la synchronisation des activités dans un procédé :

- **Choix Multiple (Multiple Choice)**

Ce patron représente un point dans un procédé où, basé sur une décision ou des paramètres, un certain nombre d'activités sont choisies. Contrairement au patron Choix Exclusif, dans le patron Choix Multiple plusieurs branches peuvent être sélectionnées et non pas qu'une seule.

- **Fusion Synchronisée (Synchronizing Merge)**

La fusion synchronisée est un point dans le procédé où plusieurs flux de contrôle convergent vers un seul. Si une seule branche du flux de contrôle est exécutée, l'activité suivante à la fusion synchronisée est activée au moment où l'exécution de cette branche se termine. Sinon, les branches qui se sont exécutées sont synchronisées avant de continuer.

- **Multi Fusion (Multi Merge)**

Ce patron représente une situation dans un procédé où plusieurs branches convergent vers un seul point mais sans synchronisation. Si plus d'une activité sont choisies, probablement concurremment, l'activité suivant la fusion est commencée pour chaque activation de chaque branche entrante.

- **Discriminateur (Discriminator)**

Ce patron est un point dans un procédé qui attend l'achèvement d'une des branches entrantes pour exécuter l'activité suivante. A ce moment là, il attend l'achèvement de toutes les branches restantes et puis les ignore. Si toutes les branches entrantes ont été déclenchées, il peut être déclenché à nouveau.

10.3) Patrons impliquant des instances multiples (Patterns involving Multiple Instances)

Ces patrons décrivent des situations où plusieurs flux d'exécution accèdent à une donnée partagée. Cela signifie qu'une activité dans un processus a plusieurs instances actives en même temps.

- **Multi-Instances avec la connaissance a priori au temps de conception (MI with a priori Design Time Knowledge)**

Ce patron représente une situation où plusieurs instances d'une activité sont produites dans une instance de procédé. Le nombre d'instances de cette activité pour une instance du procédé est connu à la conception. Il faut synchroniser l'exécution des instances avant qu'une autre activité doive être commencée.

- **MI avec la connaissance a priori au temps d'exécution (MI with a priori Runtime Knowledge)**

Ce patron représente une situation où plusieurs instances d'une activité sont produites, le nombre d'instances est connu au moment de l'exécution du procédé, mais avant le démarrage des instances.

- **MI sans la connaissance a priori au temps d'exécution (MI without a priori Runtime Knowledge**)

Ce patron représente une situation où plusieurs instances d'une activité sont produites, le nombre d'instances n'est pas connu à l'avance, cependant on peut créer des nouvelles instances à la demande.

- **MI sans synchronisation (Multiple Instances Without Synchronization)**

Ce patron représente un point dans un procédé où plusieurs instances d'une activité sont produites sans synchronisation. Chaque instance de cette activité est indépendante des autres instances.

10.4) Les patrons à base d'état (State-based Patterns)

Ces patrons représentent les enchaînements des activités qui basent sur les états d'exécution d'un procédé plutôt que les conditions explicites. Ils sont 3 patrons :

- **Choix Différé (Deferred Choice)**

Ce patron représente un point dans le procédé dans lequel une branche est choisie parmi plusieurs. Ce choix est basé sur une information ou une donnée qui n'est pas nécessairement disponibles au moment où ce point du procédé est atteint.

La sélection de la branche est alors retardée jusqu'à ce que certains événements arrivent pour donner au procédé l'information requise.

- **Étape importante (Milestone)**

Ce patron représente une situation où une activité devient disponible si et seulement si on a atteint une certaine étape importante (milestone).

- **Routage Parallèle Intercalé (Interleaved Parallel Routing)**

Ce patron représente un point dans un procédé où un ensemble d'activités sont exécutées dans un ordre arbitraire. L'ordre est décidé à l'exécution, et une seule activité est exécutée à un moment donné.

10.5) Patrons d'Annulation (Cancellation Patterns)

Ces 2 patrons représentent les constructions pour modéliser l'annulation d'une activité ou d'un procédé :

- **Activité d'Annulation (Cancel Activity)**

Ce patron représente un point dans un procédé où il annule l'exécution d'une activité.

- **Cas d'Annulation (Cancel Case)**

Ce patron représente une situation où un procédé est complètement annulé.

10.6) Patrons Structuraux (Structural Patterns)

Ces deux patrons concernent les contraintes structurelles imposées aux spécifications de procédés :

- **Cycles Arbitraires (Arbitrary Cycles)**

Ce patron représente un point dans un procédé où une ou plusieurs activités peuvent se répéter plusieurs fois.

- **Terminaison Implicite (Implicit Termination)**

Ce patron représente la situation où un procédé se termine quand il n'y a plus rien à exécuter.

11. Modélisation et vérification du workflow interorganisationnel lâche

La modélisation et la vérification des processus workflows sont deux éléments essentiels dans la conception des SGW, la première elle donne les éléments vue plus formel pour le processus à implémenter et la deuxième elle vérifiée la cohérence dans les processus pour éviter au maximum toutes les situations de deadlock (impasse) et overlook. Pour cela on va utiliser les réseaux de pétri et les diagrammes de séquence de messages.

12. Modélisation et vérification d'un processus workflow organisationnel

Un processus workflow est modélisé par l'utilisation de WF-Net qui un type particulier des réseaux de Pétri [55].

La définition suivante spécifiée les caractéristique de WF-Net.

12.1) Définition (WF-Net)

Un réseau de Pétri $PN = (P, T, F)$ est un workflow net (WF-Net) si seulement si :

- $P = \{p_1, p_2, \ldots, p_m\}$ est un ensemble finie de places.
- $T = \{t_1, t_2, \ldots, t_n\}$ est un ensemble finie de transitions.
- $P \cap T = \phi$ et $P \cup T \neq \phi$.
- $F\ (P \times T) \cup (T \times P)$ est un ensemble d'arcs.
- Le réseau *PN* possèdes deux places spéciales : *i* et *o*. *i* c'est la place source : $\bullet$ i =∅, *o* c'est la place puits : o$\bullet$=∅.
- Si on ajoute une transition t* au réseau qui connecte la place *o* à la place i (c.-à-d. $\bullet t^* = \{o\}$ et $t^* \bullet = \{i\}$) alors le réseau résultant est fortement connecté.

Les places modélisent les dépendances causales, c'est-à-dire, les prés et postes conditions pour les tâches. Les transitions modélisent les tâches.

12.2) Vérification des WF-Nets

Les deux exigences citées dans la définition précédente peuvent être vérifiées statiquement, c.-à-d. qu'elles sont reliées à la structure du réseau de Pétri. Cependant, il y aura une autre exigence qui doit être satisfaite :

D'ailleurs, il ne devrait y avoir aucune tâche morte, c.-à-d., il devrait être possible d'exécuter une tâche arbitraire en suivant le chemin approprié dans le WF-net. Ces deux contraintes additionnelles correspondent à la propriété de solidité (soundness).

12.3) Définition (sound)

Un processus workflow est modélisé par le PN = (P, T, F) est sound si et seulement si :

- Pour chaque état M accessible de l'état i, il existe un ordre de franchissement menant de l'état *M* à l'état *o*. formellement:

$$\forall M(i \overset{*}{\rightarrow} M) \Rightarrow (M \overset{*}{\rightarrow} o)$$

- L'état *o* est le seul état accessible de l'état *i* avec au moins un jeton en place *o*. Formellement:

$$\forall M(i \overset{*}{\rightarrow} M \wedge M \geq o) \Rightarrow (M = o)$$

- Il n'y a aucune transition morte dans *(PN, i)*, formellement :

$$\forall t \in T \exists M, M' \; i \overset{*}{\rightarrow} M \overset{t}{\rightarrow} M'$$

La première condition déclare qu'à partir de l'état initial (l'état *i*), il est toujours possible à arrivez à un état avec un seul jeton dans la place *o*, c.-à-d. ils ne devraient pas présentez une boucle infinie.

La deuxième condition déclare que le moment où un jeton est mise en place *o* tous les autres places devrait être vide.

La dernière condition déclare qu'il n'a y aucune transition morte (tâches) à partir de l'état initial *i*.

13. Modélisation du processus workflow interorganisationnel

Dans les deux sections précédentes, nous avons utilisés les réseaux de Pétri pour modéliser et analyser les workflows au sein d'une seule organisation. Maintenant il est temps de considérer des workflows inter-organisationnels.

Un *workflow interorganisationnel* est un ensemble de processus workflows faiblement couplés. Typiquement, il y a *n partenaires* qui sont impliqués dans un processus workflow global. Chacun des partenaires a son propre processus workflow local. Chaque processus workflow local est privé, c.-à-d. le partenaire possède un contrôle total sur son workflow local. Cependant, ces processus workflows locaux doivent communiquer parce qu'ils dépendent l'un de l'autre pour l'exécution correcte des cas. Le processus workflow

global se compose des processus workflows locaux et d'une structure d'interaction. Il y a de deux manières d'interaction entre les partenaires: asynchrone et synchrone. La manière asynchrone correspond à l'échange des messages entre les processus workflows locaux. L'interaction synchrone force les processus workflows locaux à s'exécuter des tâches spécifiques en même temps. Dans notre cas nous intéressons à l'interaction asynchrone puisque le système est faiblement couplé. La définition suivante formalise le concept d'un workflow interorganisationnel lâche (faiblement couplé) [58].

13.1) Définition (IOWF)

Un *workflow interorganisationnel lâche (WFIOL)* est un tuple $WFIOL= (PN_1, PN_2,..., PN_n, P_{CA}, AC)$, *où :*

- $n \in N$ *est le nombre de workflows locaux sous forme de WF-Net.*
- *Pour chaque* $k \in \{1,2,..., n\}$ *:* PN_k *est un WF-Net avec une place source* i_k *et une place puits* o_k*.*
- *Pout tous* $k, l \in \{1,2,..., n\}$ *: si* $k \neq l$*, alors* $(P_k \cup T_k) \cap (P_l \cup T_l) = \emptyset$*.*
- $T^{\square} = \cup_{k\in\{1,..., n\}} T_k$, $P^{\square} = \cup_{k\in\{1,..., n\}} P_k$, $F^{\square} = \cup_{k\in\{1,..., n\}} F_k$*.*
- P_{CA} *c'est l'ensemble des éléments de communication asynchrone (places de communication).*
- $P_{CA} \cap (P^{\square} \cup T^{\square}) = \emptyset$*.*
- $AC \subseteq P_{AC} \times P(T^{\square}) \times P(T^{\square})$ *c'est une relation de communication asynchrone. Tel que* $P(T^{\square})$ *c'est l'ensemble de tous les sous-ensembles non vide de* $T^{\square}$. cette relation *CA* spécifier un ensemble de transitions d'entrée et un ensemble de transitions de sortie pour chaque élément de communication asynchrone.
- *Pour toutes* $p \in P_{AC}$, $\{(p', x, y) \in AC \mid p' = p\}$ *est unique (singleton).*

14. Vérification du processus workflow inter-organisationnel lâche

Un workflow inter-organisationnel lâche qui se compose d'un certain nombre de workflows locaux peut être vérifiée la propriété de solidité (soundness) au niveau local, c-à-dire, au niveau des WF-Nets locaux, mais cette propriété n'est pas vérifiée au niveau global et vice-versa.

