Université Ferhat Abbas, Sétif, Algérie
Faculté des Sciences de l'Ingénieur
Département d'Infomatique

Thèse de Doctorat
en Informatique

Présentée par
Abdelhafid BENAOUDA

Contribution à la conception et à l'implémentation d'un langage de spécification formelle dédié à la e-maintenance des systèmes de production par l'approche des Systèmes Multi-Agents

Soutenue le 19/02/2006
Devant le jury composé de :

Pr. E. Benmahamad	Président	Université de Sétif
Pr. N. Zerhouni	Rapporteur	ENSMM, Besançon, France
Pr. M. Mostefai	Rapporteur	Université de Sétif
Pr. H. Guyennet	Examinateur	Université de Franche Comté, France
Pr. Z. Sahnoun	Examinateur	Université de Constantine
Pr. M. Boufaida	Examinateur	Université de Constantine
Dr. A. Hammad	Examinateur	Université de Franche Comté, France

v

Sommaire

Table des figures

À la mémoire
de mon père !

À ma chère mère !

À Nacéra,
mon épouse !

et à mes enfants :
Hind,
Camélia et Nadjib
et Said !

Introduction

L'entreprise étendue est désormais une réalité. C'est celle qui s'étend au delà des frontières organisationnelles "physiques". Elle est caractérisée par une externalisation d'activités et par le développement de partenariats.

La démocratisation des Technologies de l'Information et de la Communication (TIC) a fait passer ce genre d'entreprises à des e-entreprises où le support privilégié de la communication, de management, du commerce, etc. est le réseau local dédié ou Internet.

Actuellement, les stratégies d'externalisation consistent pour l'entreprise à se procurer auprès des fournisseurs, sur d'autres sites distants, des services ou produits qui étaient auparavant assurés localement, et cela grace au réseau. Ces stratégies conduisent ces entreprises à s'associer avec d'autres entreprises dans une logique de complémentarité de ressources.

La gestion des ressources dans une entreprise étendue est à la fois:

- Une gestion des stocks avec l'établissement de prévisions, des approvisionnements, une gestion de "un ou plusieurs magasins, appartenant à un ou plusieurs sites", des transports, gestion de nomenclatures, donc un ensemble d'activités logistiques classiques, où les règles et les algorithmes de la gestion de stock classiques peuvent être appliqués.
- Une activité technique de mise en oeuvre d'une politique de maintenance, de e-maintenance, particulièrement préventive, d'un ensemble industriel, composé d'un site ou d'un ensemble de sites interconnectés entre eux par le biais d'une plate-forme de e-maintenance, de e-commerce, où l'équilibrage de la disponibilité des ressources entre ces différents sites constitue la condition nécessaire pour le bon fonctionnement de l'ensemble.
- Une politique préventive entre les différentes entités de l'entreprise qui sont directement concernées par la gestion de leurs ressources d'une façon "coopérative", en assurant un équilibrage en cette matière pouvant éviter le sous-stockage ou le sur-stockage.

La modélisation et l'automatisation de cette gestion pour un site local est actuellement une tâche très aisée. Toute une pléiade de solutions dédiées, commerciales et même gratuites, est proposée actuellement.

Mais les solutions dédiées aux entreprises étendues restent des solutions de "tenue de stock" et non pas de "gestion de stock ". Car une gestion de stock des ressources

pour une entreprise, dont les sites, interconnectés par Internet par exemple, nécessite l'étude, la modélisation et la validation d'autres critères (communication, interaction, coopération, négociation, e-commerce, etc.).

Il s'agit de rechercher une modélisation adéquate pouvant répondre aux spécifications du problème posé, que nous considérons complexe. Cela est dû au fait que le modèle structurel est particulier pour chaque site de l'ensemble, ce qui pose, d'un coté, un problème d'hétérogénéité entre les différents sites du système, d'autre coté le modèle d'interaction entre eux.

Dans ce contexte, nous avons opté pour une modélisation basée les Systèmes Multi-Agents (SMA), Car, nous considérons que cette approche, a fait sa preuve pour la modélisation des systèmes distribués complexes. Ce qui va nous permettre de répondre aux spécifications du problème, c'est-à-dire, gérer de façon optimale, les ressources distribuées du système d'une façon rationnelle.

Cet objectif ne peut se réaliser que si les sites de l'entreprise optent pour une stratégie de coopération entre eux. Ce qui qui va nous permettre d'assurer, en utilisant une politique préventive, un équilibrage pouvant apporter à l'entreprise étendue la disponibilité continue et optimale des ressources, donc de la valeur ajoutée.

Et comme parmi ces ressources, la pièce de rechange représente une des ressources les plus importantes pour une entreprise, car c'est elle qui est le pilier de la maintenance préventive, et cette dernière qui est actuellement la stratégie de maintenance privilégiée dans la quasi-totalité des entreprises. Nous appliquerons donc sa gestion, comme jeu d'essai, à notre étude.

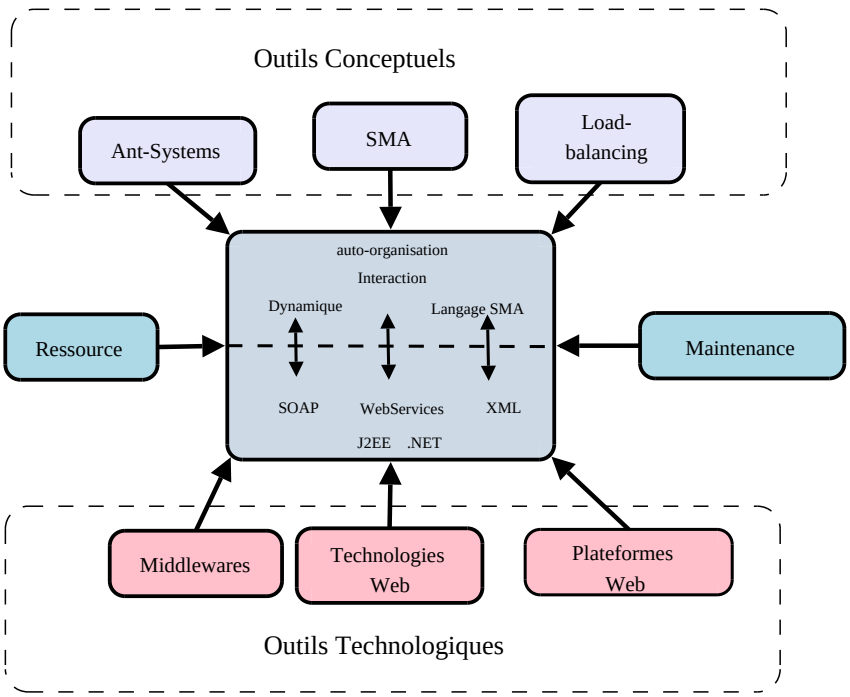

FIG. 1 – *Elements Conceptuels et technologiques intervenant dans cette contribution*

Cette étude nous pousse à travailler sur deux plans connexes, l'un dépendant de l'autre:

1. Une contribution académique conceptuelles: dans ce cadre, notre choix de SMA pour modéliser le problème, nous poussent à travailler sur:
   - un modèle d'organisation dédié au problème posé,
   - ainsi qu'un modèle d'interaction qui, une fois le SMA en oeuvre, répond fidèlement au modèle d'organisation.

   Cela nous pousse à doter nos agents de capacité de réagir aux situations posées. Ce qui nous incitent à rechercher, entre autres, des modèles mathématiques pouvant décrire les scripts d'agents du système.
2. Une contribution sur la bonne utilisation des technologies d'implémentation. A savoir faire une étude sur une politique d'intégration d'outils et de plate-formes logiciels existants, et savoir quoi développer et quoi intégrer? et quelles étapes du cycle de vie de la création de la plate-forme finale.
   Dans ce cadre, les éléments technologiques que nous devons bien étudier et cerner afin de bien les utiliser sont:

   (a) Les middlewares, car il représentent le support qui mettent les agents de notre système en communication, par conséquent, en interaction. Le langage SMA qui sera utilisé dans ce contexte ne prendra pas en considération forcément les langages FIPA classiques tels que KQML ou ACL, car nous nous baserons sur XML comme langage sémantique qui véhiculera nos messages.
   (b) Les technologies Web d'une façon générale, Services Web, XML, SOAP. Car ce sont elles qui nous permettent de mettre en oeuvre notre plate-forme sur le Web.

Afin de réaliser les objectifs cités ci-dessus, nous avons présenté notre processus de réflexion dans cette thèse qui est composée des chapitres suivants:

Chapitre1: Systèmes Complexes et Systèmes Multi-agents. Dans ce chapitre nous présentons un état de l'art du paradigme des Systèmes Multi-Agents, avec une empreinte personnelle que nous considérons importante, car nous présentons ce paradigme comme étant un aboutissement de plusieurs autres courants d'idées dont les langages d'acteurs (ACT, MERING, etc.). Dans ce chapitre, nous présentons aussi les différentes plate-formes dédiées à ce paradigme, avec critiques.

Chapitre2: L'Interaction: la dynamique d'un Système Multi-agent. Dans ce deuxième chapitre, les interactions dans les Systèmes Multi-Agents sont présentées, elles constituent la dynamique du système. Nous y présentons les différents protocoles d'interaction, leur classification. Dans ce cadre, nous faisons une nette séparation entre un langage de communication et un protocole d'interaction.

Chapitre3: Vers un équilibrage de charge dans les Systèmes Multi-agents. Dans ce troisième chapitre, nous utilisons les techniques utilisées dans l'équilibrage de charge, particulièrement dans les réseaux informatiques. Car nous les appliquons

dans le domaine des Systèmes Multi-Agents. Comme le problème posé est la gestion des ressources distribuées, nous associons la consommation d'une ressource à une charge qu'on doit bien gérer et d'une façon optimale.

Chapitre4: Les ressources distribuées: vers un équilibrage coopératif préventif. Dans ce chapitre, apparaît une de nos principales contributions qui est l'équilibrage coopératif des ressources dans une e-entreprise dont les sites doivent opter pour une politique de coopération dans la gestion des ressources afin de réaliser l'équilibrage des ressources pour toute la e-entreprise. Nous y proposons une modélisation à agents coopératifs, distribuons les rôles à ces agents et nous présentons aussi les scriptes des agents du système. Dans ce chapitre aussi, nous présentons une étude sur les méthodes de calcul du niveau de stock de ressources est présentée (méthode d'ABC, méthode de Wilson). Mais nous insistons sur notre propre méthode qui est basée sur l'approximation des moindres carrés afin de présenter le scripte de l'agent dit de prévention. Dans ce cadre, afin de valider nos propositions, ce chapitre présente une étude de cas, qui a été bien implémentée.

Chapitre5: Application à la e-maintenance d'une entreprise étendue. Au cinquième et dernier chapitre, l'application du modèle proposé sur une plate forme de e-maintenance est abordée. Un modèle d'interaction basé sur le paradigme des SMA et qui se base sur le principe d'agent facilitateur est présenté avec présentation d'étude de tous les cas d'exception associés à ce modèle. Nous y présentons aussi comment encoder nos messages en XML/SOAP.

Conclusion Générale. Une conclusion générale avec perspectives cloture notre thèse.

Chapitre 1

Systèmes Complexes et Systèmes Multi-agents

Dans ce chapitre, nous présentons un état de l'art sur les Systèmes Multi-agents, outil de modélisation par excellence des Systèmes complexes, avec une vision personnelle des origines de ce paradigme, ainsi qu'une étude comparative avec les concept d'acteur et d'agent. Nous présentons aussi les modèles d'organisation des SMA, la communication dans les SMA, ainsi que les plate-formes dédiés à ces systèmes et nous finissons ce chapitre par une critique.

1.1 Introduction

Les besoins de communication et de traitement de l'information des entreprises sont de plus en plus croissants. Ils exigent la conception et le développement de systèmes complexes et intelligents. La variété des services requis, la diversité et la répartition des ressources physiques et de connaissances, ainsi que les changements dynamiques des besoins des utilisateurs, requièrent une approche nouvelle, capable de contenir toutes ces contraintes, pour résoudre ou aborder les problèmes complexes. Les systèmes d'information sont actuellement de plus en plus complexes, répartis sur un ou plusieurs sites. Ces derniers sont en interaction entre eux, ou avec des êtres humains, via des réseaux dédiés ou le réseau Internet. Les critères de distributivité et d'interaction entre les partenaires de ce genre d'entreprises, rendent leur modélisation difficile par les outils classiques, tels que les réseaux de Petri, les grafcet ou l'approche orientée objet. Il s'avère que ces outils sont inadéquats et répondent mal à la modélisation de ce genre de problème. Les Systèmes Multi-Agents (SMA) qui sont nés du besoin à répondre à cette modélisation afin de simuler et même implémenter ce type de problèmes distribués et interactifs; se veulent un substitut à ces outils. Dans ce chapitre, nous exposerons les principaux aspects de cette approche, et situerons notre contribution au sein même de ce paradigme.

1.2 Concepts de base

1.2.1 Système

Un Système S est un ensemble d'éléments (et / ou de systèmes) matériels ou logiciels, interagissant en formant un tout. Une boite à vitesse étudiée en soi est un système, ses engrenages sont parmi ses éléments. Cette boite à vitesse n'est qu'un sous ensemble lorsqu'on la considère comme organe d'une machine telle qu'une voiture. Une plate-forme logicielle, telles que J2EE ou .Net, est un système logiciel composé d'un ensemble de sous-systèmes logiciels, représentés par des packages ou des namespaces.

1.2.2 Environnement

L'environnement d'un système, appelé aussi co-système, noté Env(S), est la partie du reste de l'univers U qui regroupe les éléments, les organes et machines, extérieurs à S mais interagissant avec S de façon non négligeable. Nous le notons Env(S) U-S.

1.2.3 Modèle

Un modèle est une représentation simplifiée d'un système réel. Il sert à comprendre le fonctionnement de ce système. On appelle donc un modèle d'un système S un Système Formel M :

- représentant tout ou partie du système S,
- observé d'un certain point de vue,
- dans un formalisme F, de telle façon que l'on puisse au moins, dans un domaine donné, répondre correctement à des questions Q sur S par l'intermédiaire de questions K sur M.

Un modèle devra donc avoir :

- une capacité descriptive.
- Au-delà,une capacité prédictive.

Le coût de complexité d'un modèle est un nombre attaché au nombre de concepts, de variables Ĕ, et donne une idée de sa difficulté d'emploi. Un modèle ni coûteux ni complexe sera dit économique ou parcimonieux, dans la ligne du précepte d'économie de G. d'Occam[1] .

1.3 Au départ, c'était le modèle acteur !

Alors que l'approche déclarative en intelligence artificielle préconisait l'emploi d'un paradigme central et général, basée sur la centralisation de toute l'expertise nécessaire à la résolution d'un problème donné. Certains contestaient la validité d'un tel modèle,

1. "Ne multiplie pas les entités au-delà de la stricte nécessité"

comme Marvin Minsky dans son livre La Société de l'esprit [73], qui proposaient, au contraire, de faire coopérer des spécialistes détenant chacun une expertise restreinte, mais pas très spécifique.

1.3.1 Acteur

C'est Carl Hewitt [39] et son équipe du MIT, qui sont à l'origine de ce courant d'idées, et qui ont mis au point le premier modèle d'acteur. En fait un acteur est considéré comme un expert autonome, vivant en société, et communiquant avec d'autres experts pour résoudre des problèmes. Chacun des experts peut lui-même être une société d'experts plus élémentaires. Le modèle est donc distribué : les connaissances et le comportement sont diffusés parmi les acteurs, qui effectuent des tâches en parallèle et de manière indépendante. A la façon d'une classe, dans l'approche d'objet, un acteur regroupe au sein d'une même entité un ensemble limité de connaissances et le moyen de les exploiter : Des données locales appelées accointances, qui sont les autres acteurs que l'acteur connaît directement. Un comportement, défini par un script, qui décrit les actions à entreprendre par cet acteur au cours de son existence, lors de son interaction, par les messages, avec ses accointances.

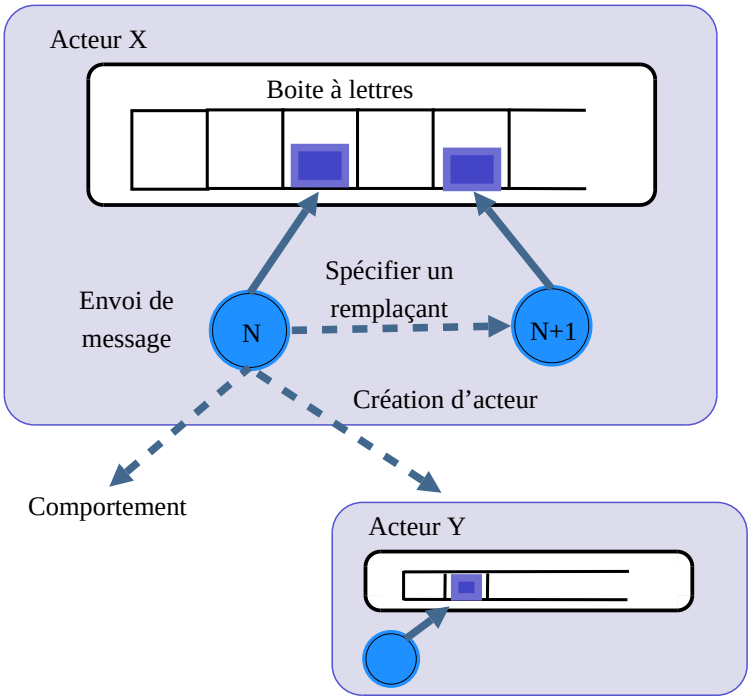

FIG. 1.1 – *Comportement d'un acteur lors du traitement du N^{me} message*

1.3.2 Les trois grands modèles Acteur

Trois grandes écoles ont apporté leur contribution à ce courant d'idées [71] :

I. Le modèle de Carl Hewitt
Ce modèle est à l'origine de cette approche. Il obéit pratiquement aux principes énoncés ci-dessus. Des langages ont tenté d'implémenter ces principes, comme PLASMA, mais qui ne sont pas sortis des laboratoires.

II. Le modèle de Gul Agha
Ce modèle se veut plus rigoureux par rapport à celui d'Hewitt. Il remplace le changement d'état d'un acteur par celui de son comportement. Un acteur pour Agha est une adresse et un comportement (Fig. 1.1).
Conceptuellement un acteur peut être vu comme une file de messages et une machine traitant un message particulier de cette file. Ce principe a été concrétisé dans des langages tels que ACT1 et ACT2.

III. Le modèle Hybride
Le deux modèles d'Hewitt et d'Agha sont des purs " acteur ", où tout est acteur. Cela rend leur implémentation difficile. Certains concepteurs ont donc choisi une approche plus pragmatique, leurs implémentations sont des extensions d'autres langages avec un esprit d'acteur. Le langage ABCL/1 de A. Yonesawa [98] est une extension de Common Lisp. OCCAM+ [7] fait parti aussi de cette tendance, car il se base sur le langage parallèle Occam2 comme éléments de syntaxe de base, à qui on a ajouté le principe d'acteur et de messagerie.

1.4 Agent

1.4.1 Acteur vs Agent

Après une évolution lente à travers les méandres de l'intelligence artificielle distribuée, le modèle acteur, devant évoluer. En effet, de nature purement distribué et s'apprêtant bien à la résolution des problèmes décomposables, en entités de simple traitement et réactives, et principalement basé sur le principe action / stimulus. Et vu qu'actuellement, les problèmes à traiter sont de natures complexes et variés : systèmes de production, e-commerce, robotique, etc. Ces entités acteurs sont appelées donc à jouer un rôle plus intelligent. C'est cette évolution qui a donné naissance à la notion d'agent. Nous considérons donc qu'un acteur n'est autre qu'un agent de traitement, dépourvu d'intelligence. Le fait de le doter d'une forme d'intelligence, le place dans un statut d'agent.

1.4.2 Acteur vs objet

Par ailleurs, l'avènement du paradigme SMA a coïncidé avec celui d'une nouvelle approche de programmation, appelée Programmation Orientée Objet (POO). Cette dernière est ancienne par son esprit, Simula en 1967 et Smaltalk en 1972, mais récente par sa

mise en oeuvre (C++, Java, C#, etc.). Cette coïncidence a provoqué un amalgame entre les deux notions agent et objet. Si la notion d'objet est liée à la façon d'implémenter, en encapsulant dans la même entité passive : les données et les méthodes qui agissent sur ces données. La notion d'agent elle, même si elle utilise dans la plupart des cas les objets lors de son implémentation, est une notion différente : C'est une entité conceptuelle liée à la manière d'analyser, de structurer et de mettre en oeuvre les traitements d'un problème complexe.

1.4.3 Qu'est qu'un Agent ?

Il n'existe pas de définition commune à toutes les tendances de recherche sur le concept d'agent. Néanmoins, et à notre humble avis, une définition doit être donnée, d'un point de vue de la distance de l'implémentabilité des concepts avancés. C'est-à-dire, du fait qu'on parle d'agent logiciel, nous devons être pragmatique et énoncer des définitions qui sont implémentables.

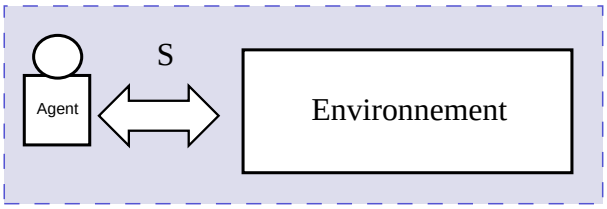

FIG. 1.2 – *L'agent et son environnement*

Sur ce, nous disons qu'un agent agt est une entité logicielle :

a) Qui est autonome, c'est-à-dire doté d'un comportement autonome et d'un état interne propre,

b) Qui est situé dans un système S, et capable d'agir dans un environnement : Env (agt) S - agt ; S étant l'ensemble des agents du Système.

c) Qui peut communiquer et éventuellement interagir avec ses accointances; c'est-à-dire les autres agents avec lesquels cet agent peut communiquer et interagir,

d) Une conséquence de a), il doit posséder des ressources propres, sous forme d'objets; cette idée n'est pas contradictoire avec le fait d'avoir des ressources partagées avec d'autres agents de son système, comme une librairie locale de classes, une imprimante, etc.

e) Qui est capable de percevoir, mais de manière limitée, son environnement ;

f) Possède un objectif à atteindre en conformité avec l'objectif commun de tout le système auquel il appartient.

Le critère d'autonomie implique évidemment que l'agent doit être capable de gérer les conflits éventuels entre ses objectifs et son état interne ou l'état de son environnement.

Les mécanismes permettant de sélectionner des actions et de résoudre les conflits font l'objet de nombreuses recherches, dont une partie est présentée par Ferber [28].

Elle distingue, par ailleurs, cinq catégories de motivations, c'est-à-dire des raisons qui poussent un agent à agir. Nous en retiendrons trois :

1. Les motivations personnelles, qui tendent à satisfaire les besoins et objectifs internes de l'agent ;
2. Les motivations environnementales, qui sont produites par ce que perçoit l'agent de son environnement ;
3. Les motivations sociales, qui découlent de l'organisation des agents imposée par le concepteur du système.

1.5 Environnement d'un agent

Il existe différents types d'environnements, le fait qu'un agent soit autonome, cela veut dire que ce dernier est capable d'agir dans cet environnement sans l'intervention d'autres agents ou d'un usager humain. Une classification a été faite par Russell [82] et donne les types suivants :

- Accessibilité : l'agent peut obtenir des informations complètes et à jour sur l'état de l'environnement; ceci ne s'applique pas sur le réseau Internet.
- Déterminisme : une action a un effet garanti dans le temps.
- Episodique: l'agent ne peut rien prédire sur l'évolution de l'environnement car les événements observés n'ont pas de corrélation entre eux et ne correspondent à aucune règle générale.
- Statique/Dynamique : l'environnement dynamique possède des processus qui agissent sur son état indépendamment du contrôle que l'agent a sur lui-même.
- Discret/Continu : un environnement discret possède un nombre fini et fixé d'actions et de perceptions sur lui.
  Pour Russel, l'environnement le plus complexe est inaccessible, non déterministe, non épisodique, dynamique et continue.

1.6 Types d'agents

1.6.1 Classification

Plusieurs classifications ont été avancées dans la littérature des agents, et sont essentiellement basées sur les deux critères d'intelligence et de mobilité.

selon le critère d'intelligence

La notion d'agent s'attache à tout problème complexe, d'où la nécessité de décomposer le problème en sous problèmes, la nature des problèmes à résoudre nous ramène à penser concevoir les agents comme des entités " intelligentes ", capables de résoudre le problème eux même, guidés par un ensemble de buts à satisfaire. A cet effet on peut citer deux grandes "écoles " de pensée :

L'école " cognitive ", est la plus représentée dans le domaine de l'Intelligence Artificielle Distribuée (IAD). Dans ce cadre, un SMA est composé d'un petit nombre d'agent " intelligents ". Chaque agent dispose d'une base de connaissance comprenant l'ensemble des informations et des expertises nécessaires à la réalisation de sa tâche et à la gestion des interactions avec les autres agents et avec son environnement.

On dit aussi que les agents sont " intentionnels ", c'est-à-dire qu'ils possèdent des buts et des plans explicites leur permettant d'accomplir leurs buts.

L'école réactive

L'autre tendance, l'école " réactive ", stipule que les agents ont un fonctionnement très simple, de type " stimulus / action ".

Un système contient un grand nombre d'agents réactifs dépourvus d'intelligence. Cette intelligence du système émerge à partir de l'interaction de ce grand nombre d'agents réactifs. Et donc un agent réactif : C'est un agent capable de réagir à des événements externes pour assurer soit la conduite et le pilotage d'un événement matériel dont il est la partie commande, soit la supervision ou la coordination des activités d'un ensemble d'agents dont il est l'agent de contrôle (partie de contrôle d'une application multi-agents).

Autres classifications

Il existe d'autres classifications que celles que nous avons énoncées, Attoui [5] propose une autre forme de classification:

1. Agent naturel; s'il est doté de possibilité d'auto-apprentissage,

2. agent réactif; s'il est capable de réagir à des événements externes, tel le pilotage d'un équipement matériel par exemple,

3. agent de traitement; si son rôle est d'effectuer des traitements et des calculs. Dans ce cas, un agent est un acteur, au sens déjà défini.

4. agent cognitif; s'il possède une expertise dans un domaine bien précis, qui ne contient pas forcément des données et des méthodes procédurales, mais une base de connaissances et qui peut, en outre, initier des inhérences sur sa base de connaissances et celles des autres de même type.

A notre avis, et en se basant sur le critère d'intelligence, les types 1. 2. et 3. ne sont que des cas particuliers d'agents réactifs.

Classification basée sur la mobilité

1. Agent Fixe est un agent réactif ou cognitif, situé sur un poste isolé, appartenant ou non à un réseau et ayant un objectif à atteindre sur la base d'exécution d'un script, en collaboration avec les autres agents du système. Un agent fixe est généralement facile à implémenter. Il est associé à thread ou un processus pour son activation, et il communique avec les autres agents de son environnement via des middlewares, bus Corba, Soap, etc.