14.1) Définition (solidité interorganisationnelle)

Un workflow interorganisationnel (WFIO) est sain si et seulement si sain localement et globalement [58]. WFIO est localement sain si et seulement si chacun de ces workflows PN_k est sein. WFIO est globalement sain si et seulement si le U(IOWF) est sein.

14.2) Définition U(IOWF)

Soit $WFIO = (PN_1, PN_2, ..., PN_n, P_{AC}, AC)$ un workflow interorganisationnel. $U(IOWF) = (P^U, T^U, F^U)$ est le déploiement de IOWF, qui définie comme suit :

- $P^U = P^{\square} \cup P_{AC} \cup \{i, o\}$.
- $T^U = T^2 \cup \{t_i, t_o\}$.
- $\{i, o, t_i, t_o\} \cap (P^2 \cup T^{\square} \cup P_{AC}) = \varnothing$.
- $F^U = F^{\square} \cup \{(t, p) \in T^{\square} \times P_{AC} \mid (p, x, y) \in AC \wedge t \in x\} \cup \{(p, t) \in P_{AC} \times T^{\square} \mid (p, x, y) \in AC \wedge t \in y\} \cup \{(i, t_i), (t_o, o)\} \cup \{(t_i, i_k) \mid k \in \{1, ..., n\}\} \cup \{(o_k, t_o) \mid k \in \{1, ..., n\}\}$.

Dans le réseau de déploiement tous les WF-nets locaux sont reliés entre eux par une transition *de début* t_i et une transition *d'arrêt* t_o. D'ailleurs, une place de source globale *i* et une place de puits globale *o* ont été ajoutées. Les éléments de communication asynchrone (P_{AC}) sont figurés sur les places ordinaires. Le résultat du déploiement est un nouveau WF-net.

La solidité interorganisationnel (*IO-sound)* d'un workflow interorganisationnel $IOWF = (PN_1, PN_2, ..., PN_n, P_{AC}, AC)$ correspond à la solidité de $n+1$ WF-nets:$PN_1, ..., PN_n$ et *U(IOWF).*

15. Conclusion

Dans ce chapitre nous avons exposé les concepts et les outils essentiels relatifs aux workflows pour nous permettre de bien comprendre notre domaine d'application.

Il est aussi nécessaire de prendre en considération certains aspects liés à l'organisation et ses objectifs métiers, son niveau technique et sémantique en vue d'aboutir à un modèle d'architecture permettant à l'organisation d'atteindre ses objectifs et d'interagir avec son environnement sans affecter son savoir faire ou son autonomie.

Chapitre III :

Une architecture à base des agents pour le workflow interorganisationnel lâche

1.Introduction :

Le workflow interorganisationnel a plusieurs formes d'interopérabilité telles que : partage de capacité, exécution chainée, sous-traitance, transfert de cas, transfert de cas étendu, public-à-privé et faiblement couplé (lâche) [25, 39]. Notre contribution est orientée vers le workflow inter-organisationnel lâche (WIOL). Cette forme d'intropérabilité est caracterisée par la présence de *n* partenaires, chacun d'eux dispose d'un workflow local privé et une structure d'interaction qui lui permet d'interagir avec les autres partenaires en utilisant des messages asynchrones.

Dans cette forme d'intropérabilité il est nécessaire de fixer un protocole d'interaction pour spécifier les différents messages circulants entre les partenaires suivant un diagramme de séquences de messages.

Chaque partenaire doit concevoir son workflow privé local d'une façon conforme au protocole d'interaction.

Certains travaux ont été effectués dans cette forme d'interopérabilité, nous citons :

Dans [59] les auteurs ont proposé une plateforme d'exécution du workflow interorganisationnel lâche à base d'agents et des règles ECA (Event, Condition and Action) ; dans cette plateforme, le workflow inter- organisationnel est modélisé par un système multiagent où chaque organisation est représentée par un agent. L'exécution du processus global est divisée en deux parties : l'intra-exécution, qui concerne l'exécution au sein de l'organisation, et l'inter-exécution, qui représente l'interaction entre les organisations.

Dans l'intra-exécution, les règles ECA jouent le rôle d'un moteur de workflow au sein de l'organisation, elles sont utilisées pour contrôler l'état interne des transitions. L'agent est utilisé pour assurer l'interaction avec les agents des autres organisations. Dans ce travail la découverte de partenaires et la négociation ne sont pas abordées ce qui rend le système rigide.

Dans [60] les auteurs ont proposé un modèle à base d'agents et des ontologies pour la coordination dynamique entre les processus workflow interorganisationnels, dans ce modèle, le service pour la disposition du workflow est représenté par trois types d'agents ; l'agent moteur qui joue le rôle du moteur workflow, l'agent de connexion qui est chargé de la connexion à l'infrastructure de médiation pour trouver les partenaires et l'agent superviseur qui est responsable de la répartition des tâches sur les agents moteurs et le

contrôle des activités dans le service pour la disposition de workflow. Les auteurs ont aussi proposé deux ontologies :

- L'ontologie de domaine qui a pour rôle de décrire touts les concepts du domaine et les relations entre ces concepts,
- l'ontologie d'interaction qui décrit les protocoles d'interaction utilisés dans le domaine. Mais le point faible dans ce travail est que le l'aspect intra-organisationnel est absent au niveau de la description des services.

Dans [39] l'auteur propose un prototype de modélisation et d'analyse des workflows interorganisationnels lâches à base des diagrammes de séquences de messages (DSM) et des réseaux de pétri (RP) ; Les DSM sont utilisés pour la description des interactions inter organisationnelles et les RP sont utilisés pour modéliser un processus workflow et analyser les propriétés des processus pour la détection des situations d'inter blocage.

Dans ce qui suit, nous allons aborder le modèle proposé.

2.L'architecture proposée

Notre contribution consiste à proposer un modèle contenant des solutions aux quatre problèmes rencontrés dans le WIOL, qui sont :

- La description des services workflow
- La découverte des partenaires,
- La négociation pour l'établissement du contrat,
- Finalement l'exécution du service et la surveillance de l'exécution.

2.1)Description du service workflow

Un workflow interorganisationnel lâche est vu comme un ensemble de services répartis, autonomes. Chaque service est décrit par une agrégation des classes complémentaires ; chacune est utilisée pour assurer une ou plusieurs tâches soit dans le workflow local ou dans le workflow global.

Dans ce qui suit un service workflow est décrit par les classes suivantes :

- **ServiceProfil** : Cette classe est utilisée pour créer une abstraction du service workflow, c'est-à-dire fournir une description du service et cacher sa partie métier.

Elle est définie par le tuple ServiceProfil= (AID, ServiceName, ServiceType, ServiceOwner, Inputs, Outputs). Où : AID représente l'identifiant de l'agent qui représente l'organisation dans l'environnement, ServiceName est le nom attribué au service. ServiceType est le type de service puisque on peut trouver plusieurs services ayant le même nom et différents dans leurs natures. ServiceOwner représente le profile d'organisation tel que le nom de l'organisation, l'adresse,…etc. Inputs est l'ensemble des paramètres d'entrée pour le démarrage de l'exécution du service. Outputs est l'ensemble des paramètres retournés après l'exécution de service (résultats). Chaque Input et Output est identifié par un nom et un type. Cette abstraction a pour rôle de décrire une offre ou un besoin d'une façon non ambiguë.

- **ServiceWorkflow**: Cette classe représente la partie métier du service à offrir. Elle décrit le processus d'exécution du service ; un processus est composé d'un évènement de début suivi par un ensemble d'activités, on distingue deux types d'activités

(i) Les activités métiers qui sont utilisées pour l'exécution des tâches métiers.
(ii) Les activités de routage appelées aussi patrons de routages constitués d'une séquence, de boucles, AND-Split, AND-Join, XOR-Split, XOR-Join. Ces patrons sont utilisés pour interconnecter les activités métiers. finalement le processus métier termine par un évènement de fin.

- **SerivceContract** : Cette classe représente la chorégraphie du service. Elle décrit l'ordre d'échange de messages entre les différents participants ainsi que la description des actions à exécuter par chaque participant, d'une façon formelle un contrat est une machine d'états fini déterministe $FSM = <N, A>$; où, N représente l'ensemble fini des nœuds et A représente l'ensemble des arcs qui relient les nœuds. Un nœud représente une clause dans le contrat, elle est décrite par une règle déontique de la forme suivante $r_i:\phi \rightarrow \theta_{s,b}(\alpha<\psi)$, où :

  - r_i représente la désignation de la règle,
  - ϕ représente la condition d'activation de la règle (pré-condition),
  - θ est un opérateur déontique qui prend une des trois valeurs suivantes Obligation (O), Permission (P) ou Prohibition (P).
  - $_s$ est le rôle qui assure la réalisation de la clause,
  - $_b$ est le bénéficiaire de la réalisation de la clause,
  - α est l'action à exécuter,

- ψ représente la post-condition qui indique si la clause est accomplie ou non.

Les arcs représentants les évènements échangés entre les rôles. Un arc est libellé par un message FIPA-ACL.

2.2) Macro Architecture

Notre architecture hérite de l'architecture orienté service SOA [13] qui est caractérisée par :

– **Acteurs** sont des agents logiciels qui représentants les organisations.
– **Rôles** : un rôle est identifié par un ensemble de capacités des acteurs qui jouent le rôle. On distingue trois macros rôles qui sont :

a. ***Facilitateur*** ; c'est un rôle joué par un acteur pour :

- Permettre aux agents ayant des services workflow à offrir dans de les publier, les mettre à jour ou d'annuler la publication d'un service s'il n'est plus disponible.
- Permettre aux agents qui cherchent des services de trouver ceux qui répondent aux critères décrits dans leurs requêtes.

b. ***Fournisseur*** ; ce rôle est joué par un agent possédant des services à fournir. Pour que les services soient visibles par les agents de l'environnement externe, le fournisseur doit les publier chez le facilitateur le plus convenable.

c. ***Demandeur*** ; c'est un rôle joué par un agent qui cherche un service en dehors des frontières de l'organisation qu'il représente. La recherche des services est effectuée par l'interrogation d'un facilitateur directement via une requête décrivant le service demandé.

– Interaction inter-rôle

Dans cette section on va identifier les différentes interactions existantes et leurs sémantiques, une interaction est dirigée par un protocole d'interaction, et attendu que notre architecture est orientée agent alors les protocoles qu'on va utiliser sont les protocoles définit par FIPA.

a. Interaction fournisseur-facilitateur (publication de service)

Cette interaction ait lieu quand : un agent veut publier le profil du service qu'il peut offrir, mettre à jour un profil déjà publié ou bien annuler une publication existante. Dans les trois situations suscitées ; le fournisseur qui est l'initiateur de l'interaction sollicite le facilitateur qui est le contractant pour exécuter une action de publication, de mise à jour ou de suppression d'un profil de service. Donc le fournisseur commence l'interaction par

l'envoi d'un message FIPA-ACL avec un performatif *request* (ce performatif ayant la sémantique d'exécuter une action) et un contenu décrivant l'action demandée (*publish(ServiceProfil), update(ServiceProfil)* ou *delete(ServiceProfil)*) et finit par une réponse de la part du facilitateur sous forme aussi d'un message FIPA-ACL avec un performatif *agree* si l'action demander est exécutée avec succès, ou avec un performatif *failure* si une exception apparait pendant l'exécution de l'action et le facilitateur n'est pas capable de suivre l'exécution de l'action, ou bien un performatif *not-understood* si l'action demander par le fournisseur n'est pas comprise par le facilitateur. Ce mode d'interaction est implémenté par le protocole FIPA-Request [62].

La figure ci-après schématise le diagramme d'interaction fournisseur - facilitateur.

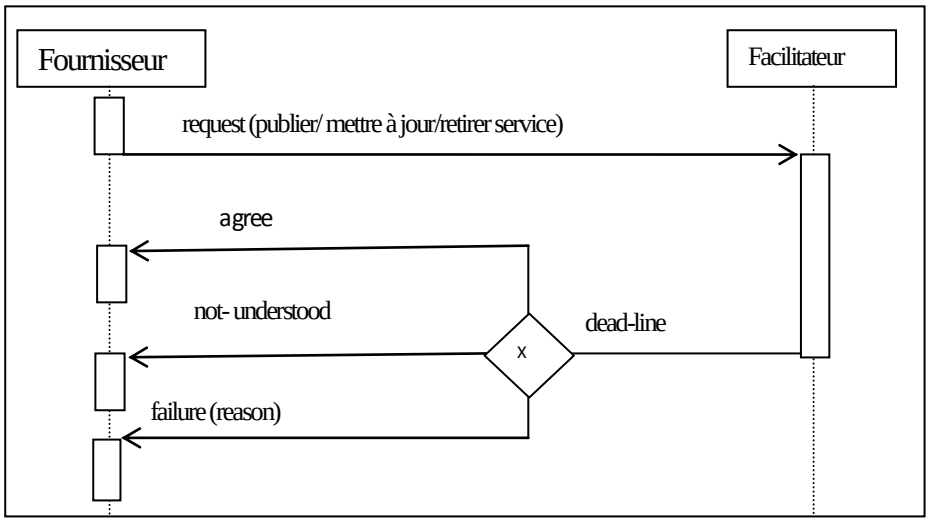

Figure III. 1: Diagramme d'interaction fournisseur - facilitateur

b. Interaction demandeur-facilitateur (recherche des partenaires)

Elle a pour objectif de fournir au demandeur une liste d'éléments selon le profil donné dans sa requête, chaque élément de la liste est une instance de la classe *ServiceProfil*.

Cette interaction commence par l'envoie d'un message FIPA-ACL avec un performatif *query* (ce performatif ayant la sémantique informatif) de la part du demandeur (initiateur) au facilitateur (contractant) contenant un profil du service cherché (souhaité) et finit par l'envoi d'une réponse contenant la liste des éléments trouvés par le facilitateur via un message FIPA-ACL avec le performatif *inform* ou via un message avec le performatif *not-understood* si le facilitateur n'a pas compris le contenu de la requête envoyée par le demandeur, en d'autres termes, le facilitateur exécute dans cette interaction un acte informatif. L'implémentation de ce type d'interaction exige l'utilisation du protocole FIPA-Query [63]. Ce type d'interaction est schématisé par le diagramme suivant :

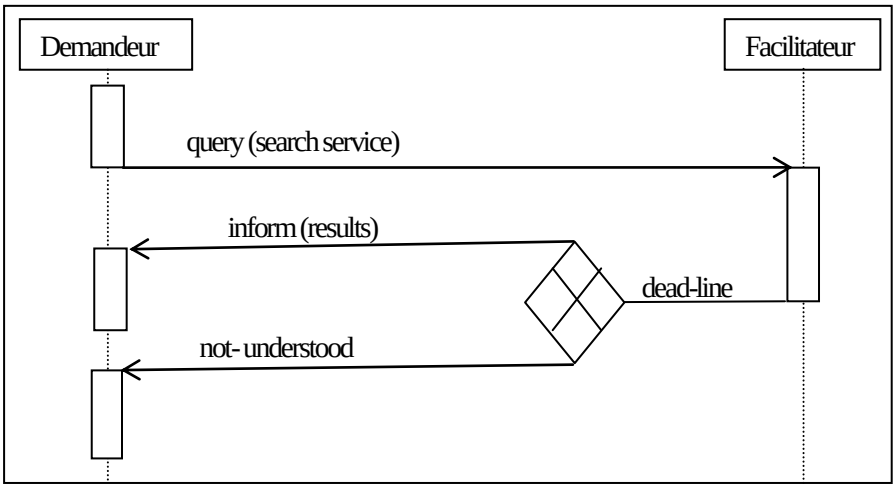

Figure III.2: Diagramme d'interaction demandeur - facilitateur

c. Interaction demandeur-fournisseur (négociation et établissement du contrat)

Cette phase tient lieu quand une liste non vide est retournée par le facilitateur au demandeur ; c'est-à-dire lors de la présence des fournisseurs qui peuvent offrir le service demandé. Un même service peut être fourni par plusieurs fournisseurs mais avec des critères de performances différents, ces critères dépendent de la qualité du service (couts, délai,…etc.). A base de ces critères le demandeur peut faire son choix pour maximiser les profits et minimiser les dépenses. L'existence des services répondants au besoin du demandeur déclenche une autre forme d'interaction ayant pour but le choix du partenaire le plus pertinent qui va exécuter le service cherché, cette forme d'interaction a un seul initiateur qui est le demandeur et plusieurs contractants qui sont les fournisseurs retournés dans la liste ; ces fournisseurs entrent dans une situation de compétition implicite parce que chacun d'eux veut gagner l'offre et ne connait pas les autres adversaires. Cette forme d'interaction est concrétisée par un des protocoles de négociation proposés dans le domaine des systèmes multiagent qui sont FIPA-Contract-Net [9] ou FIPA-Iterated-Contract-Net [64]. Pour cela le demandeur va interroger tous les fournisseurs par un message FIPA-ACL avec un performatif *cfp (Call For proposal)* qui contient le profil du service sollicité. Ainsi, il exige un délai pour la réception des réponses. De l'autre côté chaque fournisseur qui reçoit le message va réagir par :

- L'ignorance du message s'il n'est pas intéressé par la demande reçue.
- L'envoi d'un message au demandeur avec le performatif *not-understood* s'il n'est pas capable d'analyser le contenu du message.

- La création d'une réponse négative pour annoncer au demandeur qu'il n'est pas capable d'assurer l'offre selon les critères imposés par ce dernier. La réponse est envoyée au demandeur dans un message FIPA-ACL avec un performatif *refuse (reason).*
- La création d'une réponse positive pour informer le demandeur qu'il est capable d'assurer le service selon les critères imposés par ce dernier, ainsi il doit mentionner dans cette réponse les critères qu'il impose pour assurer l'exécution du service, la réponse est envoyée au demandeur en utilisant un message FIPA-ACL avec un performatif *propose (precondition).*

Les deux derniers cas sont précédés par une analyse du contenu du message reçu de la part du demandeur.

Après l'expiration du délai fixé, le demandeur entame la phase de sélection qui se déroule comme suit :

- Les massages reçus avec des performatifs *not-understood* ou *refuse* sont ignorés.
- Les messages reçus avec un performatif *propose* sont analysés pour faire le choix du partenaire le plus adéquat pour l'exécution du service et adapter le workflow privé aux critères imposés par l'autre partenaire.

S'il ya une concordance (complémentarités) entre les critères souhaités et les critères exigés mais l'ensemble des fournisseurs qui offrent le service sont plus d'un fournisseur le demandeur fait une autre sélection interne selon des critères plus précis pour choisir le partenaire le plus convenable et lui envoyer un message FIPA-ACL avec un performatif *accept-proposal (postcondition)* dans ce cas on dit que le contrat est établi entre les deux partenaires et ils peuvent passer à la phase de l'exécution du service et la surveillance de l'exécution. Les autres partenaires qui ne sont pas sélectionnés dans cette phase reçoivent de la part du demandeur des messages FIPA-ACL avec un performatif *reject-proposal (reason)* pour les informer que leurs propositions sont rejetées.

L'existence de contradictions entre les critères exigés et ceux souhaités annule l'établissement du contrat entre les deux partenaires. Dans ce cas, le demandeur envoi un message FIPA-ACL avec un performatif *reject-proposal (reason)* pour prévenir l'autre partenaire que sa proposition est rejetée et l'interaction entre eux se termine.

Dans notre contribution un service est définit par deux ensembles de propriétés, le premier contenant des propriétés radicales portants des valeurs qui ne changent pas au cours de cette phase et le deuxième contenant des propriétés additives qui peuvent avoir des valeurs changeantes.

Exemple : soit un service d'achat en ligne où un article est définit par un nom, une famille, un coût et un délai de livraison. La partie radicale de la définition de ce service est constituée des deux propriétés (nom, famille) qui constituent la partie non négociable du service, le coût et la date de livraison constituent la partie additive dont les valeurs peuvent être changées par les partenaires au cours de la phase de négociation.

A partir de cette définition un agent peut créer deux profiles de service qui sont :

1. Le profile publique contenant une instanciation des deux parties suscitées ; ce profile est utilisé comme un contenu des messages échangés avec les autres partenaires.
2. Le profile privé contenant une instanciation des deux parties où la partie radicale est identique à celle du profile publique et la partie additive porte des valeurs décisionnelles inaccessibles par l'environnement externe de l'organisation.

Si l'on fait une instanciation de la définition du service cité précédemment, en la projetant sur un cas d'achat en ligne des produits électroménagers ; un service est définit par le tuple : article (nom, marque, prix, délai de livraison). A partir de cette définition l'agent crée deux profiles pour un article :

- Le profile publique ayant pour valeurs (congélateur, LG, 30000da,03j)
- Le profile privé ayant pour valeurs (congélateur, LG, [25000da, 32000da], [02j, 05j]).

Les valeurs 'congélateur' et 'LG' constituent la partie radicale du profile de l'article

Le prix et le délai sont représentés dans le profile privé par deux intervalles utilisés comme paramètres de décision dans la phase de négociation. Par contre, dans le profile publique, elles sont représentées par les valeurs exactes.

Quelque soit le service négocié, l'agent demandeur doit être doté de deux algorithmes qui sont :

- L'algorithme qui génère les appels pour les propositions selon le profile publique du service demandé.
- L'algorithme qui évalue les propositions reçues cet algorithme est paramétré par le profile privé du service et le profile publique des propositions reçues des fournisseurs.

L'agent fournisseur doit aussi être doté d'un algorithme lui permettant de créer les propositions, cet algorithme est paramétré par le profile privé du service à fournir et le profile reçu de la part du demandeur dans le message de l'appel pour proposition.

Le schéma suivant illustre l'interaction des acteurs dans cette phase.