2. Agent Mobile Lors de l'utilisation d'agents mobiles toutes les caractéristiques des agents fixes sont conservées (voir TAB.1.1). L'utilisation des agents mobiles présente en outre, plusieurs avantages :

   - équilibrage de charge en terme de processus de calcul : Un agent mobile peut en effet se déplacer sur un ordinateur plus puissant pour effectuer un calcul complexe. De même, il peut quitter une machine qui est saturé pour aller sur une autre,
   - équilibrage de charge en terme de traitement de données: Un agent qui a besoin de traiter une grande quantité de données situées sur un autre ordinateur, comme une base de données par exemple. Il peut se déplacer sur l'ordinateur possédant ces données et revenir avec le résultat. Cela permet d'éviter de faire transiter les données via le réseau.
   - Agents nomades, à la recherche d'une information sur réseau, comme dans l'e-commerce, l'e-maintenance, la fouille de données, etc

	Statique	Mobile
Données	Un fichier Excel	Un document HTML
Code	Code du programme Excel	Applet Java
Objet	Un objet manipulé par un processus Excel	Un migrating objet Corba
Processus	Un processus exécutant Excel	Un processus Sprite
Agent	Un assistant personnel	Un aglet implémentant un assistant personnel

TAB. 1.1 – *A l'exception de ma mobilité, les entités informatiques statique et mobile sont similaires.*

Les agents mobiles présentent des intérêts évidents, mais sont cependant plus complexes à gérer. Ils sont utilisés essentiellement sur les réseaux et en particulier sur Internet où le nombre de sites n'est pas connu d'avance et où le problème et non déterministe.

1.7 Les Systèmes Multi-agents

Les SMA proposent une nouvelle approche de modélisation des connaissances. ils se situent comme un prolongement et une extension de la notion d'objet dans le cas distribué. En mettant l'accent sur l'interaction et la satisfaction individuelle. ils s'interdisent de penser le global centralisé. Dans un SMA, tout est distribué, répartis : le contrôle, la connaissance, les compétences, l'activité, la planification, etc.

De ce fait les SMA s'adaptent bien aux systèmes complexes et ouverts où il est difficile de tout décrire à l'avance. J. Ferber [28] donne la définition formelle suivante :

On appelle SMA, un système composé des éléments suivants :

- Un environnement Env, c'est-à-dire, un espace disposant généralement d'une métrique ;
- Un ensemble d'objets O, ces objets sont situés, c'est-à-dire que pour tout objet, il est possible, à un moment donné, d'associer une position dans Env. les objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, crées, détruits et modifiés par les agents ;
- Un ensemble A d'agents, qui sont des objets particuliers, lesquels représentent 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 A 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.

Il existe un cas particulier de systèmes dans lequel A = O, et Env(S) est un ensemble vide. Dans ce cas, les relations R définissent un réseau : chaque agent est lié directement à un ensemble, d'autres agents que l'on appelle accointances. Ces systèmes, que l'on peut appeler SMA purement communicants, sont très courants en IAD.

1.8 SMA et Systèmes Complexes

Les contraintes de distribution liées aux systèmes complexes sont à la fois physiques et organisationnelles. Au niveau pratique du contrôle, elles se traduisent par des caractéristiques fondamentales pour la bonne marche du système. En effet, elles agissent sur plusieurs fonctions :

- La décision : Chaque entité conserve son autonomie de contrôle, et par conséquent son autonomie de décision.
- L'accès aux informations : les partenaires dans une entreprise souhaitent coopérer, mais d'une façon limitée. Chacun possède donc son propre état et ses connaissances. Les systèmes informatiques de planification et d'ordonnancement sont hétérogènes et ne doivent pas être remis en cause, par la participation temporaire à une entreprise virtuelle ou par un changement de fournisseur par exemple.
- La communication : les parties du système devront se trouver sur des plates-formes différentes, en différents endroits. Tout mécanisme de coordination nécessite des communications, via des canaux dédiés, entre les éléments du système.

Traditionnellement, dans la résolution des problèmes distribués, l'approche utilisée (Fig. 1.3), consiste à réduire le problème posé en sous-problèmes afin de simplifier les tâches de calcul. La définition d'une tâche globale n'est pas généralement triviale. Les traitements à exécuter en parallèle pour résoudre les sous-problèmes sont assez complexes et coûteux en temps de calcul. Cette démarche classique ne garantit pas un processus

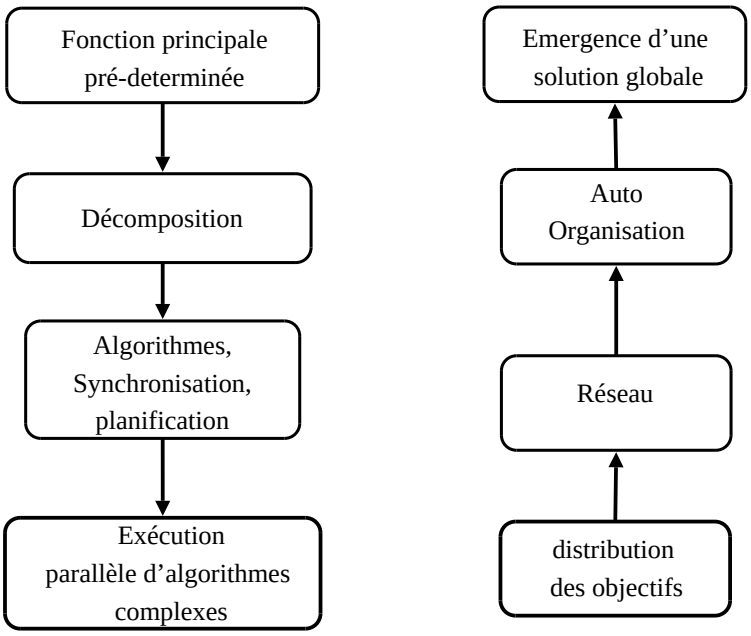

FIG. 1.3 – *L'émergence de la solution globale*

d'ordonnancement suffisamment réactif si la dynamique est importante. Divers travaux privilégient la méthode inverse de résolution, appelée généralement bottom-up.

L'émergence d'une solution globale de l'objectif assigné à tout le système (Fig. 1.3.), à partir de solutions locales, est la raison d'être des SMA. Cette idée est une inspiration de la société de fourmis et leur auto-organisation pour la recherche de la solution globale. Ce courant d'idée représente actuellement un axe de recherche, dans les SMA et l'auto-organisation d'agents qui en font partis, appelée Ant-Systems.

1.9 Architectures des SMA

On retrouve dans une architecture d'agent les paramètres généraux valables pour l'analyse structurale des organisations artificielles : le type d'approche, de composants, de structures de subordination, de couplages et de constitutions. Parmi l'ensemble des architectures possibles obtenues en faisant varier ces paramètres, seul un petit nombre a donné lieu à suffisamment d'implémentation pour pouvoir être catégorisé. Il s'agit des architectures à base de modules horizontaux, de tableau noir, de subsomption, de système dynamique et de multi-agents et les architectures connexionnistes. On pourra noter que tous les types d'architectures emploient une approche fonctionnelle, qu'elle soit horizontale ou verticale.

1.9.1 Architecture modulaire horizontale

Ce type d'architecture est certainement l'un des plus répandus, qu'il s'agisse de travaux théoriques ou d'applications pratiques, la plupart des architectures proposées pour la définition d'agents cognitifs, sont fondées sur la notion d'ensemble de modules horizontaux liés par des connexions préétablies.

Les architectures modulaires horizontales, que nous appellerons modulaires tous simplement par souci de simplification, sont conçues comme un assemblage de modules, chacun réalisant une fonction horizontale particulière.

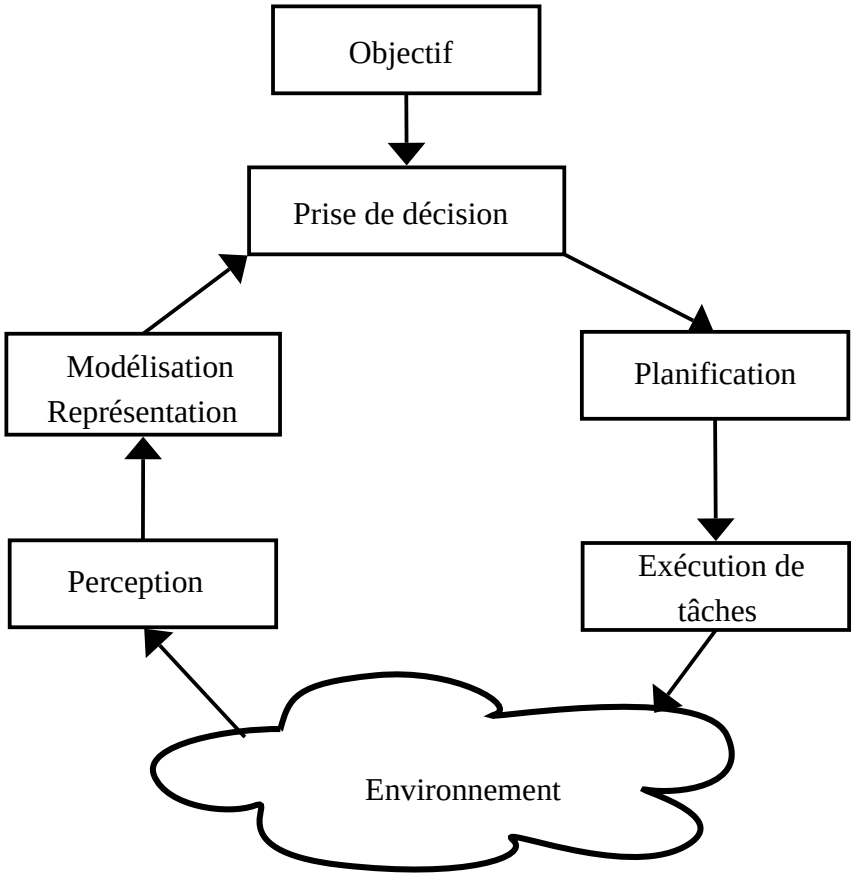

FIG. 1.4 – *Architecture Modulaire horizontale*

Les modules les plus courants sont :

- Les fonctions perceptives et motrices, s'il y a lieu.
- L'émission et l'interprétation des communications.
- La base de croyance comprenant la modélisation de l'environnement et des autres agents.
- La gestion des engagements.
- Les expertises du domaine de compétence.

- La gestion des buts et la prise de décision.
- La planification des actions,
- etc.

Dans ce type d'architecture, toutes les liaisons sont fixes. C'est-à-dire que le mode de circulation des informations est pré-défini par le concepteur. La figure 1.4 montre un exemple caractéristique d'une telle architecture, caractérisé par un flux d'information montant puis descendant.

Dans la phase ascendante, les signaux provenant de l'extérieur par l'intermédiaire de capteurs ou des boites aux lettres comprenant des messages sont filtrés de manière à obtenir une information de nature de plus en plus abstraite, jusqu'à ce qu'elle puisse s'intégrer aux modélisations de l'agent.

La fonction la plus élevée est effectuée par le module de la prise en décisions qui, à partir des informations qu'il reçoit et des objectifs qui lui sont propre, décide d'agir.

On passe ensuite à la phase descendante correspondant à la mise en application des décisions, le module de planification ordonne les actions à effectuer pour satisfaire l'objectif choisi, celles-ci sont ensuite transmises au module d'exécution.

1.9.2 Les architectures à base de tableau noir

L'architecture de tableau noir a été l'une des plus utilisé dans les SMA cognitifs symboliques, elle s'est rapidement imposée en IAD comme une architecture suffisamment souple et puissante pour pouvoir implémenter les mécanismes de raisonnement et de calculs intervenant à l'intérieur des agents.

Le modèle de tableau noir est fondé sur un découpage en modules indépendants qui ne communiquent aucune information directement, mais interagissent indirectement en partageant des informations.

Ces modules appelés sources de connaissances ou KS (Knowledge Sources), travaillent sur un espace qui comprend tous les éléments nécessaires à la résolution d'un problème.

L'architecture d'un SMA à base de tableau noir comprend trois sous-systèmes :

- Les sources de connaissance.
- La base partagée (Tableau) qui comprend les états partiels d'un problème en cours de résolution, les hypothèses et les résultats intermédiaires et toutes les informations que s'échangent les KS.
- Un dispositif de contrôle qui gère les conflits d'accès entre les KS.

Dans un premier temps les systèmes à base de tableau noir furent considérés comme des systèmes d'IAD, chaque KS pouvant être perçu comme un agent qui interagit avec les autres KS, il n'en est plus de même aujourd'hui, du fait de leurs mécanismes de contrôle très centralisés et de leur manque de mémoire locale et donc de localité des informations, ces systèmes sont maintenant envisagés comme des architectures pratiques pour la réalisation de système " intelligents " et en particulier pour implémenter la structure interne d'agent cognitif symboliques.

L'architecture de tableau noir présente de nombreux avantages dont, en tout premier lieu, une remarquable souplesse, pour décrire des modules et articuler leur fonctionnement. Son principal inconvénient provient de sa relative inefficacité, due à la très grande expressivité de son contrôle.

De ce fait, ce type d'architecture s'avère particulièrement utile lors de la phase de prototypage de la réalisation de systèmes ou lorsque les temps de réponses ne sont pas trop contraints.

1.9.3 L'architecture de subsomption

L'architecture de subsomption a été proposée pour la première fois par R-Brooks pour la constitution d'agents tropiques réactifs.

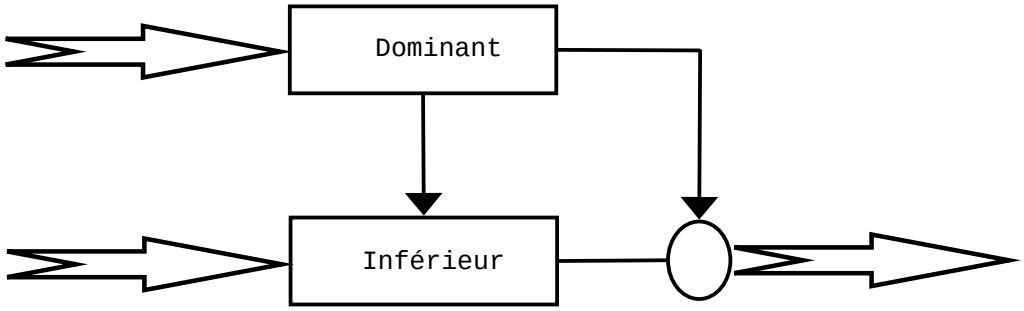

FIG. 1.5 – *Dans une architecture de subsomption, les modules supérieurs sont dominants et inhibent, si nécessaire, la sortie des modules inférieurs.*

A l'inverse de l'architecture modulaire hiérarchique qui divise un agent en modules horizontaux, l'architecture de subsomption décompose un agent en modules verticaux, chacun d'eux n'étant responsable que d'un type de comportement très limité.

Les interactions entre les modules sont fixes et s'effectuent par l'intermédiaire d'un rapport de dominance défini lors de la conception.

Cette technique a surtout été utilisée pour décrire des agents réactifs, mais il serait possible de l'utiliser pour des agents cognitifs, en considérant que les modules supérieurs sont les plus réflexes et les modules inférieurs sont les plus cognitifs, en induisant une priorité dans les modules.

1.9.4 Les architectures connexionnistes

Les architectures connexionnistes, qui sont fondés sur la métaphore du cerveau, sont formées d'un réseau d'éléments tous identiques que l'on appelle souvent " neurones formels ", à cause de leur ressemblance (assez vague en fait) avec les neurones du système nerveux des animaux.

La fonction de transfert du neurone est donnée par l'équation : $Y_j = f(\sum(W_{ij}X_i)$

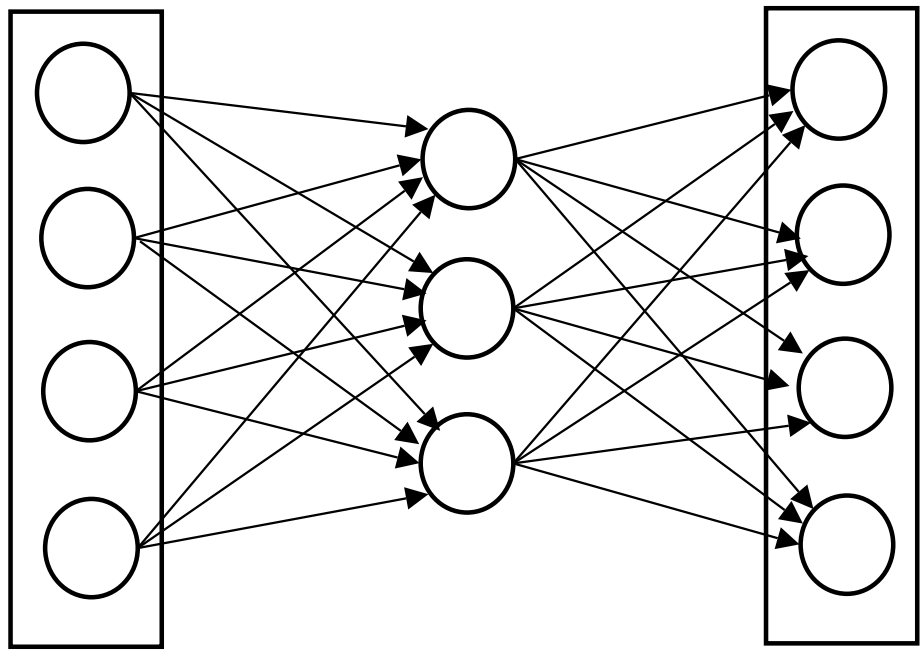

FIG. 1.6 – *Un réseau de neuronnes à trois couches comprenant une couche interne (ou cachée)*

Ou les X_i sont les valeurs de sortie des unités i, les W_{ij} sont les poids des connections reliant les neurones i aux neurones j, et f est la fonction d'activité.

Il existe un grand nombre de modèles de réseaux de neurones. Les plus classiques sont les réseaux à couches.

Ils supposent que les neurones sont regroupés en ensembles, appelés couches, et que les entrées des neurones de la couche n sont reliées aux sorties de la couche $(n-1)$.

1.9.5 Les architectures à base de système dynamique

Dire qu'il existe des architectures dynamiques est peu présomptueux pour l'instant puisqu'il n'existe qu'une seule version d'architecture de ce type, celle qu'a développé L. Steels pour la réalisation de robots autonomes coopératifs.

Le principe consiste précisément à faire table rase de la notion d'architecture et à simplement implémenter le comportement d'un agent en décrivant directement les équations qui relient les valeurs (Standardisées pour des raisons d'efficacité) des capteurs à celle des commandes appliquées aux effecteurs.

Évidemment, cette approche est très liée au matériel utilisé puisqu'il n'existe pas de représentations intermédiaires à partir desquelles il serait possible d'abstraire un com-

portement indépendant.

Néanmoins, et bien que ce type d'architecture en soit encore à ses premiers débuts, on peut gager qu'il recueillera rapidement de nombreux adeptes, du fait de sa simplicité et de sa capacité à pouvoir effectuer de manière naturelle plusieurs actions en même temps.

1.9.6 Architectures SMA et acteurs

On appelle architecture SMA d'un agent l'application de la notion de Système SMA à la définition de l'architecture des agents eux-mêmes.

Un agent étant alors considéré comme système SMA à part entière. Les premières à avoir considéré le psychisme d'un être (humain ou artificiel) comme le résultat d'interaction entre petits agents individuels est M. Minsky [73].

En particulier tous les travaux portant sur les langages d'acteurs et sur leur utilisation dans un système SMA peuvent, dans une certaine mesure, se rapporter à ce courant.

Initialement les acteurs ont surtout été étudiés comme des modèles d'exécution pour la programmation par objets concurrents.

Mais quelques travaux ont voulu rester dans les idées initiales que prônaient Hewitt et qu'il confirma avec ses notions de "sémantique des systèmes ouverts ", tout en estimant que les langages d'acteurs sont effectivement de très bons outils pour l'implémentation de calculs parallèles, considèrent néanmoins qu'ils présentent des caractéristiques tellement originales qu'il modifient par leur présence, la notion même d'architecture multi-agents en envisageant les agents et les SMA comme des extensions naturelles de la notion d'acteur.

En étendant la notion d'acteur vers celle d'agents grâce à l'utilisation de la réflexivité organisationnelle, Giro [75] propose de considérer un agent comme un écosystème, c'est-à-dire comme un ensemble d'agents en interactions, et un écosystème comme un agent. Son système Reactalk, fondé sur une extension d'Actalk, fournit aux agents la possibilité d'adapter leur comportement en fonction d'un type de message qu'ils reçoivent, et ainsi de pouvoir dialoguer aussi en mode synchrone si cela s'avère nécessaire.

Néanmoins, ces agents ne disposent pas véritablement de comportements autonomes (ils n'ont pas de systèmes motivationnels) ni de représentations de leur environnement, l'idée d'écosystème étant plutôt prise comme une métaphore.

Ces travaux montrent qu'il existe un lien de continuité entre la notion d'acteur et celle d'agents. Dans [30], il avait proposé un système Paladin, écrit comme une extension du langage Mering IV dans lequel les acteurs pouvaient Poursuivre un but et, si nécessaire, raisonner sur leurs propres compétences et expliquer leur comportement à d'autres entités du système.

Les acteurs avaient véritablement pris le statut d'agents, la réflexivité étant ici encore le bien nécessaire au passage d'un niveau "agent" ou les entités disposent d'une plus grande latitude comportementale, voire de capacités de raisonnement ou de motivations [75].

1.10 La communication dans les SMA

Tout d'abord, les agents doivent être capables, par le biais de la communication, de transmettre des informations, mais surtout d'induire chez l'autre un comportement spécifique.

Communiquer est donc une forme d'action particulière qui, au lieu de s'appliquer à la transformation de l'environnement, tend à une modification de l'état mental du destinataire.

Par exemple, demander à un autre d'exécuter une tâche, tend à provoquer chez l'autre une intention d'accomplir cette tâche et constituent donc une manière de satisfaire un objectif sans réaliser la tâche soi-même.

1.10.1 La communication par tableaux noirs

a) Le Modèle tableau noir:

C'est une structure de donnée renfermant des informations partagées par des modules hétérogènes du système, ces informations sont des objets.
Les objets de même nature sont rassemblés en niveaux hiérarchiques, l'ensemble des niveaux est généralement ordonné de telle sorte à fournir un ordre d'abstraction hiérarchique entre ces niveaux.

- Les objets : représentent le contenu du tableau noir.
- Les modules (Les agents) : dans un système tableau noir, un module contient un ensemble de connaissances homogènes entre elles, représentées dans un même formalisme.
- La communication entre module se fait indirectement par le biais du partage des mêmes informations dans le tableau. Les modules sont considérés comme des variables.
- L'architecture tableau noir est un modèle général de résolution de problèmes ou les connaissances sont regroupées en modules coopérants entre eux.
- Le tableau noir est caractérisé par un ensemble d'états qui sont des objets accessibles à tous les modules.
- La seule activité des modules consiste à lire, écrire et modifier des objets dans le tableau. Les modules ont une structure de type condition/action.
- Chaque modification du tableau, engendré par un module, induit un nouvel état du tableau qui correspond à une solution partielle.

Contrôle de tableaux noirs :

Dès qu'une configuration d'objets satisfait la partie condition du modèle, il y a création d'objets dans le tableau noir, le module ignore l'activité des autres modules ainsi que les autres objets du tableau.

- La structure tableau noir contient la solution courante du problème, chaque nouvelle configuration du tableau est une solution partielle du problème.

- Un module se déclenche dès qu'une configuration du tableau permet d'exécuter une action.
- La solution est construite de manière incrémentale.

Contrôle séquentiel :

Les modules d'un système tableau noir ont une structure de la forme condition/action similaire aux règles des systèmes experts classiques.
Quand la partie condition d'un module est satisfaite il y a création d'un KSAR (acronyme de Knowledge Source Activation Record).
Un KSAR est un couple (module, liste d'objets), l'exécution d'un KSAR est une possibilité d'évolution du système, elle implique l'application du module sur la liste d'objet correspondant. Le déclenchement d'un KSAR provoque la création d'un ou de plusieurs événements.
Un événement peut être présenté par une structure de donnée de contrôle enregistrant les changements opérés dans le tableau noir.
Un événement résulte de l'exécution d'une action. Une action consiste à :

- Créer de nouveaux objets.
- A en supprimer.
- Ou encore à modifier des objets du tableau.

Le déclenchement d'un ksar permet donc de constituer de nouvelles configurations d'objets permettant le déclenchement de certains modules du système.
Le cycle de base du système est le suivant :

- Les modules créent des KSARs qu'ils stockent dans une structure de donnée appelée agenda.
- Le contrôleur du système choisit un KSAR dans l'agenda.
- Le déclenchement du KSAR choisi, engendre l'exécution de l'action correspondante qui se traduit par la création, la suppression ou la modification d'objet.
- Les multiples implémentations du modèle tableau noir proposent un contrôle particulier de telle sorte, qu'à un instant donné, soit choisi le meilleur KSAR en fonction de l'état courant du tableau.

Donc on peut dire que le tableau noir est un modèle de résolution de problèmes ou la connaissance est structurée en modules indépendants communicants par le biais d'un mécanisme de lecture/écriture d'objets sur le tableau.

b) La communication par réseaux contractuels :

Le réseau contractuel (RC) est un exemple de système de coopération. C'est une collection de noeuds qui coopèrent pour la réalisation d'un but donné.
Ces noeuds sont en principe des objets ou de simples entités capables d'agir. Ils se transforment en agents dès qu'ils adoptent un but.
Les noeuds du RC comme tout réseau communiquent entre eux. Nous pouvons imaginer qu'ils sont tous connectés à un réseau de typologie quelconque (Fig. 1.7).

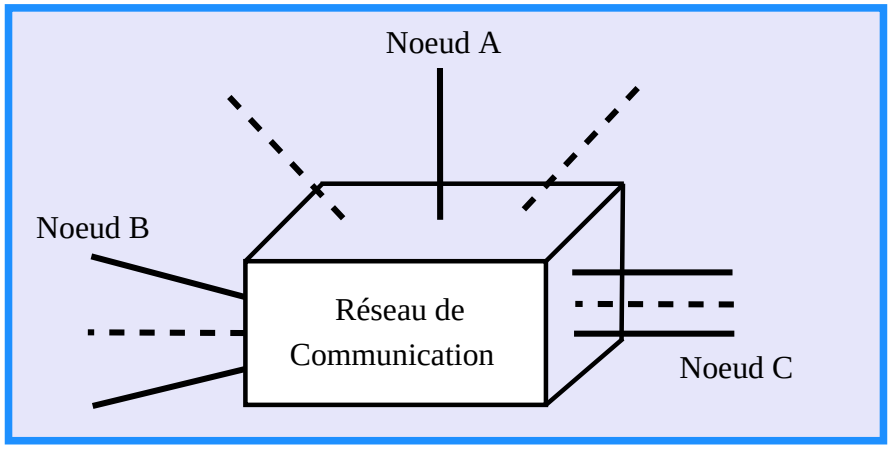

FIG. 1.7 – *Le Réseau Contractuel*

1.10.2 Langages de communication dans les SMA

Dans un système complexe hétérogène où collaborent des agents écrits dans différents langages et sous différentes plate-formes logicielles, la nécessité de définir un langage de communication est évidente. Ce langage doit cependant aussi bien définir le fond que la forme des échanges.