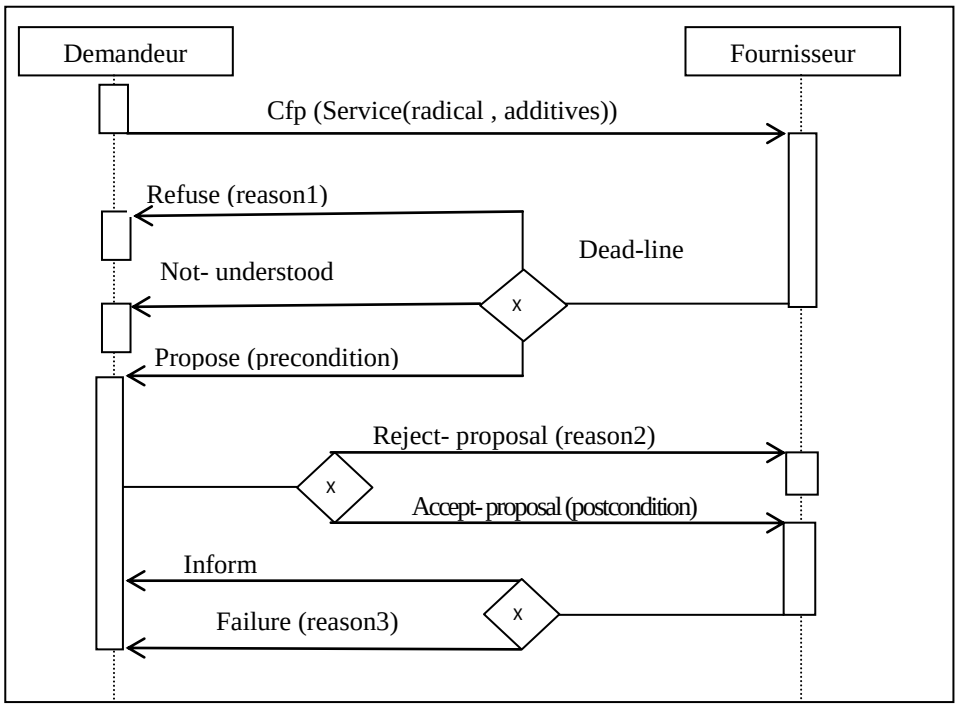

Figure III.3: Diagramme d'interaction entre demandeur - fournisseur

d. Phase de l'exécution du service et de la surveillance

Cette phase tient lieu quand un contrat est établi entre le demandeur et un fournisseur elle se déroule comme suit :

- Si le service est atomique c'est-à-dire que son exécution se déroule en une seule transaction, l'agent fournisseur doit instancier le plan et les ressources nécessaires lors de la réception du message *accept-proposal* du côté demandeur et lance l'exécution du service. Si l'exécution se termine avec succès le fournisseur envoi le résultat au demandeur dans un message FIPA-ACL avec un performatif *inform(resultat).* Si une exception apparait pendant l'exécution du service le fournisseur doit la signaler au demandeur par l'envoi d'un message FIPA-ACL avec un performatif *failure (reason)* pour permettre à ce dernier de prendre d'autres décisions ; dans cette situation on dit que le contrat est résilié entre les deux partenaires.

– Si le service est composé; c'est-à-dire ; que son exécution est effectuée en plusieurs transactions, le contrat est modélisé par une machine d'état fini déterministe. Il est mis en application par un ensemble fini et ordonné de clauses. Chaque clause a un sujet qui représente le partenaire qui exécute l'action, le bénéficiaire de la réalisation de l'action et un ensemble fini et ordonné de messages FIPA-ACL qui relient les clauses et décrivent les messages qui doivent être échangés entre les partenaires pendant l'exécution du contrat. Le contrat a une clause initiale qui est déclenchée par un message FIPA-ACL avec l'acte communicatif *accept-proposal* et finit par l'une des trois clauses suivantes. La première est déclenchée par le message FIPA-ACL avec l'acte communicatif *done* ou *inform* , si le service est exécuté avec succès; la seconde est déclenchée par le message FIPA-ACL avec l'acte communicatif *failure*; dans ce cas on dit que le contrat a échoué et un mécanisme d'une compensation doit être chargé pour réinitialiser le contrat, la troisième est déclenchée par un message FIPA-ACL avec l'acte communicatif *cancel* dans ce cas on dit que le contrat est irrécouvrable et l'exécution est avortée par un des partenaires et un mécanisme de sanction devrait être chargé par l'autre partenaire. Dans le cas habituel quand l'agent fournisseur reçoit le message FIPA-ACL avec l'acte communicatif *accept-proposal* de l'agent demandeur il commence l'exécution de la première transaction du service ensuite il envoie le résultat au demandeur dans un message de FIPA-ACL avec l'acte communicatif *done* ou *informe* (*résult*) et attend la réception du prochain message pour exécuter la transaction suivante. de la même façon, l'agent demandeur quand il reçoit un message de la part du fournisseur ; il l'évalue selon les critères indiqués comme conditions préalables, exécute l'action indiquée dans la clause et envoie le prochain message au fournisseur et ainsi de suite. Le contrat se termine par l'une des situations citées précédemment.

2.3) Micro Architecture

La microarchitecture permet d'identifier la structure interne de l'agent, elle dépend directement du rôle à jouer au niveau inter-organisationnel et intra-organisationnel. On distingue les éléments suivants :

– Un moteur du workflow : c'est l'élément qui assure l'instanciation et l'exécution des processus workflow, ainsi il assure la surveillance des instances en cours d'exécution. Cet élément est spécifié avec des règles ECAPE' [65], Où :
  - E représente l'évènement reçu, qui peut être *(i)* un message FIPA-ACL envoyer de la part d'un autre agent, ou *(ii)* un évènement déclencher par un utilisateur ou par une application au seine de l'organisation représentée par l'agent ou *(iii)* un évènement interne déclencher dans l'agent lui-même.

- C représente une expression logique à tester pour la sélection de l'action A à exécuter, ces expressions sont utilisées par l'agent comme des filtres pour les évènements reçus.
- A représente l'action a exécuté par l'agent après le recevoir de l'évènement E et l'évaluation de l'expression C, cette action peut être une action atomique ou composé.
- P représente la post-condition à vérifier après l'exécution de l'action A, elle est aussi une expression logique pour produire l'évènement E'.
- E' représente l'évènement générer par l'agent après l'exécution de l'action et l'évaluation de la condition P, il peut être un message FIPA-ACL à envoyer à un autre agent ou une mise à jour d'une connaissance dans son base des connaissances ou un évènement envoyer à une application ou l'utilisateur.

Il existe une autre forme de ces règles appelées règles ECA [59] caractérisées par l'absence des postes conditions et l'événement E' généré pour leur évaluation

- La bibliothèque des plans : Un plan est un processus workflow utilisé par l'agent pour atteindre un but bien spécifié ou pour interagir avec un évènement reçu. ce sont des instances de la classe *ServiceWorkflow*. on peut utiliser la notation BPMN[66] pour représenter chaque plan.
- la base de connaissances qui inclue les croyances de l'agent sur lui-même et sur son environnement. Par exemple les services a recherché, les services en cours de négociations,…etc.
- interface utilisateur ; qui permet à l'utilisateur et l'agent d'interagir entre eux.
- Couche d'interaction avec des applications intra-organisationnelle qui sont invoquées par l'agent pour exécuter des tâches bien déterminées.
- La couche d'interaction utilisée pour envoyer ou recevoir des messages FIPA-ACL

La figure suivante illustre l'architecture de l'agent :

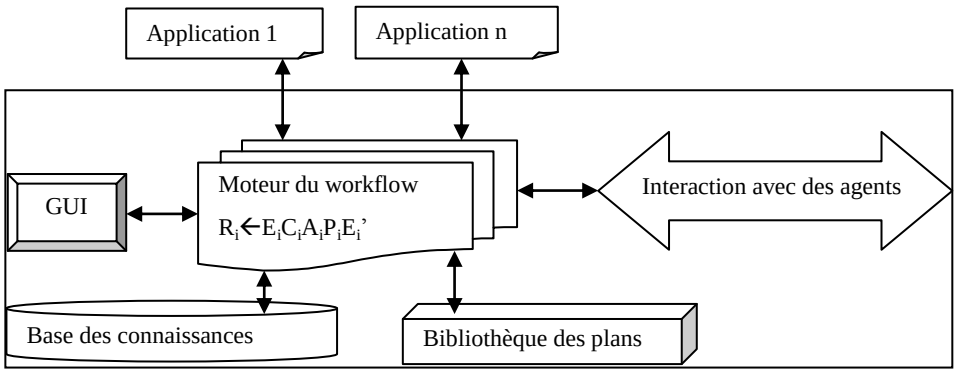

Figure III.4 : Architecture interne de l'agent

Dans les sections suivantes nous définissons les éléments les plus importants de chaque agent.

a. Agent facilitateur ; cet agent joue le rôle du contractant des les différentes interactions, il est influé par deux types des messages FIPA-ACL qui sont les messages avec des performatifs *query* et *request*, c'est-à-dire que l'agent doit interagir avec ces deux évènements externes par l'analyse de leurs contenus et l'exécution de l'action appropriée puis retourné le résultat à l'expéditeur dans un message FIPA-ACL avec un des performatifs suivants *inform, not-understood, agree, failure*. Ces messages sont des éléments de l'ensemble E'. L'agent utilise les règles ECAPE pour interagir avec chaque message reçu. La forme de chaque règle est :

Ri On <Event> If <Condition> Do <Action> Check <Post condition>Trigger <post Event>

- La bibliothèque des plans de cet agent contient :
  - *Plan1* pour insérer un nouveau profile du service dans ça base de connaissance.
  - *Plan2* pour mettre à jour un profile du service déjà existe dans la base de connaissance.
  - *Plan3* pour supprimer un profile du service de la base de connaissance.
  - *Plan4* pour la mise en correspondance du profile du service demandé avec les profiles stockés dans ça base de connaissance.
- La base de connaissance de cet agent contient les profiles de services publiés ;
- Les règles utilisées sont :

R1 On <message.request> If <message.content==publish> Do <execute.Plan1> Check <post condition>Trigger <message.agree|message.failure|message.not-understood>.
R2 On <message.request> If <message.content==upDate> Do <execute.Plan2> Check <post condition>Trigger <message.agree|message.failure|message.not-understood>.
R3 On <message.request> If <message.content==delete> Do <execute.Plan3> Check <post condition>Trigger <message.agree|message.failure|message.not-understood>.
R4 On <message.query> If <message.content==search> Do <execute.Plan4> Check <post condition>Trigger <message.inform| message.not-understood>.

b. L'agent fournisseur ; cet agent doit avoir les éléments suivant :

- *GUI* qui permet à l'utilisateur d'interagir avec eu pour insérer un nouveau service, mettre à jour les éléments d'un service et supprimer un service.

- la bibliothèque des plans qui inclus :
  - *Plan1* pour créer un profile d'un service à publié.
  - *Plan2* pour mettre à jour le profile déjà publié.
  - *Plan3* pour demander la suppression d'un profile déjà publié.
  - *Plan4* pour créer une proposition durant la phase de négociation.
  - *Plan5* pour exécuter un contrat s'il est établé pendant la phase de négociation.
- Une base de connaissance qui inclut :
  - Les services publiés.
  - Les services en cours de négociation.
  - Les services en cours d'exécution.
- Le moteur workflow qui inclut les règles suivantes :
  - *R1 on <event.newService> If <inPuts.verified> Do <execute.Plan1> Check <outPuts > Trigger <msg.request (publish(service)>.*
  - *R2 On <event.upDateService> if <service is published> Do <execute.Plan2> Ckeck <outPuts> Trigger <msg.request (upDate (service)>.*
  - *R3 On <event.deleteService> if <service is published> Do <execute.Plan3> Ckeck <outPuts> Trigger <msg.request (delete(service))>*
  - *R4 On <msg.cfp> if <inputs> Do <execute.Plan4> Check <outPuts> Trigger <msg.propose|msg.refuse|msg.not-understood>.*
  - *R5 On <msg.accept_proposal> if <inPuts> Do <execute.Plan5> Check <outPuts> Trigger <msg.inform|msg.done|msg. Failure>*

c. Agent demandeur ; cet agent doit avoir aussi les éléments suivants :

- *GUI* qui permet à l'utilisateur d'interagir avec eu pour insérer un nouveau besoin.
- Bibliothèque des plans qui inclut :
  - *Plan1* pour créer le profile du service à rechercher et interroger le facilitateur.
  - *Plan2* pour initialiser une nouvelle phase de négociation.
  - *Plan3* qui lui permet de faire la sélection des partenaires dans la phase de négociation.
  - *Plan4* qu'est utilise dans la phase de l'exécution du contrat.
- La base de connaissance qui inclut:
  - La liste des services à recherchés.
  - La liste des services en cours de recherche.
  - La liste des services en cours de négociation.
  - La liste des services en cours d'exécution.
- Le moteur qui inclut les règles suivantes :
  - *R1 On <event.newService> if <inPuts are verified> Do <execute.Plan1> Check <serviceProfil> Trigger <(msg.Query (searsh(service))>.*

- *R2 On <msg.inform> if <msg.content> Do <execute.Plan2> Check <outPuts> Trigger (msg.cfp (service)>*
- *R3 On <msg.propose> if <msg.content> Do <execute.Plan3> Check <outPuts> Trigger <msg.accept_proposal| msg.reject_Proposal>*
- *R4 on <msg.inform|msg.done> if <msg.content> Do <execute.Plan4> Check <outPuts> Trigger (event.upDate (beliefBase)>.*

3. Conclusion

Dans ce travail nous avons proposé une approche basée sur les systèmes multi agents pour résoudre quelques problèmes émergents dans les workflow interorganisationnels lâches notamment ceux liés à la recherche des partenaires, la négociation et l'établissement des contrats. Cette approche a l'avantage de permettre à l'organisation d'interagir avec son environnement sans affecter son autonomie.

Dans le chapitre suivant nous allons mettre le modèle proposé en application en faisant une projection sur un cas réel pour nous permettre de déceler les anomalies et les insuffisances existantes.

Chapitre IV :

Etude de cas et perspectives d'implémentation

1. Introduction

Dans le chapitre précédent nous avons proposé une architecture à base d'gents pour les workflow interorganisationnels faiblement couplés. Cette architecture est indépendante de tout secteur d'activité.

Pour nous permettre d'éclaircir les idées incluses dans l'architecture proposée et montrer son adaptabilité nous allons la projeter sur un cas réel.

2. Etude de cas : Organisation des stages pratiques entre les établissements de formation et les entreprises

2.1) description de l'environnement

Dans le domaine de la formation professionnelle ou de la formation universitaire, les stagiaires ou les étudiants doivent suivre des stages pratiques à la fin de leurs cursus. Ces stages ont pour rôles de permettre aux stagiaires d'acquérir le savoir faire et d'appliquer les savoirs théoriques acquis. De l'autre coté, les organisations qui activent dans le milieu professionnel et qui ont besoin de la main d'œuvre spécialisée exploitent ces stages pour intégrer les stagiaires les plus qualifiés afin d' assurer une gestion plus raisonnable et une amélioration au niveau de la productivité.

Pour atteindre leurs objectifs les deux types d'organisation doivent faire une planification des stages, cette planification nécessite une automatisation pour garantir plus de rationalité et de transparence dans la répartition des stagiaires sur les organisations activant dans le domaine professionnel.

2.2) Les éléments de l'architecture

A partir de l'énoncé du cas nous pouvons définir les rôles suivants :

Les établissements de formation qui jouent le rôle de demandeurs de stages pratiques ils doivent planifier périodiquement des stages et effectuer une recherche des entreprises les fournissant à travers le facilitateur pour désigner celles qui répondent aux critères exigés.

Les entreprises qui jouent le rôle de fournisseurs de services elles peuvent publier des stages, mettre à jour un stage déjà publié ou supprimer un stage s'il n'est plus disponible.

Le service dans ce cas représente le stage il est défini par un thème, une spécialité, une durée, une période et un nombre de postes

Chaque rôle (demandeur ou fournisseur) est représenté par un ensemble de capacités (capabilities) et chaque organisation qui veut le jouer doit créer un agent logiciel pour assurer l'interaction avec son environnement en utilisant les données qu'il reçoit de la part d'un agent humain ou une application appartenant à l'organisation qu'il représente.

- Le facilitateur : il représente l'élément intermédiaire qui assure la première étape d'interaction entre le demandeur et le fournisseur il doit être connu par les organisations.

L'acteur qui joue ce rôle doit être doté des capacités suivantes :

a. Vérifier si une nouvelle offre de service à publier est valide
b. Assurer la mise à jour des services publiés selon les demandes des entreprises.
c. Supprimer des stages s'ils ne sont plus disponibles.

Le facilitateur agit selon les messages qu'il reçoit de la part des organisations.

3. Perspectives de l'implémentation

L'implémentation de chaque type d'agent sera présentée dans les parties suivantes.

3.1) Choix de la plateforme

La première étape concernant le travail pratique consiste à choisir une plateforme orientée agent pour implémenter le modèle. Il existe des bonnes références qui présentent plusieurs outils. .

Concernant notre modèle, nous proposons les critères suivants pour faire le choix de la plateforme :

- La plateforme doit soutenir le modèle BDI
- La plateforme choisie est un logiciel libre ayant une source ouverte
- La plateforme est compatible avec la norme FIPA (pour simplifier et faciliter la programmation et le développement)

Parmi les plates-formes existantes et respectant les critères ci-dessus nous avons décidé de choisir JADEX comme environnement de développement en raison de sa flexibilité (pour l'intégration dans les autres plateformes), de sa compatibilité avec la norme de FIPA et de sa documentation suffisante.

3.2) JADEX – un moteur de raisonnement avec le modèle BDI

Les agents intelligents sont un paradigme de modélisation, basé sur la notion d'agents avec états mentaux. Le moteur de raisonnement Jadex suit le modèle de croyance (Belief), de désir (Desire), d'intention (Intention) – BDI. Il facilite la construction des agents intelligents et peut être déployé sur différents genres de middleware tels que le JADE (http://jade.tilab.com/).

Jadex est un moteur de raisonnement orienté agent, la description des agents raisonnables est effectuée en utilisant XML et le langage de programmation Java. De plus, les agents de Jadex peuvent être programmés dans les environnements de développement intégrés (IDEs) comme Eclipse, etc.

3.3) Caractéristiques

En résumé, Jadex est *basé sur Java*, un des environnements de développement des agents *compatible avec la spécification de l'agent de FIPA* (http://www.fipa.org/), il

permet de développer les *agents orientés par but* selon le modèle de BDI. Jadex fournit également un cadre et un ensemble d'outils de développement pour simplifier la création et l'essai des agents. Les sections suivantes soulignent certains caractères courants de Jadex :

- *basé sur Java*: Le projet de Jadex vise à rendre le développement des systèmes basés sur agents le plus facile possible. Avec Jadex, on peut créer des systèmes multi-agents sans avoir besoin d'apprendre un nouveau langage de programmation. Jadex est conçu pour faciliter l'exécution des agents dans le langage de programmation Java, il permet don de réutiliser plusieurs des outils et des bibliothèques existants.

- *Compatible avec la spécification de l'agent de FIPA* : Un des facteurs primaires de succès d'une nouvelle technologie est la disponibilité des normes pour garantir l'interopérabilité entre les plateformes différentes. Afin de faciliter l'interopérabilité des systèmes multi-agents développés indépendamment, FIPA a défini un ensemble de caractéristiques désigné généralement sous le nom « la norme de FIPA ». Selon les indications de la figure ci-dessous, la norme de FIPA indique une architecture de plateforme d'agent, qui définit des services tels que la gestion d'agent et un facilitateur d'annuaire (*directory facilitator* en anglais). Cette architecture permet à des agents de communiquer en utilisant un langage de communication commun. Pour réaliser FIPA-compliance, Jadex est basé sur JADE, un outil source ouvert qui fournit l'architecture de plateforme et les services de noyau et les mécanismes de transport de messages selon les exigences des caractéristiques de FIPA.

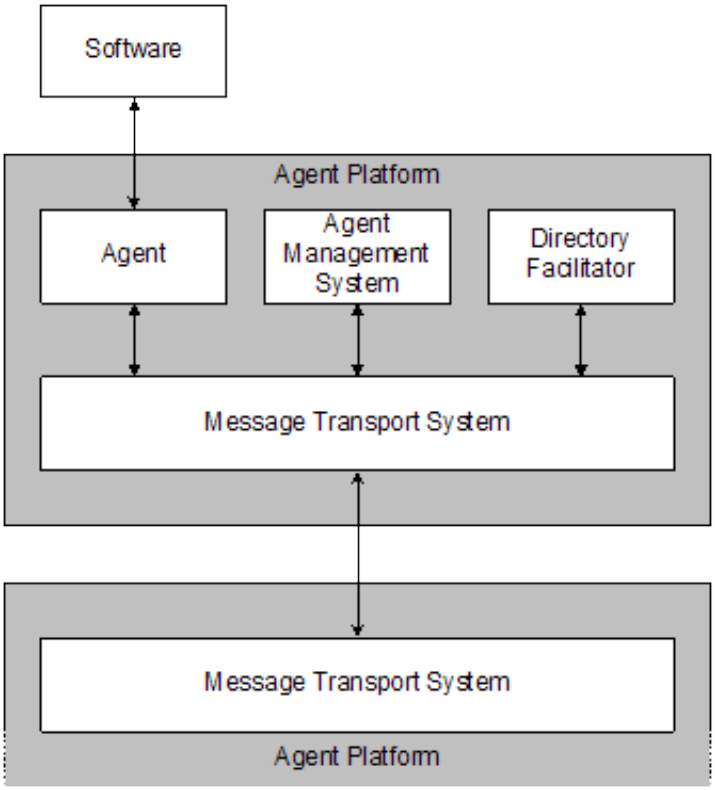

Figure IV.1: Le modèle de référence pour une plate-forme multi agents FIPA

- *Outils de développement* : Un signe important de la qualité de n'importe quel environnement de développement est la disponibilité des outils. Jadex est construit en se basant sur JADE, donc les nombreux outils disponibles qui sont inclus dans JADE peuvent également être utilisés avec Jadex. D'autre part, les nouveaux concepts présentés par Jadex doivent être aussi bien soutenus. La plupart des outils sont liés au modèle de BDI. L'outil de *BDI Viewer* permet de regarder l'état interne d'un agent de Jadex, c'est-à-dire sa croyance, ses buts et ses plans courants. Le Jadex *Introspector* est semblable au JADE Introspector, permet de surveiller et influencer l'exécution d'un agent, en observant et en influençant la manière dont des événements entrants sont manipulés. Pour la correction, l'Introspector laisse également mettre un agent dans le mode à pas unique.

3.4) Architecture abstraite d'un agent en JADEX

Jadex incorpore le modèle BDI aux agents de JADE en présentant la croyance, les buts et plans comme les classes de base des objets. Tout cela peut être créé et manipulé à l'intérieur de l'agent. Dans Jadex, les agents ont une croyance, dans tous les cas un objet de Java. Les croyances sont stockées dans une base de connaissances (beliefbase). Les buts représentent les motivations concrètes (par exemple les états à réaliser) et influencent le comportement d'un agent. Pour réaliser ses buts, l'agent exécute les plans qui sont des procédures elles aussi codées en Java.