Ainsi lorsqu'un objet reçoit une information, il doit savoir de quoi il s'agit et éventuellement comment la traiter.

La définition d'un langage de communication permet, en outre, de faire coopérer facilement plusieurs versions d'un même logiciel : Une fois le langage défini, une partie du logiciel peut évoluer sans que les autres parties n'ait à évoluer de leur côté.

Ceci suppose que dès le départ, le langage de communication choisi soit le plus complet possible, mais aussi et surtout extensible. C'est en effet grâce à cela que l'évolution et l'interopérabilité des différentes briques du logiciel peuvent être garanties.

- Le langage XML et les SMA :
  XML est un langage qui permet de décrire les structures de données. Ceci permet donc ce langage d'être un bon candidat pour l'échange de données entre les différents agents du système.
  Plusieurs solutions se présentent tout de même pour l'échange de données avec XML : Les données que l'on échange n'ont pas une structure fixe (a priori), il faut donc :

  - Soit décrire la structure des données dans le document XML puis construire un objet qui puisse accueillir ces informations,
  - Soit définir une bibliothèque d'objets qui sont susceptibles d'êtres échangés, et se référer à cette bibliothèque lorsque l'on échange des informations.

  Les deux solutions semblent envisageables et présentent chacune ses avantages et ses inconvénients.

Il faut noter que l'utilisation des langages différents entraîne d'autres problèmes, notamment sur les types de base : un entier n'a pas la même plage de valeur dans tous les langages, etc.

- Langages ACL / KIF / KQML :
  KQML (Knowledge Query Modeling Langage) est un langage qui permet de définir un moyen de communication uniforme entre les divers acteurs du SMA. Ce langage, contrairement à XML décrit la forme dont les données sont échangées (du pseudo-lisp) mais aussi le fond, c'est à dire les informations à mettre dans le message.
  Même si l'on utilise XML pour la forme, pour le contenu du message il peut être intéressant de regarder ce qui a été fait avec KQML : Le nom de l'expéditeur, du destinataire, le sujet du message, l'intention du message et / ou le type du message, la priorité du message, la date d'émission, la durée de validité (dans le cas d'une application où la notion de temps est importante), le type du contenu du message, le contenu du message, etc.

1.11 Les Modèles d'organisation

Un modèle d'organisation doit répondre aux critères sur lesquels on distribue les rôles et les tâches aux agents, et dans quels environnements. Ce modèle doit mettre en oeuvre un modèle d'exécution, lors de l'implémentation, et doit répondre, par conséquent, à une architecture d'organisation du SMA.

Plusieurs courants d'idées ont tenté de mettre en oeuvre les concepts de base des modèles d'organisation dans des plates-formes dédiées.

Dans ce qui suit, nous présentons l'existant en matière de plates-formes, et essayerons de répondre à la question : Peut-on avoir un modèle d'organisation générique qui répond à n'importe quel type de problème complexe?

1.12 La Programmation Orientée Agent

La programmation orientée agents (POA) a été proposée par Yoav Shoham en 1993 comme un nouveau paradigme de programmation, que l'on peut voir comme une spécialisation de la programmation orientée objets. Dans cette approche, les agents sont les éléments centraux du langage, de la même façon que les objets sont centraux pour les langages orientés objets. La perspective sur les agents est cognitive : les agents sont caractérisés par des notions mentales comme leurs croyances, leurs décisions et leurs obligations. De plus, à chaque agent est associé un ensemble d'habiletés qui représentent ce que l'agent sait faire. En même temps, la programmation orientée agents suppose qu'on va développer des programmes dans lesquels plusieurs agents interagissent, ce qui met l'accent sur la dimension sociale des agents. Le langage de programmation proposé par Shoham comme démonstration de ce nouveau paradigme s'appelle AGENT0.

La différence principale entre un tel langage et un langage de programmation classique que l'on pourrait utiliser pour développer des agents, vient du fait que les notions mentales qui caractérisent les agents apparaissent dans le langage lui-même, et que la

sémantique du langage est intimement liée à la sémantique de ces notions mentales. La programmation orientée agents peut être vue comme une spécialisation de celle orientée objets parce que les modules du programme sont maintenant des agents, c'est-à-dire des objets avec un état qui définit les notions mentales associées, et que les messages entre objets sont remplacés par des messages entre agents.

En quoi les messages entre agents différent-ils de ceux entre objets ?

- Premièrement, parce que ces messages sont modélisés en partant de la théorie des actes de parole, qui s'intéresse aux actions de communication comme informer, demander, offrir, accepter, rejeter et d'autres.
- Deuxièmement, puisque les agents sont autonomes et dotés de capacités mentales, ils ont la liberté de décider s'ils vont ou non exécuter l'action spécifiée dans le message.
- Par contraste, un objet recevant un message va toujours exécuter l'action spécifiée dans le message. Cette spécialisation même fait que la programmation orientée agents est différente de celle orientée objets comme le montre le tableau 1.2, tableau de différences établi par Shoham.

	P O O	P O A
Unité de base	objet	agent
Paramètres définissant l'état de l'unité de base	pas de contraintes	croyances, décisions, obligations, habiletés
Processus de calcul	envoi de messages et méthodes pour la réponse	envoi de messages et méthodes pour la réponse
Types de messages	pas de contraintes	informer, demander, offrir, promettre, accepter, rejeter, etc.
Contraintes sur les méthodes	pas de contraintes	consistance, vérité, etc.

TAB. 1.2 – *Programmation Orientée Objets versus Programmation Orientée Agents.*

Avec l'évolution des théories concernant les agents et des langages de programmation associés, on peut également souligner d'autres différences entre POO et POA :

- Les agents sont autonomes alors que les objets ne le sont pas ;
- un agent va décider par son propre processus de décision s'il exécute ou non une action requise ;
- Les agents ont leurs propres buts et ils agissent d'une manière pro-active pour atteindre leurs buts (par exemple, ils saisissent des opportunités) alors que les objets ne le font pas ;
- Les agents sont capables d'un comportement social : ils peuvent s'engager dans des interactions complexes, par exemple coopération, compétition, négociation, avec d'autres agents, ce n'est pas le cas des objets ;

- Un système multi-agents est, normalement, un système dans lequel les agents correspondent à des chemins d'exécution séparés ;
- Chaque agent a son propre chemin d'exécution alors que les objets, à part les objets concurrents, ne présentent pas cette caractéristique.

Le débat sur la différence entre agents et objets est toujours ouvert. Pour notre part, nous parlerons beaucoup plus de modélisation par les SMA, au niveau conceptuel que de programmation orientée Agents à bas niveau.

1.13 Les plate-formes SMA

1.13.1 Introduction

Une plate-forme multi-agents est un outil permettant de faciliter la construction et l'exploitation d'un système multi-agents. Elle peut prendre différentes formes, allant d'outils d'ordre méthodologique, à des outils de développement, ou des supports d'exécution. Ce qui importe, c'est qu'à un moment ou un autre, la plate-forme simplifie la tâche du développeur. Plus le développeur est assisté dans ses différentes tâches, plus la plate-forme est performante. Ainsi, une plate-forme idéale doit-elle assister toutes les tâches de la spécification à l'exécution du système multi-agents. Un autre aspect important d'une plate-forme est le domaine d'application dans laquelle elle est pertinente. En effet, un outil est fait pour servir à une tâche donnée, dans un domaine donné (par analogie, la tâche servie par un tire-bouchon est d'ouvrir une bouteille. Le domaine d'application du tire-bouchon est que le bouchon doit être en liège, et la bouteille en verre). Hors du domaine, l'outil est inadapté (on n'utilise pas un tire-bouchon pour ouvrir une bouteille de lait). Idéalement, la gamme de domaines que peut couvrir un outil doit être la plus large possible, même si, généralement, il est rare de pouvoir être générique dans tous les domaines (Le problème de l'ouverture des bouteilles en est un bon exemple). Ainsi, une plate-forme multi-agents doit éviter au maximum de limiter le domaine pour lequel elle est adaptée. Une méthodologie multi-agents doit être capable de couvrir le plus de cas possibles, un outil de développement multi-agents doit pouvoir implémenter le plus de systèmes multi-agents possibles, et le support d'exécution doit pouvoir supporter le plus de plates-formes logicielles et matérielles possibles.

Depuis plusieurs années, on a vu se multiplier les plates-formes de développement agents. Sous ce terme illusoirement générique se cachent plusieurs grandes catégories de plates-formes. Même s'il n'est pas question de voir ici en détail l'ensemble de ces plates-formes, ou même de ces catégories, nous allons en parcourir quelques exemples et noter les traits marquants.

1.13.2 Caractéristiques d'une plate-forme

1. Fonctionnalités La plate-forme devra être aussi générique que possible, et elle se limitera à mettre en oeuvre uniquement les fonctionnalités requises par une simulation.

2. Montée en charge
   Cette deuxième restriction d'ordre technique en entraînera d'autres, mais permettra aussi un certain nombre d'optimisations dans la mise en oeuvre de la plate-forme, et ce point est caractérisé par d'autre sous points :

   Performance : La plate-forme doit pouvoir faire fonctionner un nombre élevé d'agents sans que cela n'entraîne de dysfonctionnements ni de lenteurs.

   Simplicité : La programmation d'agents dans ce cas doit rester aussi simple et naturelle que possible et ne nécessite pas d'adaptation ni de reformatage.

   Documentation : L'interface de programmation de la plate-forme devra être bien documentée. Pour cela, il faut utiliser un générateur de documentation pour produire la documentation dans différents formats.

   Liberté : Les bibliothèques doivent être libres. C'est aussi une des raisons qui poussent les concepteurs de plate-formes à utiliser des bibliothèques de base en J2EE ou en .Net

1.13.3 Quelques plates-formes

Ils existent plusieurs plates-formes de développement de SMA:

1. La plate-forme PHOENIX :
   Est réalisée en java2, elle propose une API ouverte sur d'autres plates-formes utilisant les mêmes standards. Elle permet également le développement de différentes catégories d'agent, son architecture s'articule sur un noyau gérant le quadruple (Agent, Compétence, Rôle, Tâche).
2. La plate-forme MadKit :
   Est une plate-forme de développement de SMA fonctionnant en Java sur plusieurs plates-formes.
3. La plate-forme SWARM :
   est une plate-forme générique constituée d'une bibliothèque logicielle (en objective C, avec une interface JAVA) qui permet de développer des simulations à base d'agents, avec comme objectif: "expérimentation informatique" décentralisées à évènement discret de systèmes complexes.
4. La plate-forme DARX:
   permet la tolérance aux fautes grâce à la réplication dynamique des classes génériques pour l'encapsulation d'agents. Une instance de serveur chaque localisation réalisé en Java/RMI.
5. La plate-forme DIMA:

   - Environnement de développement de SMA.
   - Agents proactifs, associés à un moteur d'exécution.
   - Décomposition modulaire des agents.
   - Librairies fournissant un large éventail de classes d'agents.
   - Ensemble de briques logicielles pour construire des modèles d'agents divers.

1.13.4 Exemple de plate-forme: La plate-forme MadKit

Madkit (Multi-Agent Development Kit) [30] est une plate-forme conçu sous la direction de Ferber et Gutknecht , à partir de 1997. Elle est basée sur les concepts AGR : d'agents, de groupe et de rôle, ainsi que trois grands principes d'architecture:

- Micro noyau agent.
- Agentification des services.
- Modèle graphique componentiel.

La philosophie de base d'Aalaadin / Madkit est d'utiliser autant que possible la plate-forme pour son propre fonctionnement. Tout service non fourni par le noyau est confié à des agents spécialisés, structuré en groupes et identifiés par des rôles. Madkit est développé en Java 1.1 et testé sous Unix, Windows et MarcOS.

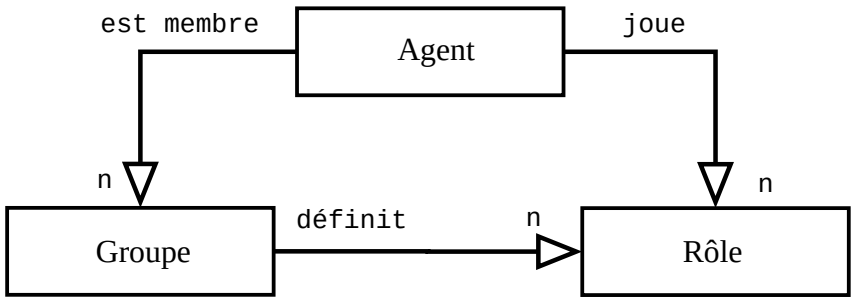

FIG. 1.8 – *Modèle d'organistion d'Aalaadin.*

Le modèle AGR, à partir duquel est construit MadKit est fondé sur une interaction entre groupes, rôles et agents.

MadKit fonctionne de manière distribuée selon un mode peer to peer. Il introduit aussi la notion de communauté, c'est à dire d'application distribuée à laquelle un ensemble d'agents peuvent se réunir et partager des applications et des documents.

Le mécanisme de communication présent en Madkit utilise les sockets de manière très simple (communication via TCP et UDP). On voudrait pouvoir optimiser cette technique de communication en offrant un véritable mécanisme peer to peer qui sache bien prendre en compte un grand nombre de sites et de modes de communication.

Il faut signaler que les concepts et influences qui ont donné naissance à la vision Madkit sont essentiellement ceux des concepts du modèle acteur (ActTalk, Mering IV, etc.).

1.13.5 Les plates-formes d'agents mobiles

Elles se situent souvent dans la filiation du précurseur Telescript, et présentent parfois une confusion entre "agent mobile" et "code mobile".

Un exemple de plate-forme d'agents mobiles est la plate-forme Aglets développée par IBM Japon, On y retrouve les mécanismes de base de la migration : un identifiant unique,

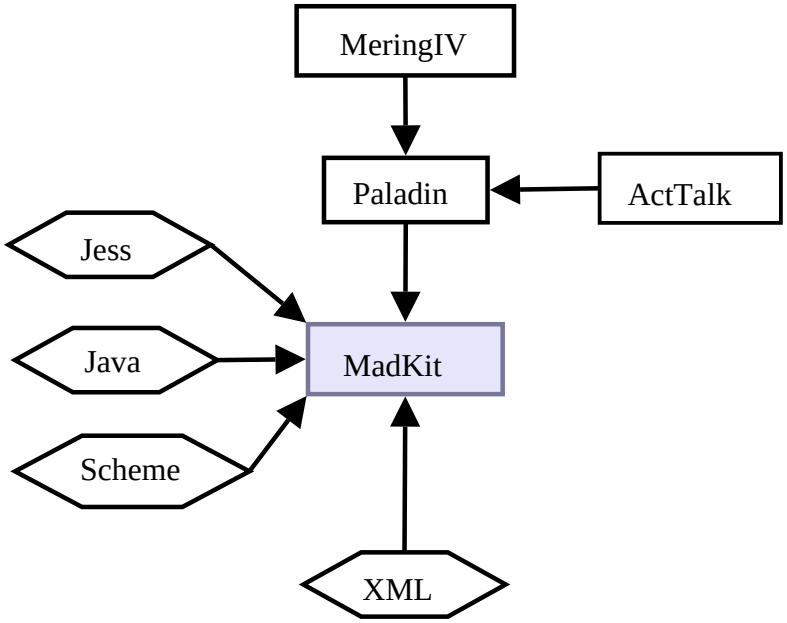

FIG. 1.9 – *Aalaadin: Origines et influences.*

un itinéraire qui décrit les différents noeuds à visiter et les actions à prendre à chaque étape.

Cela impose évidemment la mise en place d'une plate-forme aglet sur les noeuds désirés du réseau, car la migration n'est possible que si l'infrastructure est présente. Un outil de conception permet de spécifier les contraintes de sécurité à appliquer à chaque anglet sur chaque site. La communication entre aglets sur les sites se fait soit par un tableau noir, soit par passage de messages synchrones ou asynchrones.

Il faut bien dire que les fonctionnalités que propose Aglets sont plus à rattacher au domaine des objets mobiles que véritablement des agents. En effet, les aspects d'autonomie d'action ou d'interaction sont réduits à leur plus simple expression.

1.13.6 Les outils de simulation multi-agents

Ce sont souvent des outils construits de façon ad hoc pour un domaine de simulation, ou même une seule application.

Dans ce cas, l'outil propose un modèle suffisamment générique de simulation, basé sur une décomposition récursive des systèmes, sur lequel on peut bâtir éventuellement des modèles spécifiques. En fait, la force de cette plate-forme réside plus dans l'importance des outils annexes : obtention des données par des sondes que l'on peut positionner sur quasiment tous les points de l'application, traitement et première analyse par des outils statistiques visualisés en temps réel des paramètres, lancement de jeux d'expériences.

Le modèle d'agent est peut-être paradoxalement la partie la plus faible de l'outil.

1.13.7 Les plate-formes orientées modèle

Sous ce qualificatif se rangent des plates-formes à vocation généraliste du point de vue applicative, mais développées par rapport à un modèle très spécifique d'agent dont elles permettent en fait la validation pratique.

AgentBuilder appartient à cette catégorie, et réunit à la fois un outil de conception et une plate-forme d'exécution. Le modèle des agents AgentBuilder dérive du modèle Agent [84], la communication s'effectue par messages KQML.

L'architecture laisse néanmoins le concepteur libre de rajouter de nouveaux actes de langages pour s'adapter à un domaine particulier. L'outil de conception est extrêmement ciblé sur le type de modèle défini et couvre l'intégralité du processus de développement.

Les plates-formes Multi-agents

On retrouve dans cette catégorie des traits caractéristiques des systèmes construits sur un modèle d'interaction.

KQML : on se sert de la spécification KQML comme d'une base pratique pour avoir un consensus suffisant sur la sémantique des différents actes, mais on s'autorise des libertés avec l'usage et l'extension des messages. Il s'agit bien de faciliter la vie du concepteur de système plutôt que d'essayer de faire inter-opérer des agents d'applications différentes, sans parler d'intégration entre plates-formes.

Les extensions d'architectures classiques

Certaines infrastructures ont aussi étés désignées comme des technologies de choix pour l'implémentation d'applications multi-agents tels que Corba ou Jini. Il est vrai que dans certains cas, pratiquement rien ne manque, Voyager est un exemple d'infrastructure "mixte" : c'est à la base un bus objet (ORB) écrit en Java, mais étendu par quelques fonctionnalités intéressantes de mobilité d'agent. Voyager permet au concepteur d'écrire des agents, en sachant que la mobilité est fournie au niveau de l'infrastructure, avec quelques autres services. Par exemple, Voyager peut mettre en place automatiquement des adresses permettant de faire suivre les messages au fur et à mesure du déplacement de l'agent. La communication peut être synchrone ou asynchrone, via des boîtes aux lettres éventuellement distribuées.

Par contre, tout comme Aglets, Voyager repose essentiellement sur les mécanismes du langage d'implémentation Java en particulier au niveau de réalisation et gestion fine de la sécurité, mais n'utilise pas vraiment un modèle de comportement riche ou tout au moins un niveau de description authentiquement agent.

La standardisation FIPA

Une approche très différente, par rapport aux grandes catégories de plate-formes que l'on a parcourues, est proposée par le consortium universitaire et industriel, la FIPA (Foundation for Intelligent Physical Agents).

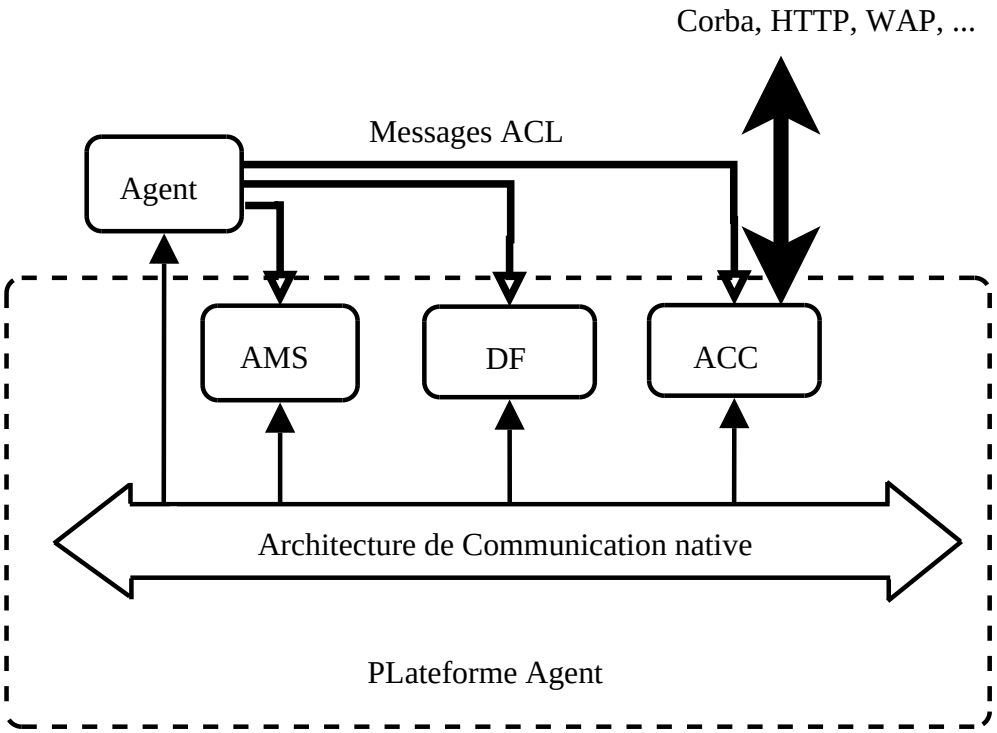

FIG. 1.10 – *Architecture de référence FIPA*

Il s'agit d'un effort d'intégration et d'interopérabilité entre applications multi-agents passant par l'écriture de spécifications, pour normaliser rapidement les technologies agents.

1. Un ensemble de spécifications :
   On peut dégager deux grands ensembles de spécifications "normatives" dans FIPA : celles ayant traits aux plates-formes qui vont héberger les agents, et celles décrivant le langage d'interaction ACL (Agent Communication Language).
   Le but est bien de mettre en place un "middleware" agent qui fournira des services génériques d'accueil, d'identification, et de communication entre agents FIPA. Cet ensemble forme la plate-forme de référence de FIPA (voir Fig 1.11). Dans FIPA, ces services de base peuvent être implémentés eux-mêmes sous la forme d'agents conformes à FIPA. Ces services sont :

   – L'AMS (Agent Management System) : c'est un peu le coeur d'une plate-forme FIPA. Il enregistre les agents actifs, gère leur identité et garde la trace de leur état.
   – Le DF (Directory Facilitator) est un service d'annuaire permettant d'identifier les services utilisateurs sur une plate-forme.

– L'ACC (Agent Communication Channel) est un agent particulier chargé de faire le lien entre le mécanisme de communication natif (et potentiellement non -FIPA) de la plate-forme et les autres agents et plate-formes FIPA éventuellement distantes.

Il faut noter qu'à l'heure actuelle, un débat a lieu au sein de FIPA sur le sujet de "l'agentification" de ces trois services qui sont des agents à part entière, au sens FIPA. Ils communiquaient en FIPA / ACL et étaient des intermédiaires obligés. Par exemple, pour envoyer un message d'un agent FIPA sur une plate-forme à un autre sur une plate-forme distante, il faut déjà faire suivre le message à l'ACC local. Celui-ci transmettrait alors son message à son homologue.
Ce mécanisme a été critiqué pour son inefficacité et sa lourdeur , une interaction ACL n'étant pas quelque chose du point de vue traitement, en double quasiment toutes les interactions par des interactions contraignantes pour des tâches purement administratives.
Le code de la Fig.1.11 montre le message nécessaire pour envoyer un message d'un agent FIPA agent_a à un autre agent_b si l'on observe scrupuleusement la norme.

```
(request
   :sender agent_a
      :receiver acc
      :content (action acc
         (forward
           (request
              :sender agent_b
              :receiver agent_b
              :content
                 ...
          )))
   protocol fipa-request
   language s10
)
```

FIG. 1.11 – *Message FIPA*

L'approche actuelle est donc de considérer l'ACC comme un service non nécessairement agent, qui peut être décrit en termes d'objet à l'usage de l'AMS ou le DF local et qui puisse éviter le déploiement d'une architecture de traitement lourde pour ce service de base. On y perd un mécanisme d'interaction unifié et extrêmement générique FIPA-ACL mais on facilite l'intégration potentielle avec des infrastructures de communication préexistantes et on allège un peu les plates-formes. La mise en oeuvre est bien plus directe, mais l'interaction avec ces services se fait toujours via un protocole d'interaction ACL.
On peut quand même s'inquiéter de ces dérives par rapport à l'idée originelle de FIPA, qui avait le mérite de la cohérence du tout-agent.

L'autre élément clé des implémentations FIPA est ACL. ACL a été dès le départ vanté comme étant un "meilleur KQML", car formellement défini, plus facilement validable, et doté de protocoles d'interactions. Cela a été remis en question par la communauté KQML assez rapidement, qui a ensuite proposé une sémantique formelle pour KQML. Après les premières expériences, il semblerait que les espoirs mis dans ACL ne sont pas tous vérifiés.

Il faut signaler les changements de mode opératoire et de cible que connaît cet organisme. Dans les premiers temps (de 1996 à 1998), des spécifications étaient établies chaque année, puis éventuellement amendées l'année suivante. Depuis 2000, le processus est plus itératif, et se rapproche du fonctionnement d'autres organismes comme l'IETF: des spécifications préliminaires sont établies, testées en vraie grandeur, puis seulement adoptées comme standards, ou bien automatiquement considérées obsolètes si rien n'arrive dans un certain délai. Cela ouvre la voie à l'expérimentation et laisse un plus grand champ d'action à la FIPA.

De façon comparable, l'architecture FIPA a été affinée récemment pour aller dans le sens d'une plus grande modularisation. Nous l'avons vu avec la levée de la contrainte d' "agentification" de l'ACC. De même, CORBA/IIOP n'est plus un pré-requis absolu, en ces temps de téléphones mobiles et protocoles WAP. L'encodage des messages peut être fait en s'appuyant sur d'autres formats que l'ASCII pur, pour ouvrir la voie à XML, Unicode ou à des formats compacts. Tout ceci va dans le sens d'une plus grande variabilité des domaines de déploiement des applications FIPA, même si le modèle de base de l'agent reste inchangé et conditionné en fait fortement le style d'application.