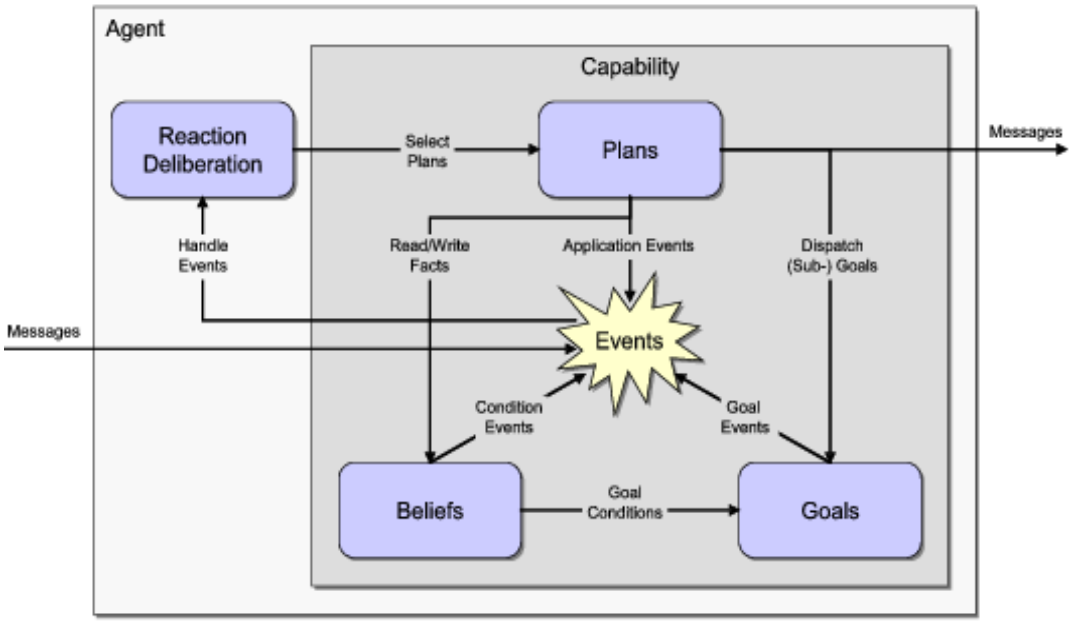

Figure IV.2 : architecture abstraite d'un agent en JADEX

4. Implémentation du modèle

Comme mentionné dans les sections précédentes une organisation est représentée par un agent chargé d'effectuer les tâches attribuées au rôle qu'elle doit jouer (demandeur de

stage qui peut être un établissement de formation ou fournisseur de stage qui peut être une entreprise ou un établissement administratif).Concrètement, chaque agent est modélisé dans la plateforme JADEX par un fichier prenant un nom et une extension doublée « nom agent.agent.xml ». Ce fichier appelé ADF (Agent Definition File) , ayant comme contenu un ensemble de Tags (voir le réf) , est inclus dans un package contenant un ensemble de classes Java . La majeur partie de ce package représente des plans utilisés par l'agent pour interagir avec son environnement suivant les entrées qu'il reçoit (messages, changement de croyances), la partie restante représente des classes pour les objets utilisés par l'agent comme croyances.

4.1) *L'organisation*

Elle joue le rôle du fournisseur de service elle est représentée par l'agent fournisseur. Son workflow est représenté par le diagramme BPMN suivant :

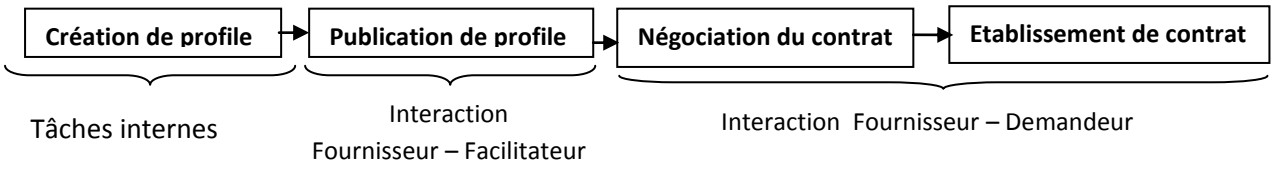

Figure IV.3: Diagramme BPMN du workflow du fournisseur

L'agent crée le profile du service à offrir, ce profile dépend des informations reçues de l'utilisateur. La figure suivante représente l'interface offerte par l'agent à l'utilisateur pour saisir les données relatives aux stages à publier.

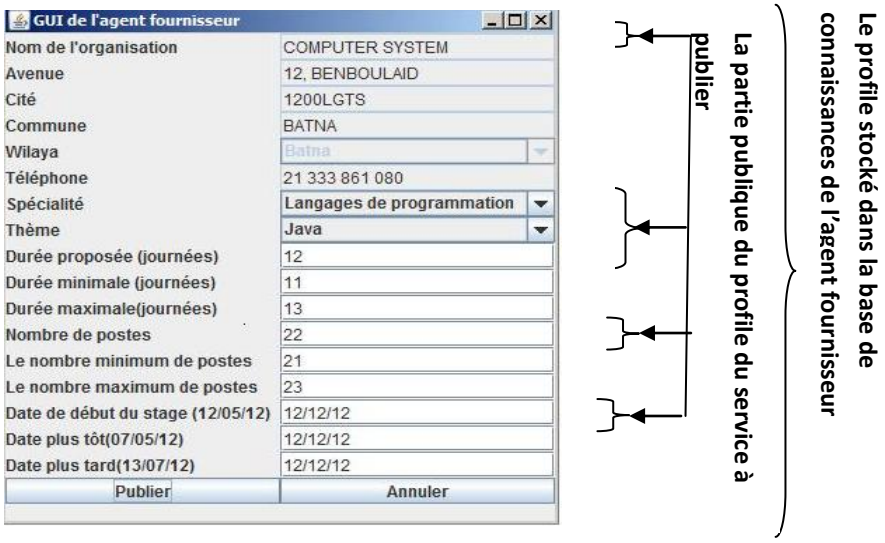

Figure IV.4: GUI de l'interface du fournisseur

En cliquant sur le bouton « Publier » l'agent déclenche le premier plan ayant pour rôle de créer le profile privé du service et mettre à jour la base de croyances de l'agent en

utilisant toutes les informations reçues de la part de l'utilisateur à travers l'interface puis il déclenche un deuxième plan ayant pour rôle d'envoyer la partie publique du profile créé au facilitateur pour le rendre perceptible par l'environnement de l'agent. La partie publique du profile saisi est définie par le tuple suivant: (thèmes proposé (name), spécialité (type), nom de l'organisation (ownership), la durée proposée (duré1), nombre de postes (nbrp1), la date de début du stage (date1)). Ces informations constituent des éléments de base pour un établissement demandeur dans la phase de sélection de liste des fournisseurs avec lesquels il va négocier pour trouver le fournisseur répondant aux critères exigés afin d'établir un contrat d'exécution du stage. La figure ci-après représente des publications dans le facilitateur de la part des organisations offrants des stages dans le domaine informatique.

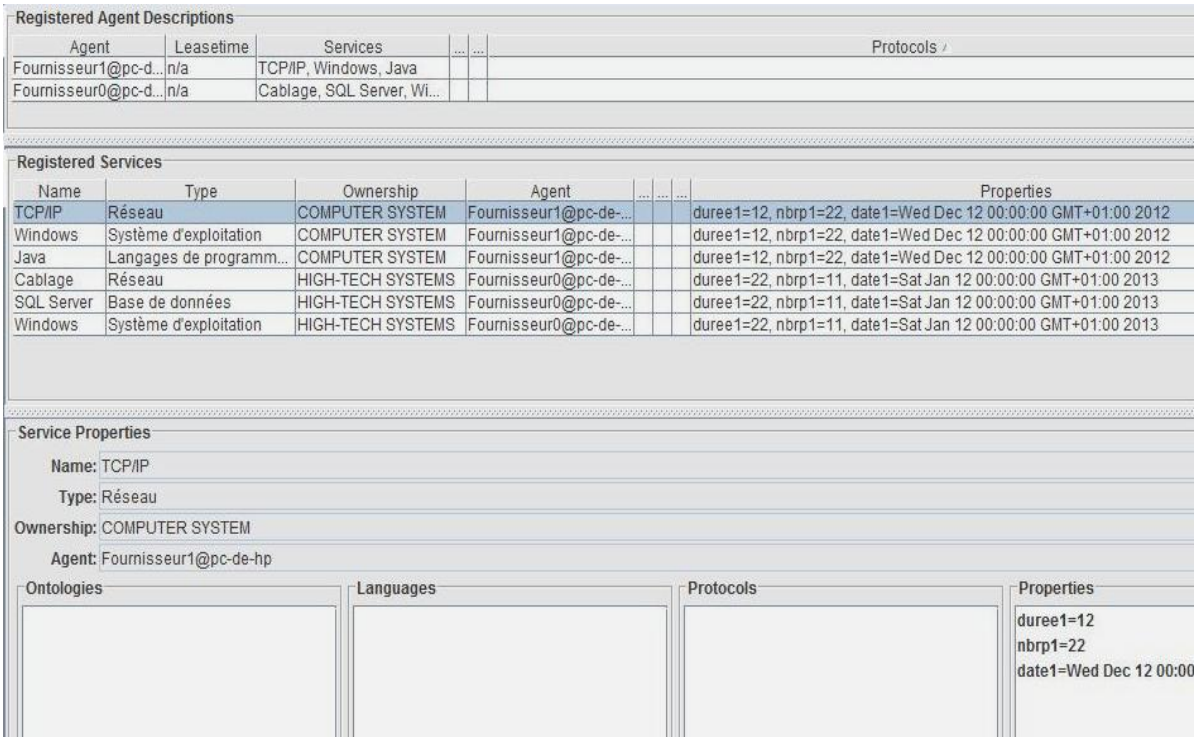

Figure IV.5: publication de services dans le facilitateur

NB : Le facilitateur est représenté par le composant ***Directory facilitator*** de la plate-forme JADEX

4.2) *L'établissement de formation*

Il joue le rôle du demandeur de service, il est représenté par l'agent demandeur. Son workflow est représenté par le diagramme BPMN suivant :

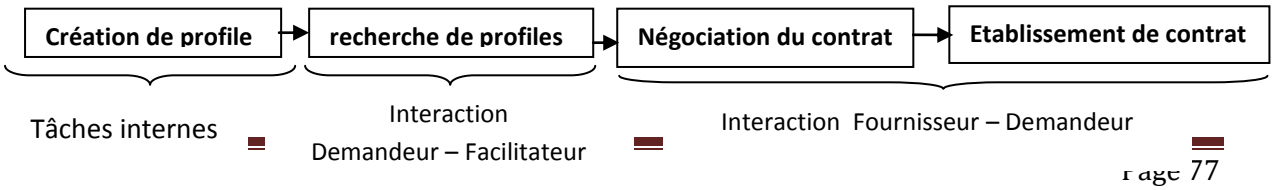

Figure IV.6: Diagramme BPMN du workflow du demandeur

De la même manière, l'agent demandeur offre à l'utilisateur une interface lui permettant de saisir les informations de l'établissement de formation ainsi que les informations relatives au stage recherché. La figure suivante représente l'interface offerte par l'agent à l'utilisateur pour saisir les données relatives aux stages recherchés.

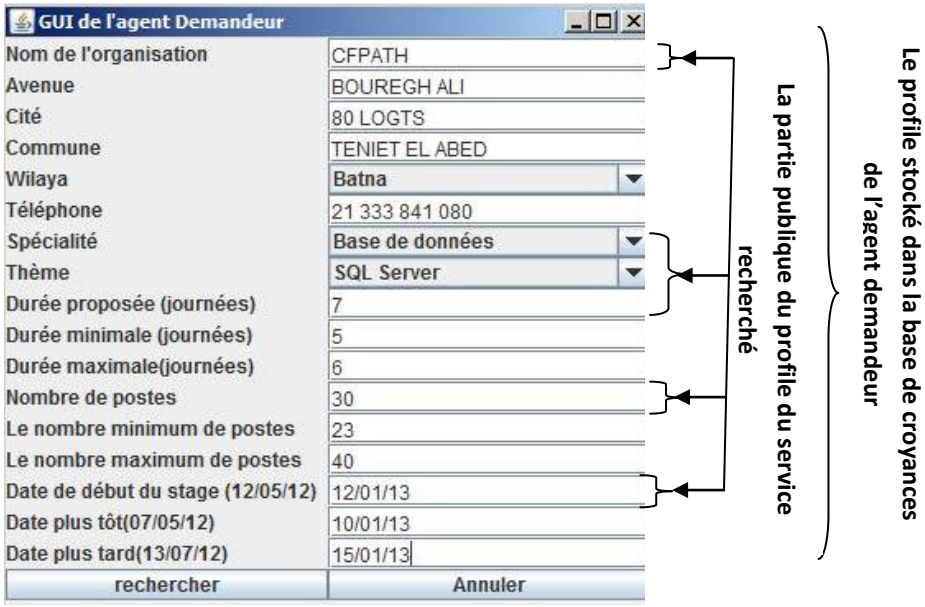

Figure IV.7: GUI de l'interface du demandeur

En cliquant sur le bouton « rechercher » le demandeur déclenche le premier plan consistant à créer le profile du service recherche et mettre à jour la base des croyances de l'agent demandeur puis invoque un deuxième plan permettant d'interroger le facilitateur en utilisant la partie publique du profile créé comme paramètre.

Le facilitateur lance une recherche dans sa base de connaissances pour trouver les profiles de services conformément aux critères exigés par le demandeur.

Lors de la réception d'une réponse positive de la part du facilitateur, le demandeur déclenche l'exécution du troisième plan, il s'agit du plan de négociation pour choisir le fournisseur le plus adéquat afin d'établir le contrat d'exécution du service.

En vue de rendre les interactions entre les partenaires du modèle proposé plus et compréhensibles, nous avons pensé d'intégrer un sniffer permettant de visualiser

graphiquement les échanges des messages durant toutes les phases de l'exécution du workflow.

La figure suivante illustre le sniffer des messages échangés entre un demandeur et deux fournisseurs :

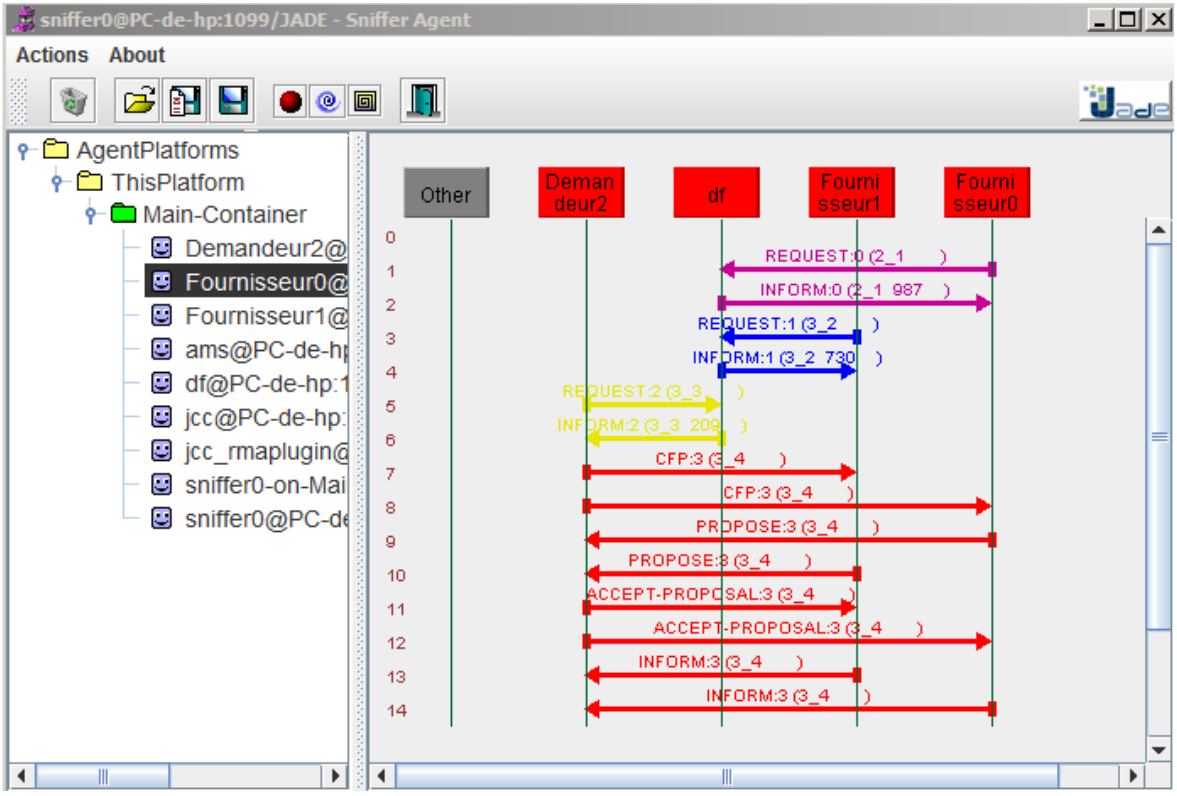

Figure IV.8: GUI de l'interface du sniffer

5. Conclusion

Dans ce chapitre nous avons essayé de mettre en application le modèle proposé a travers une étude de cas, cette dernière a été réalisée dans le cadre du suivi des stages pratiques des étudiants en informatique dans le milieu professionnel.

Nous avons utilisé la plate-forme JADEX pour l'implémentation du modèle proposé.

Le choix de cette plate-forme est dû à sa conformité aux critères que nous avons vus essentiels pour assurer la réalisation des ce travail.

Conclusion générale

1. Rappel de la problématique et les objectifs

L'interopérabilité dans les workflows est un domaine large, très complexe et pluridisciplinaire ; il est basé sur plusieurs champs d'application tels que le génie logiciel, la modélisation de l'entreprise et les systèmes workflows.

Un autre type de complexité concerne la préservation de l'autonomie et du savoir faire de l'entreprise lors de l'interaction avec son environnement.

Notre travail s'inscrit dans la problématique de l'interopérabilité faiblement couplé ou lâche caractérisée par la présence de n organisations disposant chacune d'un workflow local et une structure d'interaction lui permettant d'interagir avec d'autres organisations appartenant à son environnement selon des contrats établis. Il vise principalement sur la recherche d'une solution aux quatre problèmes posés qui sont :

- La description du service
- La recherche des partenaires
- La négociation pour l'établissement du contrat
- L'exécution du service et la surveillance de l'exécution

Dans notre travail nous nous sommes concentrés sur la proposition d'une architecture à base d'agents pour les workflows interorganisationnels faiblement couplés qui donne une solution aux quatre problèmes suscités.

L'architecture orientée agents que nous avons proposée prend ainsi en compte, les contraintes du WIO lâche à savoir :

- L'autonomie des différentes organisations
- La flexibilité requise par le WIO lâche car chaque organisation peut facilement intégrer ou quitter le workflow interorganisationnel,
- L'hétérogénéité sémantique des différents modèles (informationnels, organisationnels ou de processus) des organisations participant au WIO,

2. perspectives

Nous sommes conscients que ce travail présente des limites, il ne prétend pas donner une solution parfaite à la problématique de l'interopérabilité faiblement couplée.

Des améliorations lui peuvent être apportées notamment au niveau du recouvrement des cas d'exceptions, la prise de décision dans les situations d'échec de l'exécution des services et le développement des algorithmes de mise en correspondance sémantique.

De point de vue implémentation ; plusieurs travaux peuvent être effectués notamment pour l'implémentation de l'intégralité des agents.

Références bibliographiques

Références bibliographiques

[1] Jacques Ferber. (1995): Les systèmes multi-agents Vers une intelligence collective, InterEditions,

Paris.

[2] Wooldridge, M. (2002): An Introduction to Multi-Agent Systems. Wiley Editions. England.

[3]Chalupsky, H., Finn, T., Fritzson, R., McKay,D., Shapiro, S., and Wiederhold, G. (1992) : An overview of KQML: A knowledge query and manipulation language, Defense Advanced Research Project Agency (DARPA) White Paper, Washington, DC.

[4]FIPA Repository. http://www.fipa.org/repository/aclspecs.html, accessed on March 22, 2012.

[5]T. Finin, R. Fritzson, D. McKay, and R. McEntire. (1997): KQML as an Agent Communication Language. Software Agents.

[6] Silva, N., Rocha, J. and Cardoso, J. (2003) : E-Business Interoperability through Ontology Semantic Mapping. In Proceedings of the Processes and Foundations for Virtual Organizations, pp. 315-322. Lugano, Switzerland.

[7] FIPA Device Ontology Specification. (2002) : http://www.fipa.org/specs/SI00091.pdf. accessed on March 22, 2012.

[8] James Odell. (2010): Agent Technology:An Overview. Ann Arbor, MI USA.www.jamesodell.com

[9] FIPA Contract Net Interaction Protocol Specification. (2002). http://www.fipa.org/specs/SC00029H.pdf. accessed on March 22, 2012.

[10] Gerhard Weiss. (1999) : Multiagent Systems A Modern Approach to Distributed Modern Approach to Artificial Intelligence, pp. 38-40. The MIT Press Cambridge, Massachusetts, London, England.

[11] Fischer, K. Müller, J. P. and Pischel, M. (1996) : A pragmatic BDI architecture. Intelligent Agents II (LNAI Volume 1037), pages 203-218. Springer-Verlag: Berlin, Germany.

[12] Brown, P. and Rossak, W. (2005) : Mobile Agents. Morgan Kaufmann Publishers and dpunkt.verlag.

[13] Jacques, F. Tiberiu, S. and John, T. (2009) : Towards an Integral Approach of Organizations in Multi-Agent Systems. Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models. Information Science Reference. New York. USA.

[14] Zambonelli, F., Jennings, N. and Wooldridge, M. (2003) : Developing Multi-agent Systems: The GaiaMethodology. ACM Transactions on Software Engineering and Methodology, pp. 317–370. ACMPress.

[15] Bellifemine, F. Caire, G. Greenwood, D. (2007) : Developing Multi-Agent Systems with JADE. Wiley Editions. England.

[16] MadKit (2004) : A Multi-Agent Development Kit. www.madkit.org.

[17] Gutknecht, O. and Ferber, J. (2001). The madkit agent platform architecture. In Revised Papers from the InternationalWorkshop on Infrastructure for Multi-Agent Systems, pp. 48-55. Springer-Verlag. London, UK.