2. La plate-forme Emorphia FIPA-OS :

   Pour illustrer comment ces spécifications peuvent se concrétiser, nous prenons l'exemple d'une des implémentations existantes : FIPA-OS, une plate-forme de validation des spécifications FIPA élaborée sous la houlette des laboratoires de Nortel Networks. En particulier, elle a été le premier outil à mettre correctement en oeuvre les parties obligatoires des spécifications. Elle a été clairement positionnée comme un moyen d'accroître la visibilité et l'utilisation de FIPA.

   Son architecture est un modèle en trois couches : transport, modèle et agent. Elle permet d'accueillir plusieurs modèles d'agents à partir du moment où le modèle de base de FIPA est respecté. Ces agents utiliseront implicitement les services FIPA classiques.

   Des éléments de découplage sont néanmoins présents dans les couches de transport, qui ne sont pas forcément restreintes à Corba / IIOP et aux mécanismes de persistance d'agents.

   Les modèles d'agents de bases sont conçus pour accueillir directement des interactions conformes à ce qu'attend la FIPA. C'est en particulier le respect d'ACL et des protocoles d'interaction qui y sont associés, mais également l'utilisation des langages de contenus définis par FIPA : SL et CCL1.

3. Analyse et discussion :

   Le plus souvent, la plate-forme se confond avec son application : recherche d'infor-

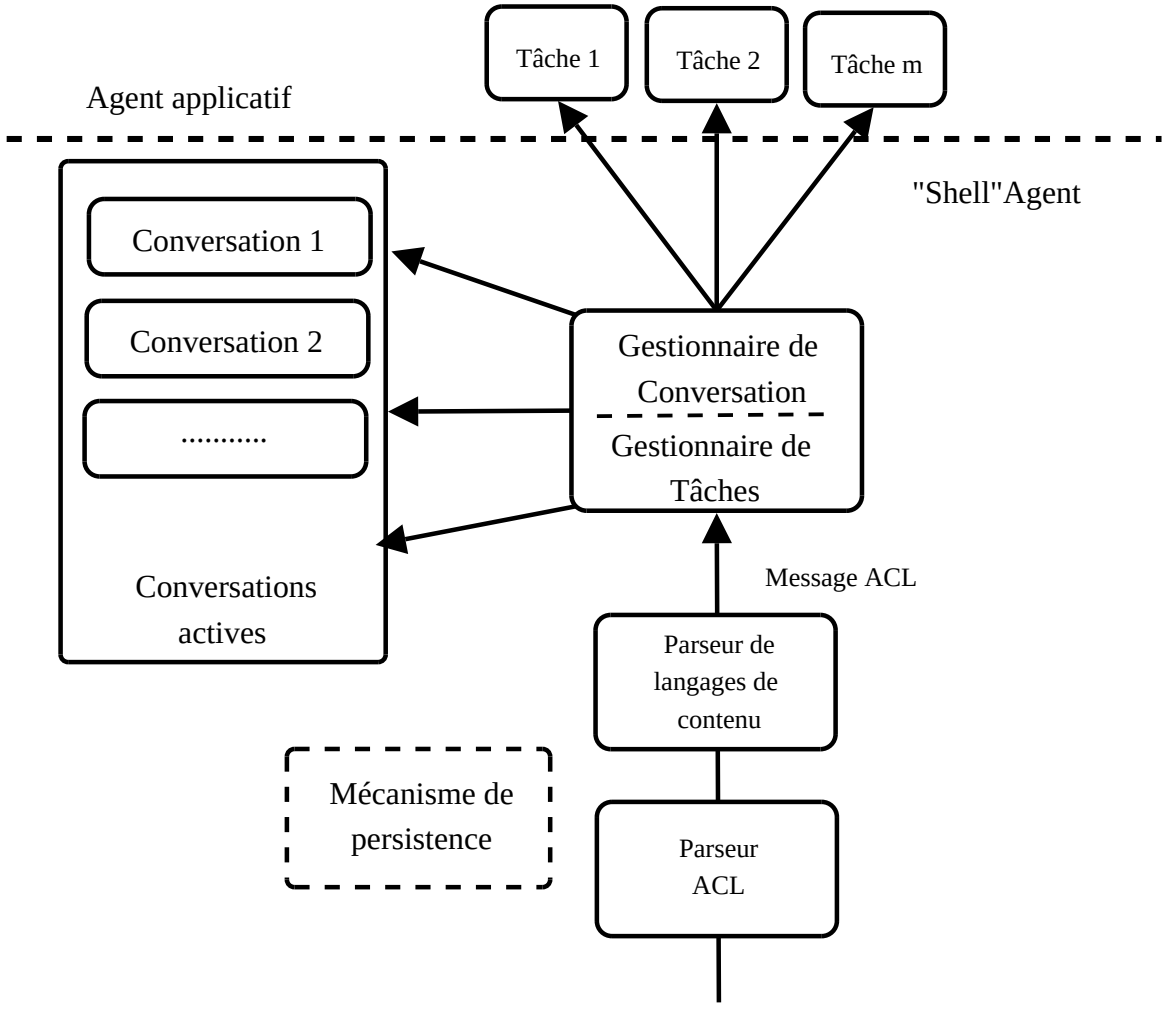

FIG. 1.12 – *Architecture d'un agent classique dans FIPA-OS*

mation, équilibrage de charge, etc. Même si la volonté de généricité est affirmée, il est souvent difficile d'évaluer ce point hors d'un corpus d'applications conséquent. D'autre part, les environnements de conception, développement, et exécution sont définis dans la majorité des cas de façon rigide, alors qu'il serait plus intéressant d'avoir un découplage et une modularisation de ces différents aspects pour permettre de spécialiser les outils en fonction du domaine ou des conditions de déploiement (environnement de test, serveur aveugle, ...). De plus, ces plates-formes posent souvent un problème d'outillage des applications : on ne dispose que rarement d'outils d'analyse du fonctionnement des agents ou de la plate-forme. Et lorsque c'est le cas, il n'est pas réellement possible de construire des mécanismes d'observation spécifique sans avoir à intervenir sur le coeur de la plate-forme.
Mais le principal reproche que l'on pourrait formuler à l'égard des plates-formes

multi-agents est leur spécialisation : la conception d'agent est centrée sur un modèle particulier, ce qui induit forcément une homogénéité sur les applications qui peuvent y être hébergées.

Certaines plates-formes échappent néanmoins à cet écueil, comme DIMA, qui reposent volontairement sur des modèles plus génériques.

La contrainte d'un modèle rigidifié d'agent entraîne souvent un écart par rapport aux choix initiaux du concepteur. Cela pourrait être anodin dans le cas d'un domaine établi, mais cela s'avère particulièrement gênant dans le cas des systèmes multi-agents où les architectures, les modèles, ou même les définitions de base ne sont toujours pas véritablement affirmées.

Même dans le cas de FIPA, qui essaie pourtant d'aboutir à une interopérabilité entre systèmes (via des spécifications), on se rend compte que l'architecture des systèmes est en fait contrainte par deux fois. D'une part le modèle de référence d'un agent et de son langage de communication oriente très fortement le concepteur vers un style particulier d'architecture. Par exemple, exprimer un système en vue d'une résolution par émergence est très malaisé.

D'autre part, l'implémentation particulière de la plate-forme en elle-même va rajouter un modèle de comportement spécifique. Le développeur d'agent n'a d'autre choix que de s'y plier, et d'évaluer soigneusement l'adéquation entre son problème et l'architecture qu'il ne pourra guère adapter à ses besoins.

Pour finir, on notera qu'un des problèmes majeurs du monde des plates-formes agents n'est peut-être ni technique, ni conceptuel, mais plutôt lié au contexte. L'atomisation du travail sur les plates-formes et la très grande spécialisation des modèles n'encouragent guère l'émergence d'une communauté d'utilisateurs.

Les faibles possibilités de partage de modèles, d'agents, ou d'expérience entre concepteurs freinent la généralisation des applications multi-agents, une pédagogie de ces systèmes et l'apparition d'infrastructures communes.

Plate-forme générique

Contexte: Imaginons que nous ayons à concevoir une plate-forme multi-agents générique. La première question que nous nous poserons sera : quelles sont donc les fonctionnalités dont on a besoin pour faire fonctionner les agents qu'on veut construire ?

On pourrait alors rapidement établir une première liste de fonctionnalités classiques: communication par message, moteur de simulation multi-agents, modèle d'environnement, définition de protocoles d'interaction, gestion du cycle de vie, mobilité d'agent, identifications uniques, observation des systèmes multi-agents en activité agents à base de règles, exécution concurrente ou distribuée, sécurité et contrôle des systèmes multi-agents, etc.

Base de communication: Fournir le minimum de ce que l'on peut attendre d'un système de communication asynchrone. Cela comprend pour tout agent l'envoi et la réception de messages via une boîte aux lettres, la garantie de transmission ordonnée et de délivrance en cas de présence du récepteur sur le système.

Gestion de l'identité et du cycle de vie: Permet d'identifier de façon unique des agents, les instancier, en tant qu'agent et non simplement objet, et les suivre jusqu'à disparition.

Politique d'exécution: Définit quelques politiques d'exécution de base (séquencées, parallèles) sur lesquelles bâtir des modèles d'exécution (influence / réaction, comportements concurrents,...)

Structuration applicative: Donner des moyens au concepteur final de faire le lien entre l'organisation de son application et les détails terre-à-terre de catégorisation de ses agents.

Maintien d'une vue organisationnelle: L'ensemble des agents définis sur la plate-forme a accès aux notions de groupes et de rôles. Tout agent peut participer à un schéma organisationnel qu'il peut interroger et avec lequel il peut interagir :

- L'architecture individuelle d'un agent est un choix fort fait par le concepteur. C'est elle qui va définir au minimum un modèle d'action et éventuellement des modèles de raisonnement, perception et de symbolisation, il faut laisser ce choix, et ne pas forcément séparer agents "cognitifs" et agents "réactifs".
- L'activité collective est le point singulier de ces modèles. Figer le modèle de structuration (et non la structure) d'une société d'agents empêche la réintégration ultérieure d'autres entités et modèles.
- La plate-forme doit s'effacer devant ses usages. Vu la richesse des domaines applicatifs des modèles agents, il est souhaitable de définir des possibilités de modularisation de l'infrastructure plutôt qu'une plate-forme monolithique.

1.14 Critiques et Conclusion

Aujourd'hui, les Systèmes Multi-Agents (SMA) commencent à s'imposer en tant que modèle à analyser et à structurer tout genre de phénomène sociétal, mettant en oeuvre des entités interactives distribuées et surtout coopérative. Les entreprises complexes en sont le meilleur exemple, car elles sont distribuées géographiquement et fonctionnellement. Elles présentent tous les critères d'une prise en charge conceptuelle et d'une mise en oeuvre adéquate par les SMA, et ce pour leur bon fonctionnement.

Contrairement à l'approche descendante dite approche conventionnelle statique, modéliser le fonctionnement d'un problème complexe en SMA, c'est partir de tâches simples distribuées avec objectifs locaux afin d'atteindre l'émergence de solution globale du problème posé.

Modéliser en SMA, c'est aussi travailler sur un modèle d'organisation qui répond au mieux au bon fonctionnement du problème posé, car nous pensons qu'il n'y pas de modèle générique pouvant satisfaire tous les critères d'organisation à un problème donné.

En outre, un modèle d'organisation est intimement lié à un modèle d'interaction qu'il faut développer et bien valider. Beaucoup de travaux sont diffusés sur cet axe et beaucoup de concepts nouveaux sont annoncés pratiquement chaque jour. Néanmoins, à notre humble avis, tout concept académique qui ne trouve pas de chemin d'implémentation restera un concept purement théorique. Les différentes plates-formes SMA, qui

sont sensées refléter les concepts des modèles d'organisation, n'arrivent pas souvent à les mettre en oeuvre, surtout pour un SMA dont le nombre d'agents est important.

L'expérience a montré qu'il faut partir d'un problème posé, lui appliquer un modèle d'organisation qui soit lié à des scripts d'interaction qu'il faut bien valider. Et c'est justement ce que nous tentons de réaliser, dans cette démarche, afin de gérer le problème posé et d'une façon optimale.

Chapitre 2

L'Interaction: la dynamique d'un Système Multi-agent

L'Interaction et la Communication sont deux principes qui se côtoient souvent dans le processus d'existence d'un SMA. Nous expliquons, dans ce chapitre, que l'interaction est l'essence même de la dynamique d'un SMA, et que la communication n'est qu'un support de messages qui provoque la dynamique du système. Nous y présentons donc le principe même de l'interaction, ainsi qu'un état de l'art des différents modèles d'interaction existants.

2.1 Introduction

Les recherches dans le domaine des Systèmes Multi-Agents se sont longtemps concentrées sur les modèles et architectures d'agents, pour en faire des entités computationnelles autonomes présentant des aptitudes de haut niveau (agents réactifs, agents cognitifs, architecture Believes/Intentions/Desires,...). Depuis quelques années, l'attention s'est portée sur le niveau organisationnel de ces systèmes, sur les concepts dont la mise en jeu conduisent les agents d'un système à "faire société" [29] [100].

En effet, le phénomène d'émergence, c'est à dire l'effet de synergie des aptitudes et comportements des agents individuels, qui est le bénéfice concret le plus important de l'approche multi-agent, semble résulter de mécanismes liés à ces concepts. Si l'émergence n'est pas le fruit du hasard, elle résulte de mécanismes concernant les relations entre les agents, c'est à dire l'organisation du système, due à l'interaction entre ses éléments, et non pas des propriétés individuelles des agents.

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

En général, les interactions sont mises en oeuvre par un transfert d'informations entre

agents ou entre l'environnement et les agents, soit par perception, soit par communication. Par la perception, les agents ont connaissance d'un changement de comportement d'un tiers au travers du milieu. Par la communication, un agent fait un acte délibéré de transfert d'informations vers un ou plusieurs autres agents.

L'interaction peut être décomposée en trois phases non nécessairement séquentielles [46]:

- la réception d'informations ou la perception d'un changement,
- le raisonnement sur les autres agents à partir des informations acquises,
- une émission de message(s) ou plusieurs actions (plan d'actions) modifiant l'environnement.

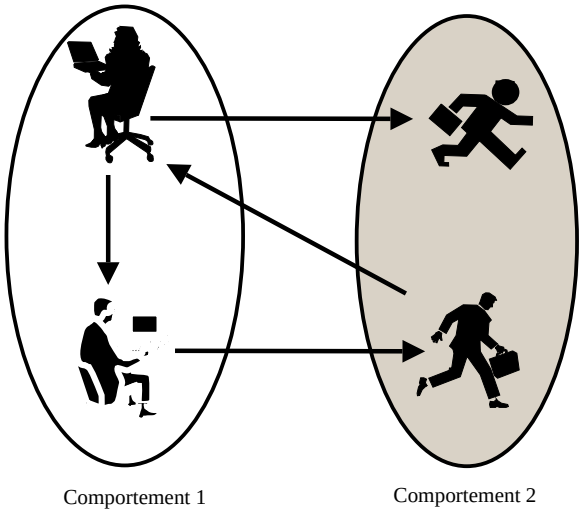

FIG. 2.1 – *Communication et interaction.*

Cette phase est le résultat d'un raisonnement de l'agent sur son propre savoir-faire et celui nécessaires pour traiter les interactions dépendant des capacités cognitives (de raisonnement) de l'agent et du fait que l'agent a connaissance ou non de l'objectif du système global.

En effet, un agent qui poursuit un objectif individuel au sein du système comme c'est le cas pour les agents dits réactifs ne focalise pas son énergie pour interagir avec les autres même s'il y est amené. Par contre, un agent qui participe à la satisfaction du but global du système tout en poursuivant un objectif individuel, va passer une partie de son temps à coopérer ou à se coordonner avec les autres agents. Pour cela, il doit posséder des connaissances sociales qui modélisent ses croyances sur les autres agents.

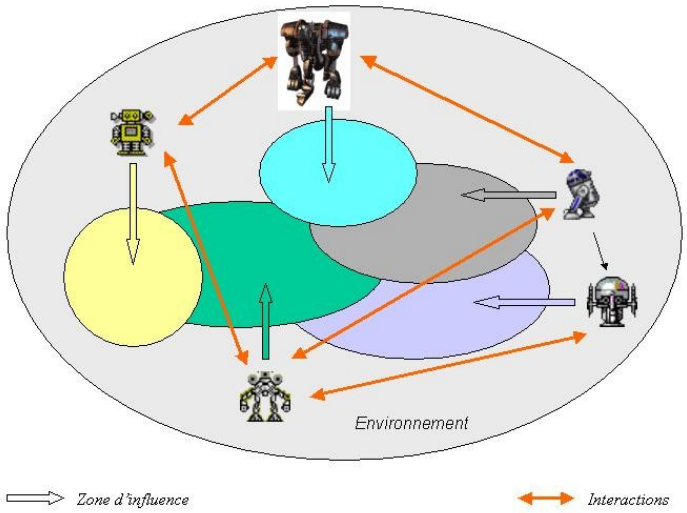

FIG. 2.2 – *Interaction et zones d'influence dans un système multi-agents*

2.2 Eléments d'une interaction

Nous considérerons que les interactions entre agents se produisent dans le contexte de conversations, c'est à dire de séquences d'interactions élémentaires (partiellement) ordonnées par des relations causales. Chaque conversation se déroule selon les règles, conventions, etc. déterminées par un certain protocole d'interaction, qui est caractérisé par [90]:

- Les rôles que les participants peuvent jouer, la notion de rôle étant ici circonscrite à un protocole donné et non pas à l'échelle de l'ensemble du système ; par exemple, un protocole d'enchère comporte les rôles de vendeur, commissaire et d'enchérisseur.
- Le type des interventions, ou interactions élémentaires, que les participants d'une conversation peuvent effectuer les uns envers les autres.
- L'état initial, c'est à dire les conditions qui doivent prévaloir pour l'ouverture d'une nouvelle conversation.
- l'état terminal c'est à dire les conditions qui marquent la fin d'une conversation et correspondent soit à son échec soit à son succès (on suppose que chaque conversation a un objectif, qui est ce qui justifie que des agents y participent, et le succès est la caractérisation de l'atteinte de cet objectif).
- Des contraintes de distribution qui portent sur l'attribution des rôles aux agents : les conditions qu'un agent doit satisfaire pour pouvoir jouer un certain rôle, le nombre des agents qui peuvent jouer chaque rôle, la compatibilité entre les rôles, les possibilités de changer de rôle, l'ordre dans lequel les rôles peuvent être attribués,...
- Des contraintes de comportement qui portent sur les occurrences d'intervention et déterminent les séquences d'interventions qui peuvent constituer le cours d'une

conversation : dans quelles circonstances un participant jouant un certain rôle peut (ou doit) réaliser une intervention d'un certain type vis à vis d'autres participants spécifiés ?

Une conversation qui suit un certain protocole est donc une instance de ce protocole, c'est à dire un processus au cours duquel des agents réalisent des interventions conformes aux contraintes du protocole. La variété des conversations que l'on peut envisager est considérable, par exemple si l'on considère un protocole dans lequel il n'y a qu'un seul rôle, tout énoncé du langage naturel est un type d'intervention, toute situation correspond à un état initial et terminal, et il n'y a aucune contrainte de distribution et de comportement.

La définition et la mise en place d'un protocole mettent en jeux un grand nombre de mécanismes et de propriétés qui concernent non seulement le protocole lui-même, mais aussi les entités en interaction, le médium de communication (ou l'infrastructure, l'environnement dans lequel chacun des agents s'exécute) et les mécanismes de régulation qui peuvent exister au niveau de l'ensemble du système.

L'ensemble de tous ces éléments caractérisent ce que l'on pourrait appeler un espace d'interaction ; cet espace détermine les caractéristiques des conversations susceptibles de se dérouler, et donc le type d'organisation du système. Il en découle une démarche de conception des SMA comportant les étapes suivantes [90]:

- choix du type d'organisation;
- choix du type d'espace d'interaction, en fonction des protocoles mettant en oeuvre cette organisation;
- choix techniques concernant les agents et leur environnement d'exécution;
- définition des mécanismes de régulation qui implantent ces protocoles en déterminant comment les agents utilisent les ressources de l'environnement d'exécution, et inversement les contraintes imposées par cet environnement sur le comportement des agents.

2.3 Les protocoles d'interaction

Les communications inter-agents ne permettent généralement pas de faire émerger aisément des enchaînements conversationnels. C'est la notion de protocole qui permet d'expliciter ces enchaînements afin d'arriver à l'émergence de l'objectif recherché.

2.3.1 Choix d'un protocole

Le choix du protocole est très important, surtout parce qu'un protocole ou un autre peut imposer un certain comportement (préféré) aux agents. Si les agents sont coopératifs, le concepteur peut imposer une certaine stratégie aux agents afin d'avoir le comportement global désiré dans la société d'agents. Si les agents sont egocentrés ou compétitifs, chaque agent est libre de choisir sa propre stratégie, notamment la stratégie qui lui apporte la plus grande utilité. Dans ce contexte, il est important de connaître les critères d'évaluation des protocoles au moment de la conception ou du choix du protocole.

Pour démontrer l'importance de ces critères, on va prendre l'exemple donné par Rosenschein et Zlotkin (1988). Supposons que des ingénieurs d'IBM, Apple et Toshiba se rencontrent et veulent concevoir des agents assistants électroniques personnels capables d'interagir et planifier des réunions communes. Pour que les agents personnels de chaque compagnie puissent interagir, les ingénieurs doivent décider le protocole à suivre.

Un des ingénieurs dit: *"Si on choisit le protocole A, les agents seront capables d'arriver à un accord très vite. La solution trouvée peut ne pas être optimale mais elle sera obtenue très vite"*

Un autre dit: *"C'est vrai, mais il est important aussi qu'un agent ne puisse pas être manipulé par les autres agents. Dans ce cas, le protocole B sera peut-être meilleur."*

Et encore un autre ingénieur dit: *"Pour moi, le critère le plus important est d'avoir l'utilité moyenne la plus grande pour tous les agents. Quel protocole faut-t-il choisir dans ce cas?"*

L'exemple cité montre qu'un protocole a certaines propriétés qui peuvent être évaluées en fonction de divers critères.

2.3.2 Classification des protocoles d'interaction

Les agents communiquent entre eux dans le système pour s'échanger des informations tout en ayant comme objectif d'accomplir des buts qui leurs sont propres ou partagés. La communication peut permettre aux agents de coordonner leurs actions et aboutir ainsi des comportements cohérents du système [69].

La coordination est une caractéristique essentielle dans les SMA. Elle permet, par exemple, d'éviter le conflit sur les sections critiques, en matière de ressources, les attentes indéfinies et les inter-blockages.

La coopération est la forme de coordination entre agents non antagonistes; à l'inverse, la négociation correspond à la coordination en univers compétitifs ou entre agents "égoïstes" ou concurrentiels. Typiquement, pour coopérer avec succès, chaque agent doit maintenir un modèle des autres agents, et développer également un modèle des futures interactions (planification).

L'objectif des protocoles s'inscrit dans deux types d'interactions:

- Agents concurrents: il faut maximiser l'utilité de chaque agent.
- Agents ayant des buts semblables ou des problèmes communs: il faut maintenir des performances globalement cohérentes sans inhiber l'autonomie de chacun. Les aspects importants de ce type d'intercation sont:
  - déterminer les buts partagés
  - déterminer les tâches communes
  - éviter les conflits
  - mettre en évidence la connaissance et la partager

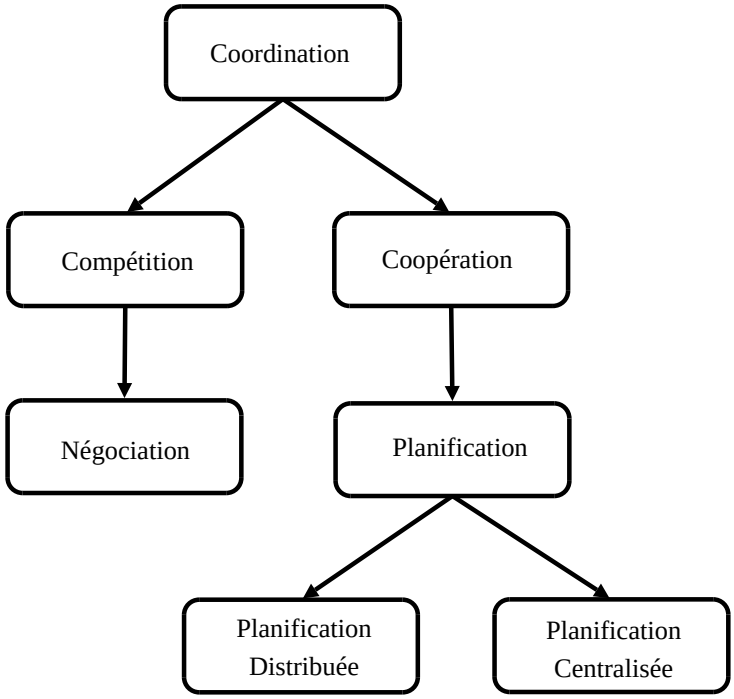

FIG. 2.3 – *Taxonomie des différentes formes de coordination entre agents cognitifs.*

2.3.3 Protocoles de coordination

La coordination se traduit par un comportement individuel visant à servir ses propres intérêts, tout en essayant de satisfaire le but global du système [69]. Cela se fait essentiellement dans des environnement à ressources limitées. Plusieurs raisons exigent la coordination entre agents d'un SMA:

- dépendances entre actions des agents,
- aucun agent ne peut posséder un auto-suffisance en matière de ressources et de compétences afin d'atteindre ses buts tout seul,
- éviter la redondance dans la résolution des problèmes.

Le contrôle distribué, sur lequel se sont concentrés les recherches récemment, concerne surtout l'autonomie des agents qui leur permettent de produire de nouvelles actions et décider des objectifs à atteindre. l'inconvénient majeur de la distribution du contrôle et des données est que la connaissance de l'état global du système est dispersé sur tous les agents où chaque agents possède une perception partielle et imprécise [69]. Il y a plus grand degré d'incertitude au sujet des actions de chaque agent ce qui rend difficile l'atteinte d'un comportement global cohérent. Pour Jennings [93], la coordination est caractérisé par deux aspects étroitement liés:

- Les engagements fournissent la structure nécessaire pour des interactions prévisibles de sorte que les agents puissent prendre en compte les dépendances inter-agents des futures activités, des contraintes globales ou des conflits d'utilisation

des ressources. Pendant que les situations changent, les agents doivent évaluer si les engagements existants sont encore valides.

– Les conventions fournissent des moyens pour contrôler les engagements dans des circonstances changeantes.Elles contraignent les conditions pour lesquelles des engagements devraient être réévalués et indiquent les actions associées qui devraient alors être entreprises: maintenir, rectifier ou abandonner les engagements. Par exemple, si les buts sont dépendants, il est essentiel que les agents soient au courant de n'importe quel changement substantiel qui les affecte. Un autre cas, lorsque les agents s'engagent ensemble à satisfaire un but et contribuent individuellement à des tâches (participation d'une équipe entière), un changement d'engagement par un participant peut compromettre les efforts de l'équipe. par conséquent, les agents auront besoin d'être informés et de se repositionner par rapport aux engagements pris collectivement.

Exemple de protocole de coordination : Le contract-Net.

2.3.4 Protocoles de négociation

La négociation est une interaction entre agents basée sur la communication avec le but d'arriver à un accord. Elle peut être vu comme un processus par lequel une décision commune est prise par deux agents ou plus où chacun d'entre eux essayant d'atteindre leurs buts ou objectifs propres.

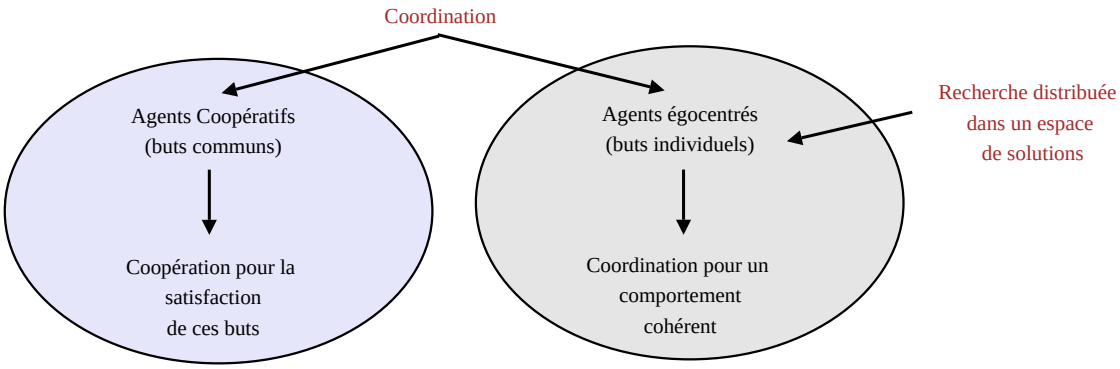

FIG. 2.4 – *La coordination à la base des protocoles d'interaction.*

Les principaux dispositifs de la négociation sont:

– un langage de communication, utilisé par les agents participants
– un protocole de négociation suivi par les agents durant tout le processus de négociation
– un processus de décision par lequel un agent décide ses positions, concessions et critères pour l'accord.

La négociation peut se faire selon :

– négociation un-à-un

- négociation un-à-plusieurs
- négociation plusieurs-à-plusieurs

Les techniques de négociation peuvent être centrées environnement ou centré agent (égocentré):

1. Négociation centrée agent: Quelle est la meilleure stratégie pour un agent évoluant dans un environnement donné?.
2. Négociation centrée environnement: Quelles sont les règles à appliquer sur un environnement afin que les agents faisant parti de cet environnement puissent être productifs?.

Le mécanisme de négociation doit avoir les caractéristiques suivantes:

- L'efficacité, ne pas perdre de ressources pour aboutir à un accord.
- La stabilité qui consiste, pour un agent, de ne pas sortir de la stratégie tracé initialement.
- La simplicité, en coût de calcul et de bande passante.
- La distribution, qui évite l'utilisation d'un agent central superviseur.
- et la symétrie

Les techniques de négociation peuvent être:

- Négociation basée sur la théorie des jeux
- Négociation basée sur des heuristiques
- Négociation basée sur l'argumentation

Coté ingénierie de protocoles, comme chaque agent peut choisir sa propre stratégie. Ceci permet aux agent du SMA d'être conçus et implémentés différemment, tout en assurant un middleware de communication entre les agents du système. De plus, on ne peut pas assumer que les agents utilisent des stratégies "coopératives" de négociation.

2.3.5 Protocoles de coopération

Ces protocoles s'appliquent surtout à la résolution des problèmes complexes. Cela consiste à la décomposition du problème en tâches affectées à des agents distribués. Cette décomposition doit prendre en considération les capacités et les ressources de chaque agent du système. Une telle décomposition peut réduire la complexité d'un problème.

Soit un problème, même s'il ne possède pas les critère d'un problème complexe mais nous pouvons lui appliquer un protocole de coopération. Cela consiste à traiter une image complexe (Fig 2.5) en lui appliquant des opérations de morphologie mathématique (érosion, dilatation, ouverture, fermeture, etc.). Cette image peut être divisée en plusieurs segments $segment_1$, $segment_2$, $segment_3$ et $segment_4$, et à chaque segment on lui affecte un agent qui assure une certaine tâche bien précise. sachant que tous agents travaillent en coopération en traitant les frontières entre les segments [15].

D'une façon générale, la décomposition d'un problème complexe peut être spatiale ou fonctionnelle. une fois la décomposition faite, il faut distribuer les tâches selon les critères suivants [69]:

- Éviter la surcharge des ressources critique.

- Assiger les tâches aux agents ayant des capacités correspondantes.
- Assigner les tâches interdépendants à des agents proches spatialement ou sémantiquement pour limiter les coûts de communication et de synchronisation.
- Réassigner des tâches pour accomplir les plus urgentes.

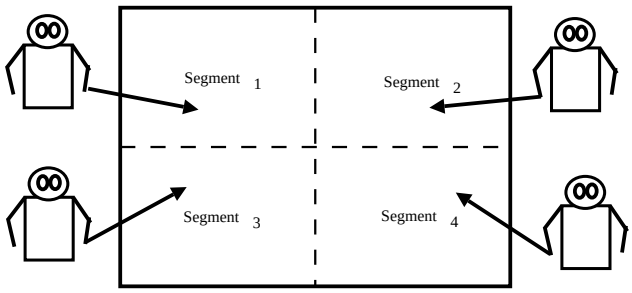

FIG. 2.5 – *Coopération entre agents pour le traitement d'une image.*

Plusieurs mécanismes sont utilisés pour distribuer des tâches. En voici quelques-uns:

- Mécanisme d'élection:des tâches sont attribuées aux agents suite à un accord ou un vote.
- Réseaux contractuels:cycles d'appels d'offres, de propositions et attribution à un agent.
- Planification Multi-agents: Des agents dédiés à la planification des tâches.
- Structure organisationnelle: les agents ont des des responsabilités pour des tâches bien particulières.

2.3.6 Critères d'évaluation d'un protocole

Les critères d'évaluation des protocoles de négociations que nous proposons dans ce paragraphe se basent sur la théorie des jeux [33]:

1. Critère d'efficacité de Paretto:
   Une solution (résultat) x à un problème de négociation est efficace Pareto (Pareto optimale), s'il n'y a pas une autre solution x' telle qu'au moins un agent ait une meilleure situation avec x' qu'avec x et aucun agent n'ait une plus mauvaise situation avec x' qu'avec x.. Autrement dit, le résultat (solution) d'une négociation est Pareto optimal s'il n'y a pas un autre résultat qui fasse gagner plus à un agent aux dépens d'un autre agent qui, lui, gagnera moins dans cette solution. L'efficacité Pareto mesure le bien global et ne demande pas la comparaison d'utilités.
   Dans le dilemme du prisonnier, le résultat efficace Pareto correspond au cas où les deux prisonniers se tairaient, notamment coopération (CC). Puisque dans ce problème les valeurs d'utilité sont comparables, le bien-être social peut être aussi utilisé et il correspond toujours au cas (CC).

2. Critère de stabilité:
   Pour pouvoir définir le critère de stabilité, il faut d'abord comprendre ce que c'est une stratégie dominante. Si on désigne par $r = f(Act_A,Act_B)$ le résultat (état) des actions Act_A de l'agent A et Act_B de l'agent B, on peut considérer que, pour différentes combinaisons d'actions possibles de l'agent on obtient différents résultats. Par exemple, dans le dilemme du prisonnier on a quatre résultats possibles, correspondant aux paires d'actions CC,CD,DC et DD. La stratégie de négociation de l'agent lui dit quelles actions il doit faire et correspond à un ensemble de résultats (états) obtenus en suivant cette stratégie. On dit qu'une stratégie $S_1 = r_{11},r_{12},...,r_{1n}$ domine une autre stratégie $S_2 = r_{21},r_{22},...,r_{2m}$ si n'importe quel résultat r dans S_1 est préféré à n'importe quel résultat r' dans S_2.
3. de bien être social:
   Le bien-être social est la somme de toutes les utilités (ou gains) des agents pour une certaine solution de la négociation. Le bien-être social peut être utilisé pour comparer deux protocoles en faisant la comparaison des résultats auxquels les deux protocoles ont conduit. Le problème qui pose ce critère est qu'il faut comparer les utilités de plusieurs agents et cela n'est pas toujours possible.
4. Critère de rationalité individuelle:
   Un protocole est rationnel pour un agent si le gain de l'agent qui participe à une négociation avec ce protocole n'est pas inférieur (d'habitude il est supérieur) au gain que l'agent obtiendrait s'il ne participait pas à la négociation. Si un protocole n'est pas rationnel pour un agent alors l'agent (autonome) n'a aucune raison de participer à la négociation. Un protocole est rationnel au niveau individuel s'il est rationnel pour chaque agent participant à la négociation.
5. Critère d'équilibre Nash:
   Deux stratégies, S_1 de l'agent A et S_2 de l'agent B, sont dans un équilibre Nash si:
   dans le cas où l'agent A adopterait S_1 l'agent B ne peut pas faire mieux que d'utiliser S_2 et dans le cas où l'agent B adopterait S_2 l'agent A ne peut pas faire mieux que de d'utiliser S_1.
   La définition peut être généralisée pour plusieurs agent qui suivent les stratégies S_1,S_2,...,S_k.
   L'ensemble de stratégies S_1,S_2,...,S_k suivies par les agents A_1,A_2,...,A_k est dans un équilibre Nash si, pour chaque agent A_i, la stratégie Si est la meilleure stratégie à suivre par Ai pourvu que les autres agents suivent les stratégies S_1,S_2,...,S_{i-1},S_{i+1},...,S_k..
   Cette forme d'équilibre mutuel est très importante car, si on a un équilibre Nash aucun agent n'aura de raison de suivre une autre stratégie que celle qui assure l'équilibre.
   Revenons au problème des prisonniers. Dans ce cas, l'équilibre Nash est assuré par le cas où les deux prisonniers dénoncent (DD). En fait, on a vu quand on a présenté le problème que DD était le comportement rationnel pour les deux agents.
   Il faut noter que dans le cas du dilemme de prisonnier l'équilibre Nash est assuré par les actions DD alors que le bien-être social est garanti par les actions (CC).

2.4 Facilitateurs

Les facilitateurs représentent une classe spécifique d'agents qui distribuent des meta-informations sur les autres agents et offrent des services de communication tels que les suivants en KQML :

- retransmission et distribution des messages(message forwarding and broadcasting)
- découverte des ressources
- routage basé sur le contenu du message
- appariement

Les performatives utilisées d'intérêt pour les facilitateurs sont:

$$advertise,broker,recruit,recommend,forward,broadcast$$

dans leurs variantes "$P-one$" et "$P-all$".

Les facilitateurs peuvent être des agents intelligents ou simplement des "pages jaunes" [17].

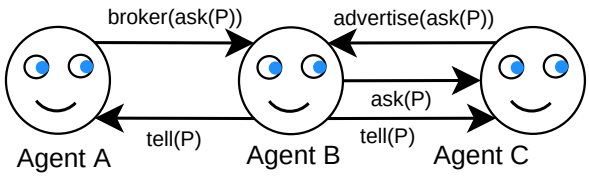

FIG. 2.6 – *Schéma des messages échangés entre les agents A et C en utilisant les services du facilitateur B dans le cas de la performative broker.*

On peut supposer que la performative $advertise(ask(P))$ a été exécutée par l'Agent C avant que l'Agent A n'ait lancé $broker(ask(P))$. D'une façon similaire au cas présenté dans la Fig.2.6, on peut représenter le dialogue de 2 agents et un facilitateur pour la performative recruit Fig.2.7

Dans le cas de la performative *recommend* le facilitateur B retransmet vers l'agent A une éventuelle performative $advertise(ask(P))$ reçue antérieurement de la part de l'Agent C.

On voit dans Fig.2.8 qu'après la réception du message $forward(advertise(ask(P)))$ émis par le facilitateur B, l'Agent A ouvre un dialogue direct avec l'Agent C.

2.5 La communication dans les SMA

La communication entre les différentes agents est l'élément essentiel dans un univers SMA. En plus la communication entre deux agents, elle permet des communications:

- à la cantonade

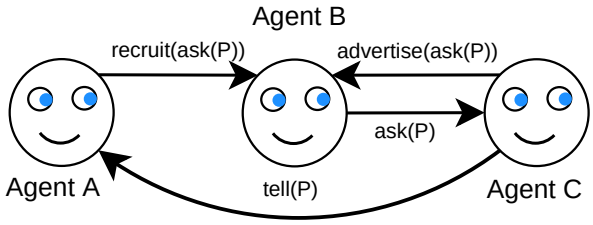

FIG. 2.7 – *Taxonomie des différentes formes de coordination entre agents cognitifs.*

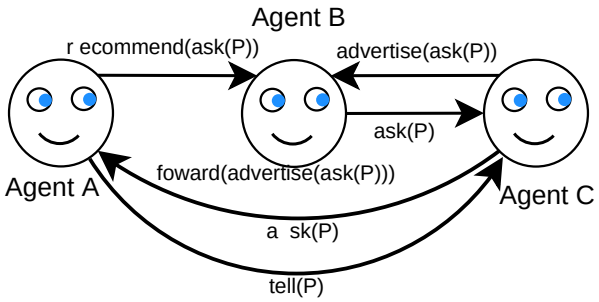

FIG. 2.8 – *La réalisation de la performative recruit à l'aide du facilitateur B.*

- à tout le monde
- à une famille d'agents
- à moi même (self)
- etc.

2.5.1 Pourquoi un langage de communication ?

En effet, les langages de communication dédiés aux agents sans un environnement SMA, calquent leur communication sur la communication humaine, avec des communications directes, indirectes, spécifiques ou ponctuelles, du genre:

- X donne à Y une information,
- X demande à Y une information,
- X donne à Y un ordre,
- X donne à Y une information à transmettre à Z,
- X demande à Y de demander à Z une information,
- X demande à Y d'ordonner à Z,
- X ordonne à Y de demander à Z ...,
- X dit aux Y ... mais n'attend pas de réponse,
- X dit aux Y ... mais attend quelques réponses,

– X dit aux Y ... mais attend une réponse de chacun,
– etc.

Sachant que X, Y et Z peuvent être agents à part entière ou des agents Moi-même (self), anonyme, groupe d'agents ou un agent nommé.

Ces questions sont abordées en considérant:
– Les messages avec accusé de réception,
– Les messages sans accusé de réception,
– avec continuation (désignant à l'intermédiaire les destinataires ultimes...),
– sans continuation.
– Un type d'émission collective, peu ou pas orienté, appelé diffusion "broadcast".

Cela a pour objectif, d'enrichir et de faciliter la modélisation, le prototypage, la simulation voire la réalisation des systèmes complexes.
C'est à ce genre de questions qu'un langage de communication doit répondre!

2.6 Langage de communication entre agents

Les communications dans les systèmes multi-agents comme chez les humains, sont à la base des interactions et de l'organisation. Une communication peut être définie comme une forme d'action locale d'un agent vers d'autres agents [28].
Un modèle de communication doit répondre aux interrogations suivantes:
– X communique quoi?
– X communique à qui?
– quand X communique t-il?
– pourquoi X communique t-il?
– et comment X communique t-il?

Grâce à la coordination, un système multi-agents peut réaliser ses tâches avec plus d'efficacité qu'un seul agent. Mais pour coordonner l'activité d'un ensemble hétérogène d'agents autonomes, il faut que les agents communiquent dans un langage compréhensible par tous les autres. On observe que dans un système ouvert un tel langage peut constituer une interface entre les agents.
L'utilisation d'un langage commun implique que tous les agents comprennent son vocabulaire sous tous ses aspects concernant[69]:
– La syntaxe, qui précise le mode de structuration des symboles;
– La pragmatique pour pouvoir interpréter les symboles;
– L'ontologie pour pouvoir utiliser les mêmes mots d'un vocabulaire commun.

La compréhension du sens des symboles, ou à quoi les symboles font référence, demande un standard rigoureux de la sémantique et de la pragmatique. De plus il est nécessaire que les agents sachent bien utiliser le vocabulaire pour atteindre leurs buts,

éviter les conflits, coopérer pour exécuter leurs tâches et modifier l'état mental d'un autre agent.

Un Langage de Communication Agent ACL (Agent Communication Language) est conçu comme un langage de haut niveau qui assure en premier lieu l'échange d'états mentaux et le sens du vocabulaire [17]. Le format utilisé pour l'échange des connaissances est donné par un langage de contenu, indépendant du langage ACL (p.ex.. KIF, FIPA-SL, FIPA-CCL). Le vocabulaire commun concerne les définitions précisées dans une ontologie. Ces composants sont représentés dans la Fig. 2.9

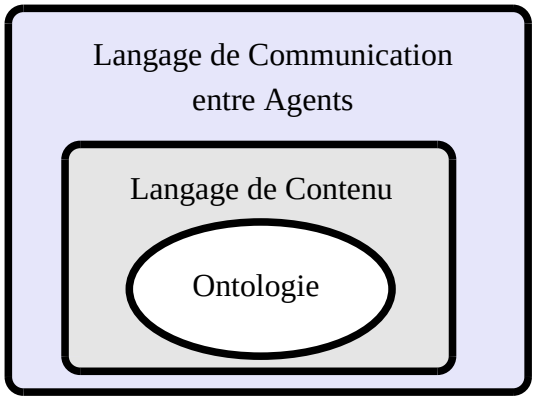

FIG. 2.9 – *Modèle des Langages de Communication entre Agents.*

Les aspects techniques d'implantation d'un ACL concernent l'existence dans le système de communication entre les agents des mécanismes ci-dessous:

- des protocoles pour la couche de transport utilisé (TCP/IP, UDTP, SMTP, IIOP, HTTP)
- des protocoles de haut niveau (e.g. contract-net, licitations ('auctions'), enregistrement des noms)
- des services d'infrastructure (broker, facilitateurs, loggers etc.)
- un mécanisme de contrôle de la communication au sein des agent

Les échanges d'information peuvent être faits par:

- Messages (point à point, diffusion, synchrone ou asynchrone)
- mémoire partagée (tableaux noirs)

Il y a une variété de points de vue dans la littérature concernant ce qui devrait être considéré comme sémantique pour les actes de langage. Une des approches est celle qui interprète comme sémantique les conditions dans lesquelles on peut dire que les actes sont accomplis (ou exécutés).

Un autre point de vue est de s'intéresser à l'identification des effets qu'un acte de langage peut avoir sur les états cognitifs du locuteur et du destinataire et comment un agent doit répondre à un acte spécifique du langage. Puisque cette approche relève des aspects perlocutoires des actes du langage, elle ne peut être classée que comme pragmatique plutôt que sémantique.

Le troisième point de vue est que la sémantique constitue les conditions par lesquelles on peut affirmer qu'un acte particulier du langage a été satisfait. Par exemple, un ordre est satisfait s'il est obéi et une promesse est satisfaite si elle est exécutée dans le futur.

Indépendamment de la sémantique qu'on adopte pour les actes du langage, il est nécessaire de faire référence aux états cognitifs des agents qui les utilisent. Après tout, les actes du langage ne sont que le résultat des actions des agents les uns sur les autres ou sur l'environnement. Ainsi la représentation et le raisonnement à propos des états des agents et de l'environnement et à propos de la manière dont les actions les affectent est un préalable pour toute approche sémantique

2.7 Conclusion

Nous avons montré dans ce chapitre que l'interaction est au coeur des système multi-agents. Comme un agent possède la capacité d'être autonome dans ses prises de décision, il doit le faire en fonction de son environnement avec lequel il est en interaction continue.

La situation dans laquelle se trouve un agent et l'objectif à atteindre par ce dernier déterminent son comportement vis-à-vis de cet ensemble d'agents avec lesquels il interagit.

Chaque comportement est fonction d'un protocole d'interaction. Chaque situation (de coopération, de négociation, etc.) exige un protocole associé et adéquat.

Nous concluons ici que les SMA dont le modèle d'organisation est basé sur le critère du groupe font appel généralement à des agents facilitateurs. Et justement l'objet de notre modèle d'interaction que nous allons présenter au chapitre 5.

Chapitre 3

Vers un équilibrage de charge dans les Systèmes Multi-agents

Chaque agent activant dans un Système Multi-agents est doté d'une charge de travail ainsi qu'une charge de ressources, à définir et à gérer selon ses capacités. Cela est caractérisé par un ensemble de tâches que cet agent est en devoir d'effectuer, durant un intervalle de temps ΔT. Comme cet agent peut être sur-chargé ou sous-chargé, cela peut affecter négativement l'ensemble des noeuds (ou sites) du système. Alors, il doit coopérer avec son environnement, c'est-à-dire l'ensemble des agents représentant les sites avec lesquels il entretient une relation, appelés généralement accointances, afin d'aider à réaliser un équilibrage de charge, pouvant assurer une meilleure émergence de solution individuelle ou globale. Dans ce chapitre, nous présentons un état de l'art des différentes politiques ainsi que différents algorithmes adoptés et qui rentrent dans l'équilibrage de charge.

3.1 introduction

Les Systèmes multi-agents sont implémentés actuellement sur des réseaux, sans lesquels ils ne peuvent pas être validés. Ces réseaux peuvent être des réseaux locaux ou carrément l'Internet. Ils présentent aussi de nouvelles opportunités pour développer des applications parallèles très performantes.

Un tel système dans lequel le nombre de ses membres augmente et les actions de ces membres nécessitent des grandes quantités de ressources. Ces dernières ne sont pas forcément centralisées.

Un service d'équilibrage de charge permet aux différents membres de ce système d'accéder et de bénéficier pleinement de ces ressources distantes afin de les utilisées au mieux de manière transparente.

3.2 Définition de la charge

En anglais " workload ", la charge se définit en fonction de la nature de l'application. Elle pourrait être l'ensemble de messages servis par un dispositif du réseau, un nombre de

tâches à exécuter par processeur ou encore un stock de pièces de rechange d'un système de production. Donc elle peut être de nature physique ou virtuelle.

3.3 Principe de l'équilibrage de charge

Le but de toute technique d'équilibrage de charge (load-balancing) est d'optimiser l'utilisation des processeurs, chargés de prendre en charge, l'exécution des tâches dédiées à ces charges tout en spécifiant ses localités.

Autrement dit, l'équilibrage de charge doit minimiser le temps d'exécution d'un ensemble donnée de tâches, ce qui revient souvent à maintenir une charge équivalente sur l'ensemble des processeurs. En général ce mécanisme est aussi utilisé pour gérer l'équité de temps d'exécution entre les tâches.

L'équilibrage de charge est également désigné sous les termes : la régulation de charge, le partage de charge ou encore l'ordonnancement distribué des charges, désigne toutes les techniques utilisées pour transférer la charge de travail des noeuds (ou sites) surchargés vers des noeuds moins chargés en respectant la capacité de chaque noeud, c'est à dire déterminer celui des noeuds le moins chargé et le plus apte à faire tourner l'application.

Même si jusqu'à maintenant la charge d'un noeud est spécifié surtout par l'ensemble des tâches (processus et threads) tournant sur le noeud. La vision nouvelle qui s'impose actuellement est la gestion et l'équilibrage des ressources, surtout en matière de logistique, dans une entreprise distribuée, dont les ressources sont distribuées et dont l'équilibrage constitue la condition de base pour la bonne gestion de l'ensemble.

3.4 Elaboration d'une fonction d'équilibrage de charge

L'élaboration d'une fonction d'équilibrage de charge pour un système requiert, de manière générale, la mise en oeuvre des fonctionnalités suivantes :

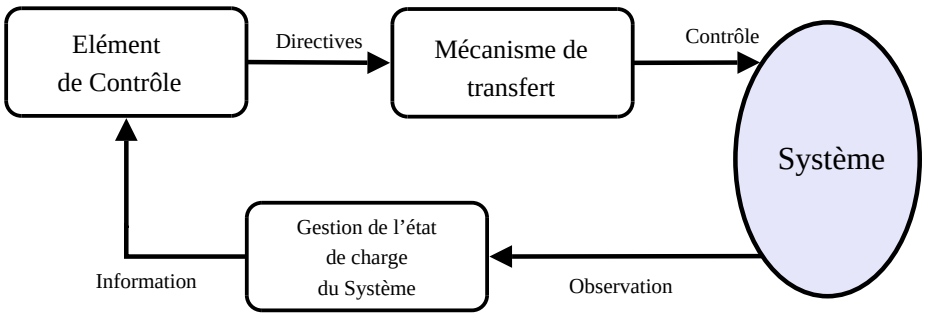

FIG. 3.1 – *Elements d'une fonction d'équilibrage*

– Un gestionnaire de l'état de charge du système (composant d'information) ; Il a

pour objectif, le maintien de l'information concernant l'état de charge de chaque site pour établir la charge globale du système, à partir duquel l'élément de contrôle prendra la décision d'équilibrage de charge. Ces informations permettant de déterminer le meilleur noeud récepteur (le mois chargé) ou de déterminer le noeud émetteur (le plus chargé). Il s'appuie pour cela sur :

- L'estimation de la charge de chaque noeud.
- L'estimation de l'état global du système.

- Un élément de contrôle et prise de décision (composant de contrôle) ;
  C'est l'élément décisionnel d'une fonction d'équilibrage de charge. Il établit des directives à partir de l'état de charge des noeuds maintenu par le gestionnaire et de la stratégie de répartition qu'il met en oeuvre. Le choix d'une stratégie est déterminé à partir des caractéristiques de l'architecture cible et de critères d'efficacité poursuivie par le système. Cet élément décide :
  L'instant où il est nécessaire d'effectuer un équilibrage de charge (politique d'Activation).

  - La quantité de charge (ressources) à transférer (politique de Détermination).
  - L'appariement des noeuds source et cible du transfert (politique de Sélection).

- Un mécanisme de transfert de la charge ;
  La répartition de charge peut être dynamique ou statique :
  Dans une stratégie statique (placement) : la décision de placement est uniquement basée sur les informations concernant le comportement moyen du système. Dans ce cas, les tâches sont allouées à leur création sur un site et y restent jusqu'à leur terminaison.
  Dans les approches dynamiques (migration) : les décisions sont prises en fonction de l'état courant du système.

1. Gestionnaire de l'état de charge de système
2. Élément de contrôle et prise de décision
3. Mécanisme de transfert de la charge

Sur la base de cette répartition des rôles (Gestionnaire de l'état de charge de système, Élément de contrôle et prise de décision et Mécanisme de transfert de la charge) nous attribuerons des rôles similaires à nos agents lors de la modélisation de notre problème dans le chapitre 4, afin qu'ils puissent assurer l'équilibrage de ressources des article (pièces de rechange) dans une e-entreprise.