[18] AgentBuilder : http://www.agentbuilder.com.

[19]Zeus : http://www.labs.bt.com/projects/agents/Zeus.

[20]Jadex : http://vsis-www.informatik.uni-hamburg.de/projects/jadex/

[21] FIPA Subscribe Communicative Act Specification. (2001) : http://www.fipa.org/specs/SC00035H.pdf. accessed on March 22, 2012.

[22] Collis, J. Ndumu, D. (1999) : The Zeus Agent Building Toolkit, The Application Realisation Guide.

[23] The Workflow Mangement Coalition. (1999) : Workflow Management Coalition Terminology and Glossary, technical report WfMC-TC-1011.

[24] Amirreza Tahamtan. 2009 : Modeling and Verification of Web Service Composition Based Interorganizational Workflows. Thèse de doctorat. Université de Vienna.

[25]Chebbi Issam. (2007): CoopFlow: une approche pour la coopération ascendante de workflows dans le cadre des entreprises virtuelles. Thèse de doctorat , Institut National des Télécommunications, France.

[26] Lambrinoudakis, C. Kokolakis, S. Karyda, M. Tsoumas, V. Gritzalis, D. and Katsikas, S. (2003):

Electronic voting systems: security implications of the administrative workflow. In Proceedings of the 14th International Workshop on Database and Expert Systems Applications.

[27] Budinska, I. Oravec, V. Gatial, E. Laclavik, M. Seleng, M. Balogh, Z. Frankovic, B. Forgac,

R. Mokris, I. and Hluchy, L.(2007) : A Knowledge Support System For Administrative Workflow Processes. In Proceedings of the Seventh International Conference on Application of Concurrency to System Design.

[28] Muehlberger, R. Orlowska, M.E. and Kiepuszewski, B. (1999) : Backward step: The right direction for production workflow systems. In Proceedings of the Australian Database Conference.

[29] Leymann, F. (1999) : Production workflow: concepts and techniques. Prentice Hall PTR Upper

Saddle River, NJ, USA.

[30] G.T.S. Ho, H.C.W. Lau, C.K.M. Lee, A.W.H. Ip, and K.F. Pun. (2006) : An intelligent production workflow mining system for continual quality enhancement. *The International Journal of Advanced Manufacturing Technology*, 28(7-8), pp.792-809.

[31] Jiang, P. Mair, Q. and Newman, J. (2003) : Using uml to design distributed collaborative workflows: from uml to xpdl. In Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises.

[32] Khetawat, A. Lavana, H. and Brglez, F. (1997) : Collaborative workflows: A paradigm for distributed benchmarking and design on the internet. Technical report, North Carolina State University.

[33] Pudhota, L. Tierney, A. and Chang, E. (2005) : Services integration monitor for collaborative workflow management. In Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise.

[34] Zhao, X. Liu, C. and Yang, Y. (2005) : An organizational perspective on collaborative business

processes. In Proceedings of the 3rd International Conference on Business Process Management.

[35] Huth, C. Erdmann, I. and Nastansky, L. (2001) : Groupprocess: using process knowledge from the participative design and practical operation of ad hoc processes for the design of structured workflows. In Proceedings of the 34th Annual Hawaii International Conference on System Sciences.

[36] The Workflow Mangement Coalition. (1999): Interoperability abstract specification, technical report WfMC-TC-1012.

[37] W. M. P. van der Aalst. (1999) : Process-Oriented Architectures for Electronic Commerce and Interorganizational Workflow. Information Systems, 24(9): pp.639-671.

[38] Boukhedouma, S, Alimazighi, Z. Oussalah, M. and Tamzalit, D. (2011) : Une approche basée SOA pour l'interconnexion de workflows : application au transfert de cas .

[39] Van der Aalst, W. (2000): Loosely Coupled Interorganizational Workflows: modeling and analyzing workflows crossing organizational boundaries. Information & Management, *37* (2): pp.67-75.

[40] Van der Aalst, W. and Weske, M. (2001) : The P2P approach to interorganizational workflows. Advanced Information Systems Engineering. pp. 140-156. Springer.

[41] Grefen, P. Aberer, K. Hoffner, Y. and Ludwing, H. (2000): CrossFlow: cross-organizational workflow management in dynamic virtual enterprises. International Journal of ComputerSystems Science & Engineering , 15 (5): pp.277-290.

[42] Lazcano, A. Schuldt, H. Alonso, G. and Schek, H. (2001): WISE: Process based e-commerce. IEEE Data Engineering Bulletin journal, volume 24, number 1, pp. 46-51, publisher: Citeseer.

[43] Grefen, P. Eshuis, R. Mehandjiev, N. Kouvas, G. and Weichhart, G. (2009): Crosswork: Internet-based support for process-oriented instant virtual enterprises. IEEE Internet Computing, volume 13, number 6, pp. 65-73, publisher: IEEE.

[44] Hofreiter, B. and Huemer, C. and Klas, W. (2002): ebXML: status, research issues, and obstacles. Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems, p.7-16. IEEE.

[45] Leymann, F. Karastoyanova, D. and Papazoglou, M. (2010): Business Process Management Standards. Handbook on Business Process Management 1 Introduction, Methods and Information Systems, pp.525- 529. Springer.

[46] Shapiro, R. (2006): XPDL 2.0: Integrating Process Interchange and BPMN. Workflow Handbook , 183-194.

[47] Banerji, A. Bartolini, C. (2002): Web services conversation language (wscl) 1.0. W3C Note , 14.

[48] Arkin, A. Askary, S. (2002): Web service choreography interface (WSCI) 1.0. Standards proposal by BEA Systems, Intalio, SAP, and Sun Microsystems.

[49] Juric, M. B. (2006): Business Process Execution Language for Web Services BPEL and BPEL4WS 2nd Edition. packt Publishing.

[50] Chun, O. Michael, A. Moe, T. and Hofstede, A. (2010) : Workflow Management. Handbook on Business Process Management 1 Introduction, Methods and Information Systems, p. 396. Springer.

[51] Ouyang, C. Van Der Aalst, W.M.P. Dumas, M. Ter Hofstede, A.H.M. and La Rosa, M. (2007): Service-oriented processes: an introduction to BPEL. Semantic Web services: theory, tools, and applications journal, pp. 155-188. Publisher: Information Science Reference.

[52] Hofstede, A. and Van Der Aalst, W. (2005): YAWL: yet another workflow language. Information Systems , 30 (4), pp.245-275.

[53] Russell, N. ter Hofstede, A. Edmond, D. and van der Aalst, W. (2005): Workflow data patterns: Identification, representation and tool support, Conceptual Modeling--ER 2005 journal, pp. 353-368, publisher: Springer.

[54] van Der Aalst, W.M.P. Ter Hofstede, A.H.M. Kiepuszewski, B. and Barros, A.P. (2003) : Workflow patterns. Distributed and Parallel Databases, volume:14, number:(1), pp. 5-51.

[55] Wil van der Aalst and Kees van Hee. (2000): workflow theory. Workflow Management Models, methods and systems, pp. 264-281.

[56] FIPA English Auction Interaction Protocol Specification. (2001): http://www.fipa.org/specs/XC00031E.pdf. accessed on March 22, 2012.

[57] FIPA Dutch Auction Interaction Protocol Specification. (2001) : http://www.fipa.org/specs/XC00032E.pdf. accessed on March 22, 2012.

[58] W.M.P. van der Aalst. (1999) : Interorganizational Workflows: An Approach based on Message Sequence

Charts and Petri Nets. Systems Analysis - Modelling - Simulation, 34(3), pp. 335-367.

[59] Donghui Lenya et al. (2007): Interorganizational Workflow Execution Based on Process Agents and ECA Rules, pp.x-y.

[60]Lotfi Bouzguenda et al. (2008): Dynamic Plugging Of Business Processes In Cross-Organizational. International Journal of Computer Science and Applications , 5, pp.141-164.

[61] Shen, W. et al. (2007): An agent-based service-oriented integration architecture for collaborative intelligent manufacturing. Robotics and Computer-Integrated Manufacturing. 23(3), pp. 315-325. Published by Elsevier.

[62] FIPA Request Interaction Protocol Specification. (2002) : http://www.fipa.org/specs/SC00026H.pdf. accessed on March 22, 2012.

[63] FIPA Query Interaction Protocol Specification. (2002): http://www.fipa.org/specs/SC00027H.pdf. accessed on March 22, 2012.

[64] FIPA Iterated Contract Net Interaction Protocol Specification. (2002). http://www.fipa.org/specs/SC00030H.pdf. accessed on March 22, 2012.

[65] Mohamed Boukhebouze et al. (2009): A Rule-Based Modeling for the Description of Flexible and Self-healing Business Processes. ADBIS 2009, LNCS 5739, pp. 15–27. Springer-Verlag Berlin Heidelberg.

[66] Stephen, A. Derek, M. (2008): BPMN Modeling and Reference Guide: Understanding and Using BPMN. Published by Future Strategies Inc. Book Division. USA.

ملخص

هذه المذكرة تحتوي مزجا بين فرعين في حقل معالجة المعلومات ، يتعلق الأمر بمجال سير العمليات المؤسساتية الحر أو الغير محكم الترابط الذي يهدف إلى جعل عمليات موزعة ، مستقلة و غير متجانسة تنشط في مؤسسات مختلفة تتعاون باستخدام أدوات معلوماتية قابلة للتكيف مع مختلف الأنساق.

تعتبر المرونة، التكيف و التوزيع من أكبر التحديات التي تواجه سير العمليات في مختلف المؤسسات.

في هذه المذكرة نقترح نموذجا لسير العمليات المؤسساتية ذو طبيعة حرة أو غير محكم الترابط ينقسم تنفيذه إلى قسمين :

- *قسم داخلي يمثل تنفيذ العمليات داخل المؤسسة*
- *قسم خارجي يمثل تفاعل المؤسسة مع محيطها الخارجي*

في هذا النموذج يمثل العامل الإعلامي المحرك الأساسي لسير العمليات حيث يقوم بتنسيق ومراقبة تنفيذ العمليات داخل المؤسسة بالإضافة إلى تفاعلها مع محيطها الخارجي.

كلمات مفتاحية

سير العمليات، العامل الإعلامي، البحث، التفاوض، التعاقد.

Résumé

Dans ce mémoire nous proposons une combinaison entre deux disciplines du domaine informatique, celle du workflow interorganisationnel lâche qui a pour but de faire collaborer des processus workflow répartis, autonomes, hétérogènes, issues et s'exécutant dans différentes organisations, et la technologie des agents qui connait une projection sur divers domaines comme la recherche d'information, la simulation, les systèmes d'aide à la décision,…etc. en raison de leurs adaptation aux différents contextes.

La flexibilité, l'adaptation et la distribution sont vues comme les majeurs défis pour le workflow interorganisationnel. Dans ce travail nous proposons une architecture pour le workflow interorganisationnel lâche à base des agents.

De ce qui précède, on peut dire que l'exécution entière du modèle implémenté est divisée en deux parties : l'intra-exécution, qui exprime l'exécution au sein d'une organisation, et l'inter-exécution, qui représente l'interaction entre les organisations. Un agent joue le rôle du moteur de workflow.il assure la supervision de l'exécution de processus workflow au sein de l'organisation et l'interaction avec l'environnement externe de l'organisation.

Mots clés : *workflow, agents, interaction, ontologie.*