De manière plus précise, les algorithmes d'équilibrage de charge peuvent se distinguer par cinq caractéristiques majeures, comme décrit par Cyril Fonlupt [11]

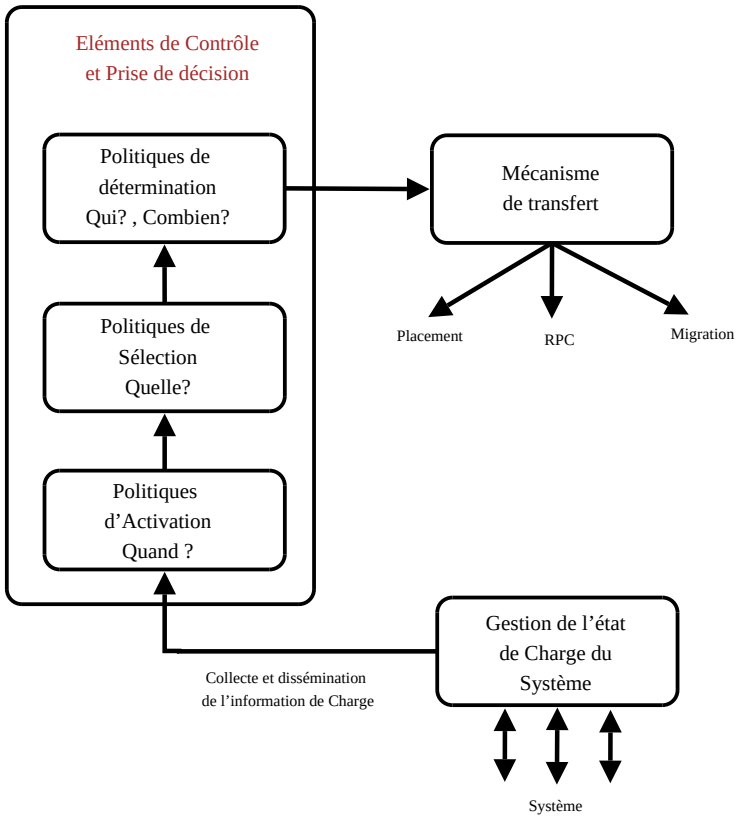

FIG. 3.2 – *Elements d'une stratégie d'équilibrage de charge*

3.5 Caractéristiques d'équilibrage de charge

Notre objectif ici est l'étude de l'équilibrage de charge sur réseau informatique en terme de nombre de processus à exécuter sur chaque poste et pouvoir appliquer ces algorithmes et ces stratégies à notre cas qui est celui de l'équilibrage de ressources dans une e-entreprise, tout en considérant qu'une ressources est une charge à bien gérer et d'une façon optimale.

3.5.1 Politique de mise à jour des informations

La politique d'informations tente d'obtenir un état ou une carte du système en faisant transiter ou en collectant des informations sur l'ensemble ou une partie des noeuds. Il faudra alors déterminer quel type d'information est utile, quand sont elles nécessaires et quels sont les processeurs qui en sont demandeur/fournisseur.

Mesure de la charge: L'état de charge des différents processeurs est la principale source d'informations pour les techniques d'équilibrage. Ce qui nous amène à expliciter la notion de charge d'un processeur. Le but premier de la mesure de la

charge est d'estimer la quantité de traitement attribué à un CPU. Aucune solution universelle existe pour représenter parfaitement cette information. Une technique courante consiste à se servir du nombre de processus en attente d'exécution. Cela a l'avantage d'être rapide à calculer. Il est tout à fait possible de se servir d'autres critères, voir même une combinaison linéaire de plusieurs. Pour éviter les fluctuations brusques des valeurs il arrive qu'elles soient lissées sur le temps.

Déclenchement de la transmission d'informations: Trois classes de politiques peuvent être distinguées.

- Les politiques à la demande, dans ce cas les informations sont assemblées et envoyées à chaque fois qu'un processeur a besoin des informations.
- Les politiques périodiques, les informations sont récoltées de manière régulière et elles sont stockées soit de manière centralisée ou distribuées sur un ensemble de processeurs.
- Enfin les politiques à changement d'état, les fournisseurs décident lorsqu'il faut transmettre une mise à jour des informations parce qu'ils passent d'un état remarquable à un autre (inactif ou surchargé par exemple).

3.5.2 Politique de déclenchement

Afin d'éviter le gaspillage des ressources du système, la politique de déclenchement détermine le moment optimal d'équilibrage à partir des informations fournies par la politique précédente. Un équilibrage trop tardif provoque une augmentation de l'inactivité des processeurs, tandis qu'un déclenchement trop précoce entraîne des sur-coûts importants dus à des communications et des redistributions de données inutiles. Deux types de déclencheurs peuvent être envisagés, des mécanismes décentralisés dont la responsabilité est distribuée sur l'ensemble du système et des mécanismes centralisés où l'opportunité de déclenchement est décidée par une seule entité. Les politiques de déclenchement décentralisées sont réservées aux machines asynchrones qui permettent une certaine autonomie des noeuds. Dans ce cas le processus de décision obéit en général à une technique à base de seuils (charge très haute ou très faible par exemple). Les politiques de déclenchement centralisées impliquent en règle générale des déclenchement à des fréquences fixes, ce qui fonctionne bien sur des ensemble de processeurs synchronisés. Lorsque le déclenchement est décentralisé on peut encore distinguer les politiques en fonction du type du demandeur. Si ce sont les noeuds chargés qui déclenchent l'équilibrage afin de pouvoir "pousser" certaines tâches vers des noeuds moins chargés alors c'est une stratégie dite active. Si, au contraire, ce sont les noeuds ayant une faible charge de travail qui tentent de "tirer" des tâches à partir de pairs plus chargés alors la stratégie est dite passive. Comme nous le verrons par la suite, Linux utilise une telle stratégie. Chacune de ces stratégies a ses avantages et ses inconvénients concernant l'opportunité du déclenchement (ou du non déclenchement). Pour tenter de réunir le meilleur des deux il existe parfois des stratégies mixtes où à la fois les processeurs chargés et ceux non chargés déclenchent l'équilibrage (une telle stratégie est employée dans FreeBSD).

3.5.3 Politique de sélection

La politique de sélection détermine les différents noeuds déséquilibrés qui joueront le rôle d'émetteur ou de receveur. La politique de sélection détermine, si la participation d'un site à l'équilibrage est ou non propice. Dans l'affirmative, elle détermine quel rôle jouera chaque site ; dans la plupart des méthodes existantes, on distingue deux cas :

- Les noeuds sous-chargés qui joueront le rôle de récepteur ;
- Les noeuds avec un excédant de charge qui joueront le rôle d'émetteur.

Il existe trois classes de politique de sélection. De nombreuses stratégies de sélection sont fondées sur une politique à base de seuils où un noeud compare sa charge locale avec un ou plusieurs seuils. Suivant le résultat de cette comparaison, le noeud deviendra émetteur ou récepteur de charge. Des stratégies de comparaisons peuvent également être employées, où le processeur adopte un comportement en fonction de l'état de l'ensemble ou d'un sous-ensemble de noeuds. Certaine méthodes dites systématiques ne tiennent pas compte de l'état actuel des noeuds. Ces techniques sont basées sur des résultats théoriques. Elles provoquent des échanges de travail dans un ordre donné et désignent alternativement les noeuds comme émetteurs, receveurs ou inactifs.

3.5.4 Politique de désignation locale

La politique de désignation locale identifie les processus qui vont être transférés. Cette étape est réalisée immédiatement après la politique de sélection lorsque le noeud est désigné comme émetteur. Par contre, elle s'exécute après la politique d'appariement si le noeud est receveur. Lorsque le système est capable de déplacer un processus déjà en cours d'exécution, c'est un système préemptif. Un tel système est plus souple mais il est plus difficile à implanter. Dans ce sens, Linux est un système préemptif. Il y a deux types de méthode pour sélectionner les tâches. Les méthodes systématiques considèrent toutes les tâches identiques et sélectionnent de manière très simple celle à déplacer, par exemple par FIFO ou par tirage aléatoire. L'avantage est que le processus de sélection est pratiquement immédiat et donc le coût de l'équilibrage est faible. Le deuxième type de méthode consiste à filtrer les tâches à migrer. Ceci peut être fait manuellement par l'utilisateur ou en ordonnant les tâches en fonction de leur propriété (utilisation du CPU, affinité à un noeud, âge du processus...).

3.5.5 Politique d'appariement

Il s'agit de trouver un ou plusieurs partenaires au processeur qui a été choisi par la politique de sélection. Cette partie est à juste raison considérée comme étant le coeur d'une méthode d'équilibrage. Le partenaire peut être recherché sur un domaine limité spatialement à un voisinage de processeurs ou sur l'ensemble du système. La politique d'appariement est bien évidemment étroitement liée à la politique d'informations ; il est par exemple beaucoup plus aisé d'introduire une politique d'appariement centralisée si la politique d'information est également centralisée.

Les politiques d'appariement sont extrêmement nombreuses. En plus des techniques aléatoires (choix du noeud au hasard), il y a des politiques centralisées, d'autres distribuées et des politiques mixtes. Une stratégie centralisée désigne un noeud comme serveur, il détient des informations qui seront utilisées pour la recherche de noeuds destinataires. Cette stratégie est dite "aveugle" lorsque le serveur choisit le processeur de destination en se basant uniquement sur ses actions passées (aucune information du système n'est nécessaire). Une stratégie asynchrone signifie que les processeurs recherchant des tâches à prendre ou à transmettre demandent au serveur de désigner pour eux un processeur adapté à la requête. Les stratégies d'appariement distribuées sont caractérisées par une répartition du choix du site destinataire. Les stratégies distribuées permettent d'éviter l'inconvénient majeur des stratégies centralisées, à savoir la formation d'un goulot d'étranglement lorsque la taille du système devient importante. La recherche d'un site destinataire peut s'effectuer en aveugle sans connaissance de l'état des autres noeuds. Elle peut aussi être faite sous forme de sondages en interrogeant un certain nombre de noeuds pour choisir le plus adapté. Il est aussi possible d'effectuer une recherche exhaustive ou de lancer des enchères.

3.6 Intégration des politiques

Ainsi l'ensemble des cinq types de politiques choisies permet de définir un équilibrage de charge. La combinaison de ces politiques est cruciale mais non aisée. En particulier comme les politiques ont de nombreuses interactions entre elles, le choix d'une stratégie à une étape donnée amène des contraintes sur le choix des politiques des autres étapes. Comme nous l'avons vu, quelque soient les techniques employées, l'équilibrage dynamique impose en permanence un sur-coût de travail qui doit être estimé en regard des améliorations qu'il peut apporter. Un bon mécanisme d'équilibrage de charge doit donc être adapté au système qu'il contrôle.

3.7 Propriétés comportementales d'un équilibrage de charge

Le comportement est défini par la manière qu'un équilibrage de charge améliore l'exécution des tâches. Les seuls choix qu'il puisse prendre sont de déplacer une tâche vers tel ou tel autre processeur ou ne pas la déplacer du tout. Il y a trois principales propriétés qui permettent de distinguer la qualité de différents équilibrages de charge.

Ce sont donc ce que l'on cherche à améliorer lorsque l'on modifie un équilibreur de charge:

- Efficacité: La qualité principale d'un équilibreur de charge doit être de pouvoir utiliser la puissance de la machine à son maximum.
  Cela implique que les processeurs doivent passer le minimum de temps oisif (lorsqu'il y a du travail de disponible). En général il est possible de vérifier l'efficacité d'un équilibrage de charge en observant directement la proportion de temps passé oisif pour chaque processeurs.

- Équité un instant donné, les tâches ayant les mêmes priorités doivent se voir attribuer le même temps d'exécution.

  En corollaire, le temps accordé doit être fonction croissante de la priorité. Sur un processeur donné c'est l'ordonnanceur qui se charge d'assurer l'équité entre les tâches.

  Cependant il n'a pas les moyens d'ajuster le temps octroyé à chaque tâche en fonction de ce qui se passe sur les autres processeurs. C'est à l'équilibreur de charge d'assurer l'équité inter-processeur.

  L'observation de cette propriété peut consister à comparer les temps d'exécution de deux tâches avec la même priorité et le même travail, si l'équité est bonne ils sont toujours très proches l'un de l'autre.
- Localité Les tâches qui ont une affinité particulière pour un ou plusieurs processeurs doivent y être affectées en priorité. Typiquement sur les ordinateur SMP classiques cette localité peut être temporaire parce que la tâche a des données dans le cache mémoire.

  Sur les systèmes NUMA (Architecture à mémoire non uniforme) une tâche peut s'exécuter plus rapidement sur un groupe de processeurs car les temps d'accès mémoire y sont plus courts.

  Dans ARTiS il y a aussi une affinité spécifique entre les tâches RT1+ et les CPU RT, parce que les temps de latence y sont meilleurs. Cette propriété de l'équilibrage de charge implique en général une réduction du nombre de migration et aussi, pour une tâche donnée, un ratio de temps d'exécution supérieur sur certains processeurs que sur d'autres.

Après avoir eu une vue d'ensemble de l'équilibrage de charge nous allons nous intéresser de près aux différents algorithmes existants dans l'état de l'art de l'équilibrage de charge.

3.8 Algorithmes d'équilibrage de charge

Les algorithmes de l'équilibrage de charge se distinguent par leurs objectifs. Ils peuvent être utilisés dans un système pour répondre à différents critères d'efficacité. Leurs utilisations la plus courantes consiste à optimiser l'utilisation des ressources de calcul (partage de charge) et/ou le temps de réponse (équilibrage de charge) d'une application. Ces objectifs peuvent être complétés par des contraintes temporelles plus strictes dans le cadre d'applications temps réels ou des contraintes de tolérance aux pannes modifient le comportement du régulateur de charge. Afin d'étudier ces differents algorithmes, on va les appliquer pour résoudre un problème qui a toujours suscité un très vif intérêt dans differents domaines de recherche, c'est de pouvoir gérer l'allocation de la charge produite par l'exécution d'une application sur les differents processeurs d'une architecture. De nombreux algorithmes existent dans la littérature [37]:

3.8.1 Algorithme aléatoire

Cette stratégie utilise un seuil critique. Lorsque la charge atteint ce seuil, tout nouveau processus sera envoyé vers un autre processeur tiré au hasard. Cette stratégie ne nécessite aucune connaissance distante ou globale des charges des autres noeuds, cet aspect permet une grande simplicité de mise en oeuvre.

3.8.2 Algorithme pair

C'est un algorithme coopératif [20]. Les processeurs échangent des processus deux à deux. Chaque processeur maintient 2 listes :

- Une liste contenant les processus résidant sur le processeur à cet instant,
- L'autre contient les processus qui vont lui être envoyés mais qui n'ont pas encore été reçus.

La stratégie est la suivante :

- Un processeur A envoie une requête à un processeur voisin B capable d'entreprendre des migrations de processus, afin de former une paire temporaire.
- La réponse a deux objectifs, d'une part accepter la liaison, et d'autre part fournir sa liste de processus actifs en indiquant le temps passé et en estimant le temps restant.
- Le processeur contacté a trois solutions pour répondre à la requête du processeur A :

  1. Il rejette la proposition du processeur A qui enverra une requête à un processeur voisin.
  2. Il accepte la relation avec A, ceci implique qu'il refusera toute autre relation tant qu'elle ne sera pas rompue.
  3. Il suspend A, car il est entrain de migrer des processus. A doit attendre que B répond à sa requête en acceptant ou en refusant.

- Après établissement de la relation, le processeur avec la plus forte charge (A par exemple) sélectionne un de ses processus.
- Après calcul, si aucun des processus n'a pu être sélectionné, A informe B de la rupture de la relation. Au contraire, si un processus est élu, la migration peut se réaliser et ceci est répété jusqu'à ce qu'on obtienne plus de gain en migrant un processus.

3.8.3 Algorithme vecteur

C'est un algorithme simple préemptif, présenté par Barak [74]. Il n'utilise pas de connaissance a priori sur les ressources nécessaires. Chaque noeud maintient des informations sur la charge d'un petit nombre fixé de noeuds. Cette information est fréquemment échangée en suivant un cheminement aléatoire. Cet algorithme est coopératif et

son but est de réduire la variance de la charge entre les différents noeuds du système. Nous supposons que le système possède N noeuds et qu'ils sont reliés deux à deux par des liens directs. L'algorithme est constitué de trois parties :

- L'acquisition d'information locale : est réalisée périodiquement et elle est enregistrée dans le premier élément au vecteur L_0. Les autres éléments sont des valeurs de charge d'un sous-reseau de noeuds.
- La stratégie d'échange de messages : est responsable de la mie à jour du vecteur L. L_j est la charge du processeur j $(0 \leq j \leq l-1)$. Des parties du vecteur L sont fréquemment échangées en suivant un cheminement aléatoire. L'algorithme est conçu de telle manière que les informations reçues remplacent les informations les plus anciennes. La stratégie d'échange de messages fonctionne périodiquement et effectue :
  - Une mise à jour de sa propre charge.
  - Un choix aléatoire d'un noeud.
  - Un envoi de la première moitié de son vecteur de charge, noté Lr au noeud choisi.
  - La réception d'un demi-vecteur.
- La fonction de migration d'un processus : cette partie effectue :
  - Une estimation de la charge des autres noeuds.
    Pour cela, il entrelace le demi-vecteur reçu avec son propre vecteur, de la manière suivante :

$$L_{2i} = L_i \quad avec \quad (1 \leq i \leq \frac{l}{2} - 1)$$

    et

$$L_{2i+1} = Lr_i \quad avec \quad (0 \leq i \leq \frac{l}{2} - 1))$$

  - Un calcul estimant le temps de réponse des noeuds à partir de valeurs de charge reçues.
  - Une migration de processus au noeud qui a le temps de réponse minimal.

La performance obtenue dépend de la taille l du vecteur. Lorsque la taille de vecteur est le nombre de noeuds du système, alors l'algorithme est optimal globalement, si le coût de communication est nul. L'algorithme ne fonctionne pas efficacement lorsqu'un grand nombre de processus survient simultanément.

Plusieurs adaptations ont été développées depuis l'algorithme original:

l'utilisation de deux charges par noeud, l'une est locale, l'autre est la charge exportée d'un voisin direct.

La différence entre ces deux valeurs permet d'établir une meilleure stabilité dans le système, et la décision de migrer est réalisée dés qu'une information de charge est reçue.

3.8.4 Algorithme Drafting (ou double seuils)

Cet algorithme est préemptif et indépendant de la topologie. En effet, ce sont les processeurs libres qui offrent du temps de calcul aux processeurs surchargés. Le transfert d'un processus s'effectue en cours d'exécution, ceci signifiant qu'un mécanisme de migration est nécessaire. L'avantage de cet algorithme est de pouvoir s'implanter sur des architectures point à point, l'architecture en bus (réseau local) étant considérée comme une connexion point à point complète. L'algorithme permet donc de recevoir des processus migrés provenant d'un processeur surchargé voisin. Il paraît clair que le phénomène de propagation d'un processus est très faible, donc cet algorithme ne convient pas parfaitement à une machine multiprocesseur ne possédant que peu de points d'entrée. L'évaluation de la charge est réalisée selon la méthode du double seuil. Il y a un seuil de sous-charge S_{min} et un seuil de surcharge S_{max}. Ce mécanisme crée un état intermédiaire de charge normale qui évite à un processeur l'effet d'oscillation entre les deux états extrêmes. Chaque processeur posséde une table de l'état de charge de tout son voisinage. A chaque changement d'état du processeur, celui-ci informe ses voisins par un message de diffusion. Lorsqu'un processeur devient déchargé (drafting processor), il envoie une requête de travail à son voisinage direct surchargé. Chaque processeur recevant cette requête évalue l'opportunité de migrer ses processus éligibles age. A partir de cette suite de valeurs, l'algorithme applique une fonction nommée draft-age qui estimera l'intérêt du processeur à effectuer des migrations. Ce résultat est renvoyé au drafting processor qui choisira le processeur qui aura renvoyé le meilleur draft-age.

3.8.5 Algorithme du Gradient

Cet algorithme est préemptif et indépendant de la topologie. Il équilibre la charge en faisant migrer des processus le long d'un gradient de charge. Un potentiel qui est fonction du taux d'occupation est associée à chaque processeur. Les noeuds les moins occupés ont un potentiel faible. Le réseau peut donc être vu comme un paysage vallonné où les processus migreraient des points les plus hauts vers les points les plus bas. Lin [64] a conçu une version décentralisée où la valeur du gradient représente la distance au plus proche processeur inactif. Dans un premier temps, chaque processeur détermine son état local en fonction d'un seuil minimal et d'un seuil maximal. Il peut prendre trois états qui sont :

- Sous-chargé, la charge est inférieure au seuil minimum.
- Neutre, la charge est comprise entre les deux seuils.
- Surchargé, la charge est supérieure au seuil maximum.

Dans un second temps, un calcul de proximité w(i) de processeurs sous-chargés est déterminé, à partir de la charge locale g(i) et la distance minimale par rapport à un processeur k libre distance(i, k).

$$W(i) = min\{\forall k / g(k) = 0; distance(i,k)\}\}$$

Ce sont ces valeurs de proximités qui constituent la surface de gradient (pente). A fin d'approcher le plus possible la valeur théorique de proximité, une valeur de propagation est utilisée :

$$PP(i) := min\{g(i),1 + min\{j/distance(i,j) = 1; PP(j)\}\}$$

Lorsqu'une surface de gradient est stables $PP(i) = w(i)$ Lorsqu'un noeud est inactif, il envoie une requête à ses voisins directs en transmettant son PP nul. Si un voisin est surchargé, un processus est migré vers l'initiateur, sinon la requête est propagée à des noeuds plus éloignés. Un défaut de cet algorithme est lié au nombre très important de messages qui circulent dans le réseau, d'autres défauts apparaissent mois clairement, mais l'ancienneté des informations et l'inondation d'un processeur par des processus distants sont inévitables en cas de charge importante.

3.8.6 Algorithme des enchères

Cette technique a été crée par J. Ferber [31], puis d'autres algorithmes furent présentés par Smith [87], Hwang [43], Stankovic [89] et Ramamritham [79]. Des processeurs coopèrent en utilisant la méthode des enchères : Les demandeurs de ressources envoient une annonce de tâche, les processeurs intéressés répondent par une offre et, lorsque le demandeur et le receveur se sont mis d'accord par un contrat, la migration peut avoir lieu. Le concept utilisé avec cette technique est la négociation entre agents coopératifs du système, et soumission d'enchères pour contrat. Une négociation avec enchéres a 4 caractéristiques :

1. Le processus est local et n'implique pas de contrôle centralisé,
2. Les informations sont bidirectionnelles,
3. Chaque composant de la négociation évalue l'information avec son propre jugement,
4. L'agrément final est donné après un vote.

Une offre est un message qui peut être :

- Le temps pendant le quel la machine distante a été libre [76].
- L'âge des processus.
- La taille des processus [89].
- La rapidité du processeur [43].
- Le nombre des tâches que le noeud offrant peut garantir [79].

Dans la technique des enchères, les décisions prises sont de type heuristique et utilisent l'historique de l'activité du systéme. Cette méthode est utilisée pour sélectionner un processeur distant qui pourra être un processeur cible pour une migration de processus. On ne posséde pas d'informations sur l'état des autres processeurs ou d'informations ne permettant pas de faire un choix. L'efficacité de ce type d'algorithme dépend des informations échangées durant la négociation.

Dans l'algorithme de Stankovic et Sidhu, qu'utilise également la technique des enchères, les décisions prises sont du type heuristique et sont basées sur l'heuristique de l'activité du système. L'algorithme utilisé est du type adaptatif. L'heuristique utilisée privilège les informations connues sur les caractéristiques des processus. Ces caractéristiques sont l'état du processus, sa taille, a priorité, ses besoins de ressources, le nombre de fois qu'il a déjà migré, sa position dans la file d'attente de ressources, ses besoins en ressources spéciales, les contraintes et le besoin des groupes centralisés et distribués.

3.8.7 Algorithme optimal distribué

Cette stratégie ne peut être réellement mise en oeuvre, mais elle constitue une base de référence. Son objectif est de conserver une différence de charge entre les processeurs au plus égale à un processus. Quand un nouveau processus arrive, il doit être placé sur le processeur le plus faiblement chargé. Lorsqu'un processus se termine, la charge doit être rééquilibrée, si nécessaire, en migrant éventuellement un processus depuis le processeur le plus chargé vers celui sur lequel le processus vient se terminer.

3.8.8 Algorithme centralisé

Le modèle M/M/k est représenté par une file d'attente commune à k serveurs. Cependant, les résultats théoriques servent de comparaison avec les stratégies distribuées.

3.8.9 Comparaison entre les différents algorithmes

Nous présentons une comparaison (Tab.3.1) entre les différents algorithmes afin de situer, particulièrement l'algorithme des enchères que nous utiliserons son principe, dans le chapitre 5, afin de développer notre modèle d'interaction dédié à notre système multi-agents.

3.9 Les Clusters et l'équilibrage de charge

Un cluster est un groupe d'ordinateurs qui travaillent ensemble pour exécuter un jeu commun d'application, et fournir l'image d'un seul système pour le client et l'application. Les ordinateurs sont connectés physiquement par câbles et connectés par des programmes via des logiciels de cluster. Ces connections permettant aux ordinateurs d'utiliser le mécanisme de basculement et l'équilibrage de charges qui ne sont pas disponible avec un ordinateur autonome. Le cluster répond à plusieurs problématiques:

- Haute performance,
- Haute disponibilité,
- Scalability,
- Tolérance aux pannes,
- Et surtout l'équilibrage de charge.

Algo/Critères	Modèle d'information distance	Modèle d'information distance	Extensible (Scalable)	Nb de Communications locale
Aléatoire	Un seul seul de processus	-	Oui	0
Paire	Nb de processus	Un processeur au hasard	Oui indépendant	2 messages en surcharge
Vecteur	Nb de processus	Nb de processeurs d'un sous-ensemble de processeurs	Oui	2 Messages périodiquement
Drafting	2 seuils de processus	Ensemble des voisins directs	Oui, mais la charge se propage difficilement	Nb voisins périodiquement
Enchères	Nb de processus	Seuil de processus à l'ensemble des processus	Oui, mais c'est limité à K	$2(n-1)$ $2k$ au plus
Gradient	Nb de processus	Proximité de site sous-chargé	Oui, mais la charge se propage difficilement	Nb voisins lors de transmission de l'état
Optimale	Nb de processus	Nb de processus de tous les processeurs	Oui si le coût de Comm. est nul	$2(n-1)$

TAB. 3.1 – *Comparaisons entre les différents algorithmes cités.*

3.9.1 Le choix d'un cluster Linux?

Les premiers à avoir exploités des clusters Linux intégrant jusqu'à plusieurs dizaines de PC ont été les laboratoires universitaires, recherchant une alternative aux supercalculateurs. Depuis la maturation de cette technologie, les clusters fonctionnant sous Linux ne sont plus limités aux calculs scientifiques, elles étendent leurs champs d'actions dans les entreprises à équilibrage de charge pour les sites web et à la haute disponibilité. Le fonctionnement en mode Linux a donné naissance à de nombreuses solutions logiciels, nous en citons :

- Beowulf: mise en point par la Nasa : Pour le cluster haute performance.
- LifeKeeper et Kimberlite : Pour le cluster haute disponibilité.
- LVS (Linux Virtual Server): Concernant l'équilibrage de charge.

Notons que le nouveau système Linux dédié à ce genre de situation, appelé Parallel Linux, permet de mettre en pratique des clustering qui permettent de réaliser de bon

systèmes d'équilibrage de charge.

3.10 Le choix de la plate-forme

Le choix de la plate-forme sur laquelle l'application d'équilibrage de charge doit être exécutée [15], ou du moins simulée et validée, est fondamental. Le fait que l'architecture du système en question est distribuée, généralement deux solutions se présentent devant nous dans notre cas:

- Implémenter le système sur une architecture réellement parallèle.
- ou bien Simuler l'exécution sur :
  - Un système d'exploitation multitâche à part entière (genre Linux), où chaque agent associé à sous système sera associé à un processus (au sens informatique du terme) durant l'activation de l'agent).
  - un système Windows-xx par multithreading où l'activation de chaque agent est simulé par un thread,
  - Une plate-forme réseau windows-NT ou parallel Linux par exemple, où chaque agent s'exécute carrément sur un poste.

3.11 Conclusion

L'objectif des politiques de l'équilibrage de charge est de répartir équitablement la charge du système sur les différents sites. Le principal problème avec ces politiques concerne le choix d'un seuil de charge admissible, ainsi qu'une politique à mettre en oeuvre. Certains travaux ont montré l'avantage d'avoir un seuil variable de charge, dépendant de l'état globale du système "une faible charge globale diminue le seuil de charge locale, alors qu'une forte charge globale l'augmente".

Nous verrons que dans le cas de notre thèse qu'une étude sur le seuil est fondamental pour l'amorçage du processus de l'équilibrage de charge. C'est que nous verrons dans le chapitre 4.

Dans notre cas, l'algorithme des enchères constitue pour nous une bonne inspiration pour la réalisation d'un modèle d'interaction associé au modèle d'organisation basé sur le paradigme des Systèmes multi-agents que nous présenterons dans le chapitre 5.

Pour la l'implémentation de ce genre de problème, La tendance actuelle dans ce domaine est le clustering réel ou simulé pour les problèmes de calculs complexes, les réseaux locaux et le réseau Internet pour les problèmes de e-maintenance, de e-commerce, etc. Ils se basent principalement sur l'échange des messages entre les différents noeuds (sites).

Pour cela, l'utilisation d'un outil de communication et un support pour véhiculer les informations et mêmes les charges, en terme, processus de calcul, entre sites afin d'assurer l'équilibrage de charge est indispensable. Ils sont appelé communément middlewares, et est sont les principaux supports dans genre de situations. Differents middlewares sont

disponibles, nous en citons : Corba, RMI, DCOM, XML-RMC et récemment SOAP que nous présenterons en chaptire 5.

Chapitre 4

Les ressources distribuées: vers un équilibrage coopératif préventif

Dans ce chapitre, nous présentons notre démarche conceptuelle en matière d'organisation en "agents" et interagissant dans un système. Elle consiste à réaliser, pour une entreprise étendue via une e-plate-forme , un équilibrage de ressources basé le principe de coopération entre les différents sites du système afin de prévenir des situations de dépassement de seuil en matière de stockage de ressources.

4.1 L'entreprise étendue, les TIC et la gestion de ressources

L'entreprise étendue est désormais une réalité. C'est celle qui s'étend au delà des frontières organisationnelles " physiques ", elle est caractérisée par une externalisation d'activités et par le développement de partenariats, par le biais des Technologies de l'Information et de la Communication (TIC) qui en sont le support. Les stratégies d'externalisation consistent pour l'entreprise à se procurer auprès des fournisseurs des services ou produits qui étaient auparavant assurés localement. Ces stratégies conduisent ces entreprises à s'associer avec d'autres entreprises dans une logique de complémentarité de ressources.

La gestion des ressources dans une entreprise étendue est à la fois:

- Une gestion des stocks avec l'établissement de prévisions, des approvisionnements, une gestion de "un ou plusieurs magasins, appartenant à un ou plusieurs sites", des transports, gestion de nomenclatures, donc un ensemble d'activités logistiques classiques, où les règles et les algorithmes de la gestion de stock classiques peuvent être appliqués.
- Une activité technique de mise en oeuvre d'une politique de maintenance, de e-maintenance, particulièrement préventive, d'un ensemble industriel, composé d'un site ou d'un ensemble de sites interconnectés entre eux par le biais d'une plate-forme de e-maintenance, de e-commerce, où l'équilibrage de la disponibilité des

ressources entre ces différents sites constitue la condition nécessaire pour le bon fonctionnement de l'ensemble.

- Une politique préventive entre les différentes entités de l'entreprise qui sont directement concernées par la gestion de leurs ressources d'une façon "coopérative", en assurant un équilibrage en cette matière pouvant éviter le sous-stockage ou le sur-stockage.

La modélisation et l'automatisation de cette gestion un site local est actuellement une tâche très aisée. Une pléiade de solutions dédiées, commerciales et même gratuites, est proposée actuellement.

Mais les solutions dédiées aux entreprises étendues restent des solutions de " tenue de stock " et non pas de "gestion de stock ". Car une gestion de stock des ressources pour une entreprise, dont les sites, interconnectée par Internet par exemple, nécessite l'étude, la modélisation et la validation d'autres critères (communication, interaction, coopération, négociation, e-commerce, etc.).

Le "zéro stock" est un abus de langage car, le stock est indispensable, mais un stock dépassant un seuil non toléré affecte négativement l'entreprise. Un sous-stockage provoque une rupture de stock, ce qui peut générer l'arrêt de la machine qui peut à son rôle altérer la chaîne de production.

Par contre, un sur-stockage provoque des coûts inutiles (assurances, locaux, gardiennage, vieillissement, etc.) La pièce de rechange constitue la ressource la plus importante, sa gestion revient à gérer le stock en quantité stratégique juste suffisante pour répondre au besoin du bon fonctionnement de l'entreprise. Cela revient donc à chercher un stock optimal S*, pour toute l'entreprise étendue, au sens coût et disponibilité sur une période T, afin de ne pas l'exposer à une situation anormale. Notre contribution s'inscrit justement dans cet axe. C'est une contribution à la modélisation et à une méthodologie d'implémentation, par l'approche des Systèmes Multi-Agents (SMA) du problème posé. Car, cette approche, a fait sa preuve pour la modélisation des systèmes distribués complexes. Ce qui va permettre de gérer, de façon optimale, l'équilibrage des ressources distribuées. En utilisant une politique préventive, pouvant apporter à l'entreprise étendue la disponibilité continue et optimale des ressources, donc de la valeur ajoutée.

4.2 Les ressources dans une entreprise étendue

4.2.1 Le cas de pièce de rechange

Identification d'une pièce de rechange

L'identification est un point crucial dans le traitement de la pièce de rechange. Dans ce cadre, la documentation est le point essentiel qui permet d'identifier cette ressource. Elle doit reposer sur des documents très fiables, informatisés ou non, tels que; dessins, photographies, schémas et qui permettent de :

- trouver l'identification d'une pièce sur le matériel,
- identifier une pièce à vue hors matériel,

- savoir sur quels matériels est montée une pièce à partir de son identification,
- reconnaître une pièce sur le matériel à partir de son identification (repère topo-fonctionnel et éclaté),
- connaître le nombre d'exemplaires d'une pièce sur un matériel,
- connaître le nombre de matériels sur le site qui utilisent une pièce,
- connaître les pièces strictement équivalentes, interchangeables ou adaptables (avec les contraintes et limites d'utilisation),
- déterminer l'ensemble supérieur à laquelle appartient cette pièce,
- trouver éventuellement ses caractéristiques fonctionnelles, dimensionnelles économiques et techniques - y compris ses plans - particulièrement dans le cas de pièces spécifiques à un équipement réalisé par l'utilisateur,
- trouver ses caractéristiques logistiques (réparables ou non, degré et niveau de maintenance, gamme de dépose, pose, test et réparation, conditions de stockage, durée de validité),
- identifier le ou les fournisseurs de cet article et surtout les autres partenaires de l'entreprise étendue avec lesquels la coopération est possible et avoir accès à leur catalogue, particulièrement sur Internet.

Cette documentation est régie par la norme AFNOR X60-210.

4.2.2 Les indicateurs de la disponibilité

Chaque fois qu'un équipement ou un sous ensemble de cet équipement tombe en panne, il faut un certain temps pour le remettre en état. La vie d'un équipement ou d'un article est donc composée de temps de bon fonctionnement et de temps de réparation. Les temps de réparation ne comprennent pas les temps de maintenance préventive.

A partir de ces définitions et en analysant la vie d'un équipement, on peut considérer des indicateurs :

- MTBF (Mean Time Between Failure) : moyenne des temps entre deux défaillances.
- Taux de défaillance: (inverse du MTBF) représenté par la lettre λ : Moyenne du nombre de défaillances par unité de temps (non compris les temps de réparation)
- MTTR (Mean time To Repair) : moyenne des temps de réparation:

$$MTTR\ (en\ heures) = \frac{Total\ temps\ reparation}{nombre\ de\ pannes}$$

- Disponibilité : aptitude du système, sous les aspects combinés de sa fiabilité et de sa logistique de maintenance, à remplir une fonction à un instant donné ou dans un intervalle de temps donné:

$$Disponibilite = \frac{MTBF}{(MTBF + MTTR)}$$

4.2.3 Classification de la Pièce de Rechange dans une entreprise étendue

Nous pouvons appliquer la classification dite de ABC, basée sur le principe de 20/80 de Pareto à cette ressource. C'est une classification qui peut s'appliquer selon les prix, les quantités utilisées, ses fréquences d'utilisation, les quantités minimales d'achat, les délais. Mais surtout leur impact en cas d'indisponibilité, etc. :

Classe A, appelée aussi classe d'Articles à Rotation Lente (SMI) [25] : Articles très chers, rares, délais longs. Ils ne doivent pas être disponibles sur tous les magasins de maintenance. Ils peuvent être mis en disponibilité commune, dans un site bien particulier, pour l'ensemble des sites de l'entreprise.

Classe B, appelée aussi classe d'Articles à Rotation Moyenne (NMI) [25] : Articles moyennement chers, disponibilités aléatoires sur le marché. Doivent être dans des sites bien particuliers et accessibles, au moment opportun, pour l'ensemble des sites de l'entreprise étendue.

Classe C, appelée aussi classe d'Articles à Rotation Rapide (FMI) [25] : Articles courants, généralement peu chers. Ils peuvent être disponibles dans tous les magasins de l'ensemble des sites, mais en quantités optimales.

La gestion d'un article est spécifique à la classe à laquelle il appartient. La politique de réapprovisionnement en découle étroitement. C'est en fonction de cette classification que les réponses aux questions suivantes doivent être trouvées:

- Quoi réapprovisionner ?
- Quand réapprovisionner ?
- Combien réapprovisionner ?

4.2.4 Politiques de réapprovisionnement pour un site donné

Selon la période T et la quantité de stock S prévue. Quatre grandes politiques sont à prendre en considération :

1. T fixe et S fixe:

   Cette méthode dite " calendaire " prévoit des livraisons des articles à des dates fixes. C'est une méthode qui s'applique généralement pour les articles de faible valeur de la classe C, quand la consommation est surtout régulière.

   Avantages :

   - Simplicité de la gestion de l'article,
   - Gains d'échelle négociables par les acheteurs.

   Inconvénients :

   - Simplicité de la gestion de l'article.

2. T variable et S fixe :
   Cette méthode dite de " point de commande " consiste à définir le niveau de stock, appelé point de commande ou seuil de commande, qui déclenche l'ordre d'achat de façon à être livré juste au moment de l'utilisation du dernier article.
   Avantages :
   - Méthode qui permet d'éviter la rupture de stock
   - C'est une méthode qui est adaptée surtout aux consommations partiellement irrégulière.

   Inconvénients:
   - Suivi permanent des stocks.
3. T variable et S variable :
   Cette une méthode qui est utilisée principalement pour les articles de la classe A, dont le prix de revient varie fortement ou dont la disponibilité n'est pas permanente. C'est une commande opportuniste.
   Avantages :
   - Méthode qui permet de profiter de tarifs intéressants.

   Inconvénients :
   - Faire un suivi permanent afin de guetter les prix les plus intéressants,
   - Applicable surtout pour un type d'article bien particulier, particulièrement ceux de la classe A.

Il en découle que la politique de stockage et de réapprovisionnement d'un article P_i d'un site S_i dépend surtout de sa classification.

4.2.5 Le cas d'une entreprise étendue

Les politiques de réapprovisionnement que nous avons citées présentent la stratégie à adopter vis-à-vis de chaque article, après étude au préalable (la demande, coût, période, etc.), mais s'appliquent surtout pour un site autonome.

La politique de réapprovisionnement dans une entreprise étendue dépend des besoins de l'ensemble des sites interconnectés entre eux. Les articles de la classe C, par exemple, peuvent se retrouver dans tous les sites, mais les articles supposés appartenir aux classes B et C peuvent être disponibles dans des sites particuliers, considérés importants par les gestionnaires de l'entreprise étendue.

Le principe de gestion, dans ce cas, est de maintenir un équilibrage, en disponibilité en premier et en coût plus tard, entre tous les sites et pour tous les articles dont il a besoin chacun de ces sites.

Pour ce faire, il s'agit de trouver les solutions adéquates, conceptuelles et implémentatives, afin de prévenir et éventuellement d'éviter un déséquilibre de disponibilité de ces ressources dans un contexte multi-sites distribué. Avec la contrainte que chacun des sites offre et demande des ressources différentes. Sachant que chaque ressources obéit à des critères différents de coût, d'importance en cas de panne, etc.

4.3 Modélisation

4.3.1 Pourquoi une modélisation Système Multi-agents?

– L'architecture d'une entreprise étendue est distribuée et communicante. Elle est composée de N sites. Chacun d'eux, à son tour, peut être composé d'un ou de plusieurs autres sous sites. Sachant que l'ensemble est concerné par l'utilisation de la pièce de rechange. Chaque site est autonome par la politique adoptée vis-à-vis de la gestion préventive des articles.

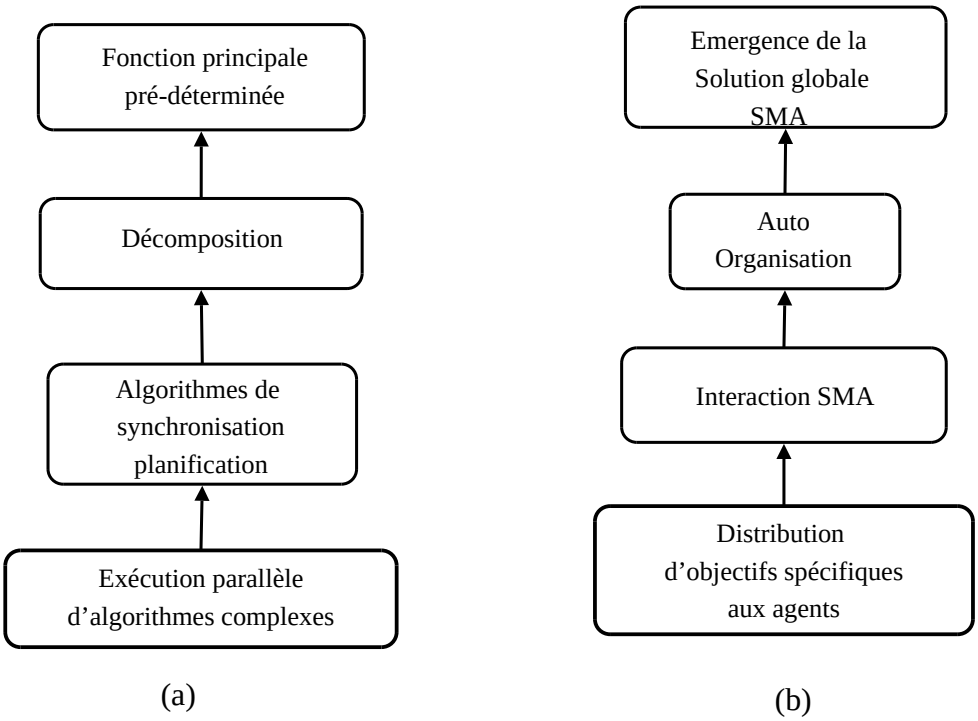

FIG. 4.1 – *Méthode classique VS Emergence de solution globale dans les SMA*

– L'interaction entre les sites du système est évidente, afin de réaliser, par coopération, l'objectif assigné par le système.
– Cet objectif est dans ce cas: un équilibrage de ressources du système, ce qui exige donc une approche basée sur :
  – La coopération entre les sites du système, pour l'émergence de la solution qui est l'équilibrage de la disponibilité des ressources de l'entreprise étendue.
  – La dynamique du système, l'équilibrage continue au fil du temps.

C'est en se basant sur toutes ces caractéristiques que la modélisation du problème posé par l'approche SMA s'impose.

4.3.2 Agent et SMA

Nous avons présenté au chapitre 1 tout le paradigme des SMA et au chapitre 2 l'interaction dans les SMA. Nous considérons maintenant que le lecteur, et sur cette base, possède une idée de ce qu'est ce paradigme.

Cependant, il faut rappeler, qu'il n'existe pas une définition adoptée par la communauté SMA sur le terme "agent". En revanche, tout le monde s'accorde à dire que la notion d'autonomie est au centre de la problématique des agents. Pour notre part, nous considérons que J. Ferber [28] [29] a donné une définition académique consistante à cette notion.

Pour notre cas d'étude, nous disons qu'un agent est une entité logicielle, située dans un environnement, dotée d'un comportement autonome lui permettant d'atteindre, en interagissant avec cet environnement, les objectifs qui lui ont été assignés à sa conception.

Par ailleure, un SMA est un ensemble d'agents répondant à l'architecture d'un système, ainsi qu'à sa dynamique, obéissant à un modèle d'organisation, et sur la base d'un modèle d'interaction entre les agents de ce système, qui activent pour l'émergence de la solution recherchée.

Dans notre étude, nous prenons acte des mots clés : organisation, autonomie, interaction et objectif qui constituent, à notre sens, les principales caractéristiques que doit avoir la paire (agent - SMA) et qui constituent la raison de la dynamique du système afin qu'il remplisse l'objectif recherché.

Contrairement à l'approche descendante dite approche conventionnelle statique [66], modéliser le fonctionnement d'un problème complexe en SMA, c'est partir de tâches simples distribuées avec objectifs locaux (Fig. 4.1.) afin d'atteindre l'émergence de la solution globale du problème posé.

Dans notre cas, l'objectif local est la disponibilité optimale des articles en quantité, c'est-à-dire ni sous-stockage ni sur-stockage, et aussi en coût. L'objectif du système, en tout, est l'émergence d'un équilibrage de ressources entre tous ses sites.

Modéliser en SMA, c'est aussi rechercher un modèle d'organisation adéquat. Il doit être associé à un modèle d'interaction. Dans le but que cela réponde, au mieux, au bon fonctionnement et à la dynamique du système. Car nous considérons qu'il n'existe pas encore de modèle générique pouvant satisfaire tous les critères d'organisation et de fonctionnement d'un problème donné.

4.3.3 Identification des différents agents de notre système

Sur la base des types et le volume de charge des tâches que doit effectuer le système. Une répartition des rôles sur des agents logiciels du système doit être faite (Fig. 4.2.). Pour cela, nous avons pris en considération les différents rôles au sein de chaque site. Sur cette base, nous avons opté pour la répartition des rôles suivants :

1. Agent superviseur.
2. Agent prévention.
3. Agent gestionnaire de stock.

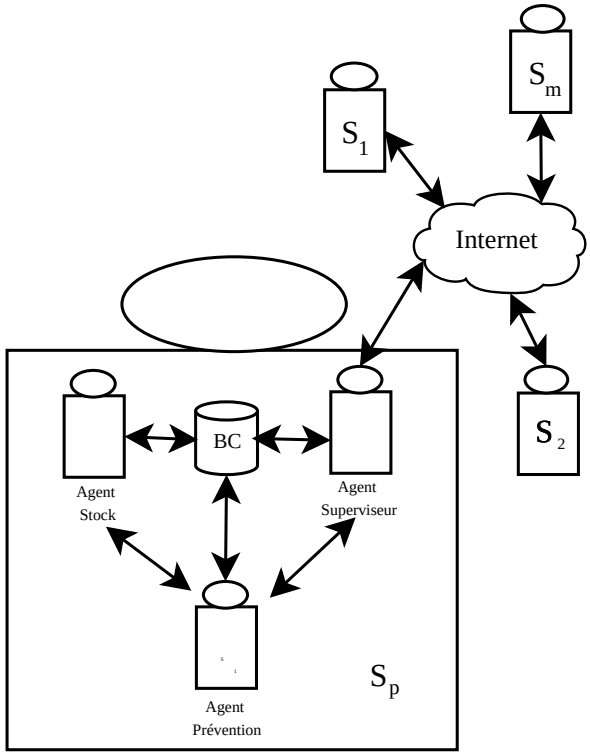

FIG. 4.2 – *La répartition des rôles*

Sachant que chaque site lui-même peut être considéré comme un agent. Car il sera représenté par un interlocuteur vis-à-vis des autres agents du système qui est l'agent superviseur.

4.3.4 Description des différents agents du système

Agent superviseur

C'est l'agent maître du système associé à un site, son rôle (Fig. 4.3.) est de gérer et contrôler le fonctionnement du système. L'agent superviseur décrit l'ensemble des objectifs du système et observe son état.

C'est lui qui communique les quantités entrantes et sortantes d'articles. C'est lui aussi qui lance aussi les requêtes d'acquisition et d'offre des articles. C'est le superviseur qui reçoit et envoie les messages aux autres sites. Il joue donc le rôle d'une interface par rapport aux autres sites. selon le script (Fig. 4.3.).

Agent prévention

L'agent prévention est un agent intelligent, muni d'une capacité d'intelligence lui permettant de décider en fonction de sa base de connaissance (BC).

```
//Je suis l'interlocuteur du site S_p
FAIRE_EN_PARALLELE
  SI "je reçois un message d'offre de Sm pour l'article k" ALORS
       // Entrer en négociation avec S_m
  FINSI
  SI "je reçois un message de demande de Sm pour l'article k" ALORS
     // Entrer en négociation avec S_m
  FINSI
  SI "je reçois un message de demande de Sm pour l'article k" ALORS
       // Entrer en négociation avec S_m
  FINSI
  SI "je reçois un message de sous-stockage de mon agent prévention" ALORS
     // Diffuser la demande dans mon environnement
  FINSI
  SI "je reçois un message de sur-stockage de mon agent prévention" ALORS
     // Diffuser l'offre dans mon environnement
  FINSI
FIN_FAIRE_EN_PARALLELE
```

FIG. 4.3 – *Script de l'agent Superviseur*

Il se base sur des algorithmes calculant les consommations antérieures et d'autres critères de disponibilité, de MTBF, de taux de défaillance, de MMTR, etc. Il a donc la capacité de prévoir les cas de sous-stock ou de sur-stock des quantités d'articles, sur la base des T_{prev} calculés pour chaque article. Il envoie des messages d'urgence à l'agent superviseur en lui indiquant, pour un type d'article, s'il y a sur-stockage ou sous-stockage.

Cet agent consulte donc la base de donnée d'une façon continue et périodique et guette les cas qui peuvent provoquer des situations anormales afin de prévenir et envoyer des messages d'alerte à l'agent superviseur.

Notre étude accorde plus d'importance à l'aspect prévention, donc au script de l'agent dédié à ce rôle, car cela assure une gestion optimale des ressources à l'entreprise étendue dans un cadre coopératif entre tous les sites de cette dernière. Le script de cet agent sera présenté une fois la démarche de calcul présentée (Fig. 4.12.).

Agent gestionnaire de stock

c'est un agent réactif, dépourvu de toute forme d'intelligence, son rôle est la gestion de stock, au sens classique, telles que la consultation de la base de donnée et sa mise à jour. Cet agent reçoit l'ordre de l'agent superviseur.

Il en découle que le système possède des agents cognitifs et d'autres réactifs. L'agent prévention est un agent cognitif, c'est-à-dire qu'il possède une capacité de raisonner sur la base de son passé, qui est la consommation antérieure associée autres informations se trouvant dans sa base de connaissance.

4.3.5 Modèle d'organisation dans un SMA

Un site est représenté par son superviseur, il est l'interlocuteur auprès des autres sites du système. Un groupe d'agents, selon un critère bien défini, est représenté par un agent Coordonnateur des Accointances du Groupe baptisé CAG, auquel s'adressent toutes les requêtes émanant des superviseurs du groupe. Les agents qui appartiennent au même groupe d'agents, se reconnaissent via leurs superviseurs qui communiquent entre eux. C'est une forme d'agence immobilière regroupant toutes les informations sur les requêtes (offre et demande) des agents du groupe. En cas où la requête trouve une réponse favorable chez ce CAG, donc il y a un agent répondeur. Ce CAG met les deux agents en négociation directe.

Négociation basée sur le réseau d'accointances

En modélisant la représentativité d'un site par un agent superviseur associé à ce dernier. S_1, S_2 et S_3 représentent trois instances durant leur activation. Ces dernières peuvent constituer un groupe.

Le CAG a, pour rôle, l'établissement d'une négociation entre agents d'un même groupe après recherche d'agents proposant des coûts optimaux.

En général, un CAG détient toutes les informations concernant les sur-stockages et les sous-stockages des différents articles utilisés par les membres du groupe et envoyés à leur CAG périodiquement, ces informations sont $P_{i,p,surplus}$ et $P_{i,p,deficit}$, sachant que : $i \in E_p$ et E_p représentant l'ensemble des articles associés à S_p, avec:

$$\begin{cases} E_1 \cup E_2 \cup ... \cup E_N = E & \text{et} \\ E = \{E_i\}, i = 1,N \end{cases} \tag{4.1}$$

Sachant que :

- $P_{k,p,surplus}$, la quantité en surplus, considérée comme sur-stockage, en articles numéro k, sur le site S_p, pour une période T.
- $P_{k,p,deficit}$, la quantité en déficit, considérée comme sous-stockage, en articles numéro k, sur le site S_p, pour une période T.

Rôle du CAG

Le rôle d'un CAG_m associé à un groupe de sites (S_1,S_2,S_p,...) est la coordination et la réception des besoins en quantité d'articles des différents sites du groupe, sur-stockage $P_{k,p,surplus}$ et sous-stockage $P_{k,p,deficit}$ pour chaque article k. Il classe ces besoins, optimise les coûts pour chaque article et répond aux demandeurs, selon le script (Fig. 4.4.).

Habituellement, un agent, en cas d'autosuffisance des ressources au sein du même groupe, reçoit ses requêtes essentiellement de la part des membres du groupe. Dans le cas contraire, il peut toujours en recevoir d'autres, émanant d'autres sites appartenant à d'autres groupes, via le Coordonnateur Inter-Groupes, que nous avons appelé le CIG (Fig. 4.5.). Son rôle consiste en la contribution à l'établissement de l'équilibre des ressources

Réceptionner les offres d'appels ( )
Optimiser les coûts ( )
Classer les offres ( )
Répondre aux requêtes émanant des sites selon
le meilleur côut ( )

FIG. 4.4 – *Rôle du Coordonnateur des Accointances du Groupe*

au début, une fois cet équilibre atteint, il lui reste qu'à le surveiller, c'est-à-dire éviter le déséquilibre.

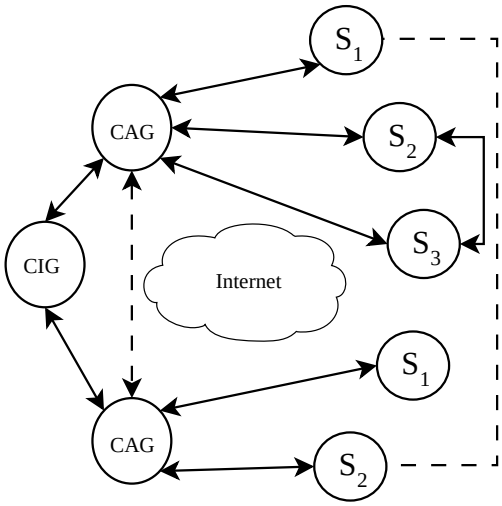

FIG. 4.5 – *Le Coordonnateur des Accointances du Groupe (CAG) et d'Inter-Groupe (CIG)*

Plusieurs demandes émanant de plusieurs demandeurs peuvent concerner le même article à un instant T donné, pour son meilleur coût ou sa rareté et provoquant ainsi une section critique.

La centralisation de la négociation d'un groupe au niveau d'un CAG possède deux inconvénients qui sont :

- goulet d'étranglement,
- risque de panne du coordonnateur d'accointances du Groupe.

Elle présente néanmoins plus d'avantages que d'inconvénients:

- Avoir le maximum d'informations. disponibles à un instant T,
- Éviter les messages de diffusion (broadcast) aux sites du même groupe plusieurs fois pour l'acquisition ou l'offre d'un article quelconque.
- Permet de déléguer des opérations aux autres groupes via le CIG.
- La solution proposée est facilement implémentable.

4.4 Équilibrage coopératif et préventif de ressources

4.4.1 L'objectif

L'objectif de notre SMA au moment des son exécution est la réalisation d'un équilibrage des ressources entre les différents sites du système. Pour ce faire, l'agent prévention de chaque site détecte les situation anormales de sur-stockage et de sous-stockage, et en informe son superviseur qui entre en interaction, par principe de coopération, avec les autres agents superviseurs des autres sites, afin d'assurer cet équilibrage.

Dans ce qui suit, nous montrerons la démarche préconisée afin de réaliser cet objectif.

4.4.2 L'équilibrage de ressources

Supposons qu'un site S_1 dispose d'une quantité de sur-stockage d'un article P sur une période T, que nous notons P+, au moment où le site S_2 accuse un sous-stockage pour le même article, durant la même période, que nous notons P-.

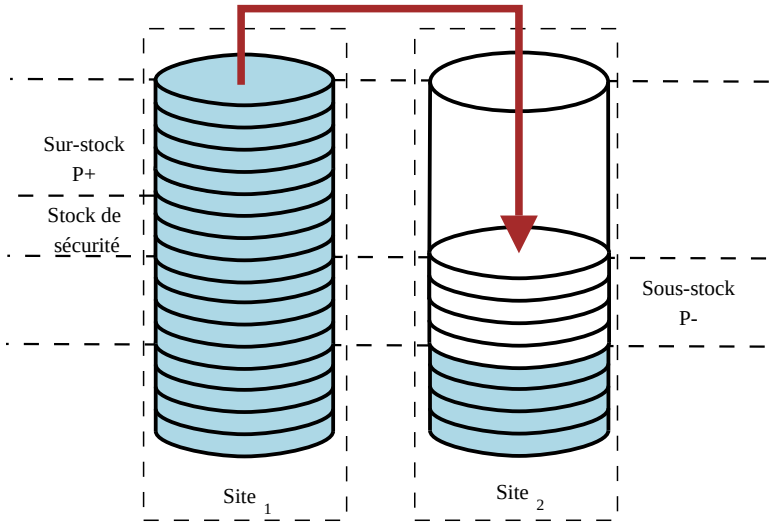

FIG. 4.6 – *Équilibrage entre sites pour une ressource.*

Un équilibrage en cette ressource entre sites S_1 et S_2, serait simplement le transfert d'une quantité de P, à déterminer, de S_1 vers S_2 (Fig. 4.8.(a)).

4.4.3 L'équilibrage coopératif de ressources

Si S_1 dispose d'articles P_1et P_2 avec des quantités P_1+ et P_2- . Et S_2 dispose des mêmes articles, avec des quantités P_2+ et P_1- .

Afin d'équilibrer les deux sites en ces ressources, S_1 doit envoyer à S_2 une quantité de P_1, selon des conditions posées par S_1 et acceptées par S_2. Ce dernier, de son coté, peut en recevoir du P_2 selon d'autres conditions. (Fig. 4.8.(a) et Fig. 4.8.(b)).

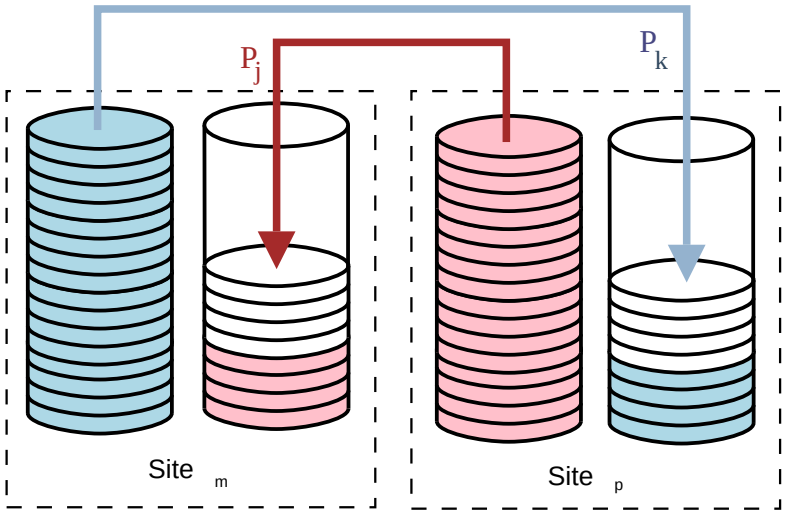

FIG. 4.7 – *Équilibrage entre sites pour plusieurs ressources.*

Le même principe peut se généraliser à l'interaction d'un ensemble de N sites, chacun gérant M articles.

4.4.4 Choix sélectif de ressources

Supposons maintenant que le site S_1 accuse un sous-stockage en articles P_1 et en P_2, alors que deux autres sites S_2 et S3 possèdent des sur-stockages pour les mêmes articles, et proposent les articles P_1 et /ou P_2 au site S_1 à des coûts différents Le problème consiste à optimiser pour S_1 l'acquisition des articles P_1 et P_2.

Plusieurs possibilités lui sont offertes (Fig. 4.9.) :

- acquérir P_2 du S_1 et P_1 de S_2
- acquérir P_1 du S_1 et P_2 de S_2
- acquérir P_1 et P_2 de S_1
- acquérir P_1 et P_2 de S_2

D'une façon générale, un site peut se trouver offrant de M articles et acquéreur de P autres articles en même temps.

Sur cette base, le choix peut varier selon plusieurs critères :

- Le choix peut être porté sur un critère de coût,
- Le choix peut être porté sur le groupe,
- Le choix peut être porté sur un critère géographique,
- Le choix peut être prix sur une durée et sur la base d'un contrat de partenariat,
- Mais, un choix peut être pris pour d'autres considérations, autres que celui du coût, au sens optimal.

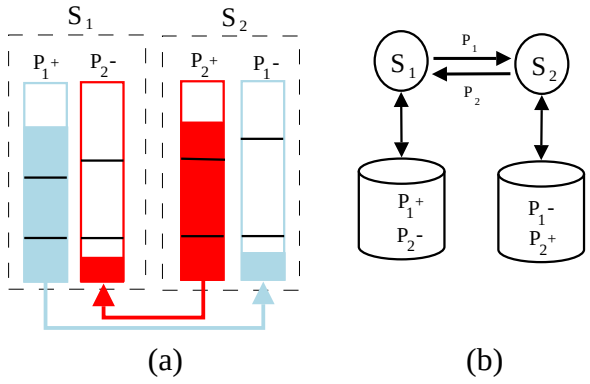

FIG. 4.8 – *Principe de l'équilibrage coopératif de ressources*

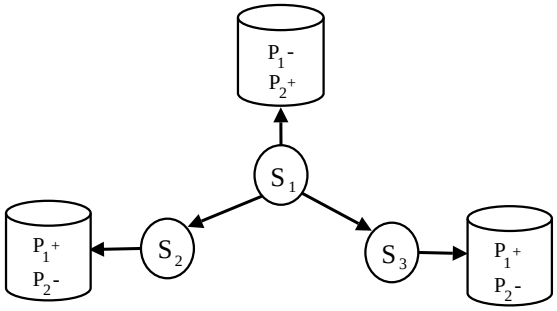

FIG. 4.9 – *Choix sélectif de ressources*

Dans ce qui suit, nous prendrons en considération, en premier, le critère de la disponibilité de l'article, dans son groupe, avant d'opter pour d'autres groupes. Ce critère est fondamental pour la maintenance, car l'indisponibilité d'un article peut provoquer des arrêts de production.

4.4.5 Formulation du problème

Soit un site S_p à un instant T donné, notons : T_{task}, la période associée au déroulement d'une tâche entre deux réapprovisionnements.

$P_{k,p,av}$, la quantité disponible, en articles numéro k, sur le site S_p.

Pk,p,req , la quantité exigée, en articles numéro k, sur le site S_k, pour la période T_{task}.

Par conséquent, on a, pour chaque site S_p:

En attribuant un poids $\rho(k,p)$, associé à l'article k pour le site numéro p, on a :

$$\rho(k,p) = P_{k,p,av} - P_{k,p,req} \tag{4.2}$$

P_1	$P_{1,p,av}$	$P_{1,p,req}$
...	...	...
P_k	$P_{k,p,av}$	$P_{k,p,req}$
...	...	...
P_N	$P_{N,P,av}$	$P_{N,p,req}$

TAB. 4.1 – *Tableau de bord des ressources du site S_p*

Ce poids détermine le niveau de disponibilité de la ressource par rapport au besoin de la tâche en voie d'exécution, qui peut être une maintenance préventive ou non et où la présence de cette ressource est indispensable :

$$av = \begin{cases} \text{surplus} & \text{Si} \quad \rho(k,p) > 0 \\ \text{suffisant} & \text{Si} \quad \rho(k,p) = 0 \\ \text{déficit} & \text{Si} \quad \rho(k,p) < 0 \end{cases} \tag{4.3}$$

Une optimisation du problème consisterait en la vérification de la condition suivante (*) :

$$\rho(k,p) = 0 \tag{4.4}$$

Car cette solution n'est pas optimale dans les deux autres cas :

- Si $\rho(k,p) < 0$, c'est un sous-stockage de la quantité de l'article k, sur le site S_k. Les tâches de maintenance sont affectées. Par conséquent, le bon déroulement du système est affecté.
- Si $\rho(k,p) > 0$, c'est un sur-stockage de la quantité de l'article k, sur le site S_k. C'est un stock inutile engendrant de mauvais facteurs : vieillissement de l'article, coûts du sur-stockage (capital stagné, locaux, assurance, gardiennage).

Mais par ailleurs, gérer l'instantané sur une durée T_{task} qui est la durée nécessaire pour le fonctionnement d'une tâche, à déterminer, revient à appliquer :

$$P_{k,p,av} \geq P_{k,p,req}(T_{current}) \tag{4.5}$$

Il faut signaler qu'économiquement, la condition (*) est très sévère et impossible à réaliser sur le terrain, car on ne peut pas fonctionner au jour le jour. L'idéal est de se fixer une quantité dite quantité préventive, qui permet au système d'être à l'abri de sortir des seuils tolérés. C'est une condition du genre:

$$\rho^*(k,p) = P_{k,p,prev} = P_{k,p,av}(T_{current)} - P_{k,p,req}(T_{prev}) \tag{4.6}$$

La recherche d'une fonction de consommation de chaque article k, pour chaque site, sur un intervalle régulier est nécessaire. Cela reviendrait à trouver une quantité que nous appelons quantité préventive. Elle est calculable, nous la notons $P_{k,p,prev}$. Cette dernière doit être consommée avant T_{prev} et elle va permettre au site d'éviter toute forme de sur-stockage ou de sous-stockage (rupture de stock avant T_{prev}) pour cette même ressource.

4.5 Recherche de la fonction consommation

4.5.1 Fonction basée sur la la consommation de Pareto

Méthode A-B-C

Cette méthode a pour objectif de sélectionner les articles pour lesquels il convient d'organiser en priorité la gestion des stocks (voir également méthode 20/80).

Quand on constate que:

- 10% des articles (groupe A) représentent 60% en valeur,
- 40% des articles (groupe B) représentent 30% en valeur,
- 50% des articles (groupe C) représentent 10% en valeur,

Il convient donc de classer les articles et d'assurer une gestion d'autant plus rigoureuse que les valeurs sont importantes (les produits du groupe A seront donc les mieux suivis, puis les produits du groupe B, enfin les produits du groupe C).

Méthode 20/80

Cette méthode a pour objectif de sélectionner les articles pour lesquels il convient d'organiser en priorité la gestion des stocks (voir également méthode A-B-C).

Quand on constate que:

20% environ des produits référencés représentent environ 80% de la valeur du stock,

on applique à ces produits une gestion complexe et rigoureuse (les 80% en quantité qui ne représentant que 20% en valeur se voient appliquer une gestion beaucoup plus souple).

Ces méthodes de gestion des stocks sont théoriques. Leur application réelle peut varier beaucoup d'une entreprise à une autre

4.5.2 Fonction basée sur la Méthode de Wilson

Wilson a établi une formule basée sur un modèle mathématique simplificateur dans lequel il considère que la demande est stable sans tenir compte des évolutions de prix, des risques de rupture et des variations dans le temps des coûts de commande et de lancement (on dit aussi "en avenir certain").

Les hypothèses du modèle :

- La demande annuelle est connue et certaine.
- La consommation est régulière (linéaire).
- Les quantités commandées sont constantes.
- La pénurie, les ruptures de stock, sont exclues.

A remarquer que, dans ce cas, la gestion du stock s'effectue généralement sur une période annuelle.

Calcul de la quantité économique

En posant :

Q : le nombre de pièces approvisionnées ou lancées en fabrication en une seule fois.

Pu : le prix unitaire de la pièce.

SS : le stock de sécurité envisagé pour cette pièce.

T : le taux de possession de l'entreprise exprimée en pourcentage.

CL : le coût d'approvisionnement ou de lancement en fabrication.

On a les coûts suivants:

$\frac{N}{Q}$: Nombre annuel de lancement

$\frac{N}{Q}.CL$: Coût annuel de lancement

$\frac{Q}{2}+SS$: Stock moyen dans l'entreprise (en supposant que la consommation est régulière)

$(\frac{Q}{2}+SS).t.Pu$: Coût annuel de possession

$Ct = \frac{N}{Q}.CL + (\frac{Q}{2}+SS).t.Pu$: Coût annuel

Minimiser ce coût revient à dériver Ct

$$\frac{\Delta(Ct)}{\Delta Q} = \frac{\Delta(\frac{N}{Q}.CL + (\frac{Q}{2}+SS).t.Pu)}{\Delta Q} = 0$$

D'où la formule de Wilson

$$Qe = \sqrt{\frac{2.N.CL}{t.Pu}}$$

Limites du modèle de Wilson

Dans la pratique on ne peut commander exactement la quantité optimale Qe, notamment du fait des unités d'achats imposées par les fournisseurs (quantités minimales, conditionnements, etc.). Il est donc plus judicieux de s'intéresser à la "zone économique", constituée par la partie inférieure de la courbe des coûts totaux.

Du fait des hypothèses simplificatrices, le modèle de Wilson ne peut fournir au mieux qu'un ordre de grandeur si consommation et/ou prix sont sujets à variations.

Le recours aux lancements de fabrication économiques est anti-flexible par essence. Ce genre de politique amène fréquemment des encours importants, risque de gonfler le stock de produits finis, reportant les coûts et pertes en aval du process.

Par ailleurs, du fait qu'elle est statique, c'est à dire que le calcul du stock se fait sur un intervalle régulier, elle ne peut pas être appliquée d'une façon continue, en temps réel.

4.5.3 Fonction basée sur la fonction d'approximation de la consommation

Dans ce qui suit, nous supposons que sur T_{task} , les quantités disponibles ou niveaux de stock d'un article $P_{k,p,av}$ utilisé pour l'ensemble des machines d'un site sur un intervalle régulier est linéaire, surtout pour les articles de la classe C.

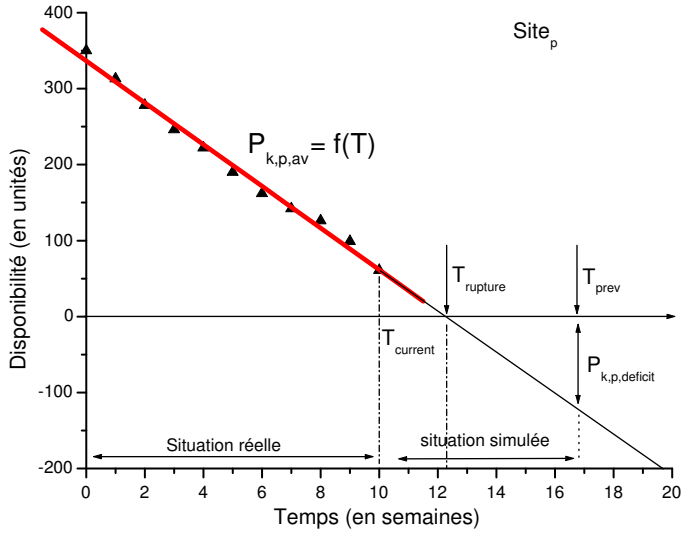

FIG. 4.10 – *Principe de Prévention: La rupture de Stock*

Cette approximation est très représentative. Et même si ce n'est pas le cas pour des types très particuliers d'articles, surtout ceux de la classe A. Notre étude restera valable ; il suffit de remplacer la fonction d'approximation, qui est la droite des moindres carrés dans notre cas, par une nouvelle fonction d'approximation qui répond au mieux à la consommation antérieure de la nouvelle classe d'articles.

On peut approximer la quantité $P_{k,p,prev}$ à T_{prev}, en utilisant des méthodes se basant sur l'approximation mathématique ou sur des méthodes classiques telles que ABC, Wilson, etc. . Cela permettra au décideur du site, au moment opportain, de se réapprovisionner.

Le principe consiste donc en l'étude et l'analyse de la consommation antérieure (Fig. 4.10.) sur des intervalles réguliers à $T_{current}$, déduire une fonction d'approximation ajustant les points de consommation sur cet intervalle, afin de prévenir un T_{prev}.

Deux cas peuvent être posés :

1. T_{prev} , cet article est en excédent car, soit sa date de péremption sera à terme, ou soit il est inutile pour les tâches à venir. Il est considéré donc comme un sur-stockage (Fig. 4.11.).
2. De la même façon que pour un sur-stockage par rapport à T_{prev}, nous pouvons prévenir aussi un sous-stockage, donc une rupture de stock avant T_{prev} calculable,c'est-à-dire à

$$T_{rupture} < T_{prev}$$

Le site accusera un sous-stockage d'une quantité $P_{k,p,prev}$ en article k à T_{prev}. Il est impératif donc aux décideurs du site de se procurer cette quantité avant T_{rupure} sinon la

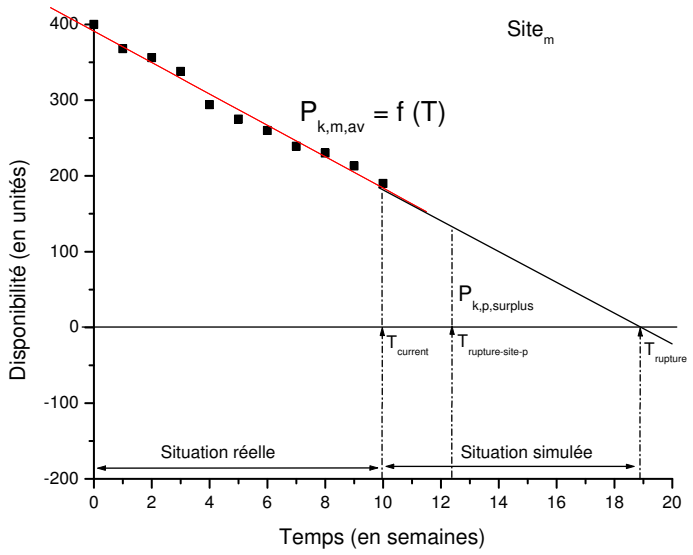

FIG. 4.11 – *Principe de Prévention: L'excédent de Stock*

tâche associée à cet article serait affectée.

4.5.4 Calcul de T_{prev} associé au niveau de stock d'un article

La loi de consommation d'un article sur un intervalle $\Delta T = T_{current} - T_0$ qui est $P = f(T)$ est linéaire. Comme le site dispose certainement d'un historique des niveaux de stock du produit P_i :

$$P_i = f(T_i), i = 1,n \tag{4.7}$$

Sur la base de cette consommation, nous allons trouver cette fonction qui est une fonction de droite, associé à un article k dans un site p, d'équation:

$$P = aT + b \quad \text{avec} \quad \text{a} \quad \text{et} \quad \text{b} \quad \text{à calculer} \tag{4.8}$$

Pour cela, nous utilisons la droite des moindres carrés. Afin de chercher les coefficients a et b. Il suffit de résoudre le système :

$$\begin{cases} \dfrac{\delta(\sum_{i=1}^{i=n}(aT_i + b - P_i)^2}{\delta a} = 0 \\ \dfrac{\delta(\sum_{i=1}^{i=n}(aT_i + b - P_i)^2}{\delta b} = 0 \end{cases} \tag{4.9}$$

qui est équivalent au système suivant:

$$\begin{cases} \sum_{i=1}^{i=n} P_i & = & a\sum_{i=1}^{i=n} T_i + nb \\ \sum_{i=1}^{i=n} P_i T_i & = & a\sum_{i=1}^{i=n} T_i^2 + b\sum_{i=1}^{i=n} T_i \end{cases} \tag{4.10}$$

C'est un système d'équations linéaires d'inconnues a et b. En considérant $z1$,$z2$,$z3$ et $z4$ qui simplifient les sommes, on obtient :

$$\begin{cases} z_1 & = & z_3 a + nb \\ z_1 & = & z_3 a + nb \end{cases} \tag{4.11}$$

Avec

$$z_1 = \sum_{i=1}^{i=n} P_i \quad , \quad z_2 = \sum_{i=1}^{i=n} P_i T_i \quad , \quad z_3 = \sum_{i=1}^{i=n} T_i \quad , \quad z_4 = \sum_{i=1}^{i=n} T_i^2$$

La solution est immédiate:

$$\begin{cases} a & = & \dfrac{z_1 z_3 - n z_2}{z_3^2 - nb} \\ \\ b & = & \dfrac{z_3 z_4 - z_4 z_1}{z_3^2 - nb} \end{cases} \tag{4.12}$$

Maintenant que nous connaissons l'équation de la consommation, nous pouvons approximer la quantité préventive $P_{k,p,prev}$ qui est :

$$P_{k,p,prev} = aT_{prev} + b \qquad \text{avec} \quad T_{prev} \quad \text{donné} \tag{4.13}$$

Par approximation, la rupture de stock arrivera donc à :

$$T_{k,p,rupture} = -\frac{b}{a} \qquad \text{quand} \quad P_{k,p,prev} = 0 \tag{4.14}$$

En se basant sur la démarche précédente de calcul, le script de l'agent prévention est maintenant réalisable.

4.6 Étude de cas

Dans ce qui suit, nous montrons, les consommations antérieures étudiées à $T_{courant}$ et respectives aux sites S_1 et S_2 en articles $P_{k,1,cons}$ et $P_{k,2,cons}$ de l'article k, de la classe C.

S_2				S_1			
T	$P_{k,2,cons}$	$Cumuls$	$P_{k,2,av}$	T	$P_{k,1,cons}$	$Cumuls$	$P_{k,2,av}$
0	0	0	400	0	0	0	350
1	32	32	368	1	37	37	313
2	12	44	356	2	35	72	278
3	18	62	338	3	32	104	246
4	44	106	294	4	24	128	222
5	19	125	275	5	32	160	190
6	15	140	260	6	28	188	162
7	21	161	239	7	20	208	142
8	9	170	230	8	16	224	126
9	17	187	213	9	27	251	99
10	23	210	190	10	38	289	61

TAB. 4.2 – *Consommation de l'article k dans les sites S_2*

Sur la base des résultats théoriques déjà présentés, en remplaçant dans le système d'équations et en résolvant le système correspondant, on obtient les deux fonctions d'approximation de la disponibilité de l'article k pour les sites site1 et site2 sur les 10 semaines et calculé à $T_{current}$= 10ème semaine.

Ce traitement fait parti du script décrivant le rôle des deux agents prévention, associé respectivement à chacun des sites S_1 et S_2, et dont les fonctions de disponibilité de l'article k sont les suivantes.

$$P_{k,1,prev} = -27{,}50\ T_{k,1,prev} + 336{,}50 \quad pour\ S_1 \tag{4.15}$$

$$P_{k,2,prev} = -20{,}73\ T_{k,2,prev} + 391{,}18 \quad pour\ S_2 \tag{4.16}$$

Les ruptures de stock pour le produit k arriveront pour les sites S_1 et S_2 à :

$$T_{k,1,rupture} = 12{,}24 \quad quand \quad P_{k,1,prev} = 0 \tag{4.17}$$

$$T_{k,2,rupture} = 18{,}87 \quad quand \quad P_{k,2,prev} = 0 \tag{4.18}$$

Si le réapprovisionnement doit se faire entre la 13ème et la 18ème semaine, cela peut affecter négativement le site S_1, car sa rupture de stock en cet article est approximativement dans deux semaines.

Le site S_2 peut donc passer une quantité d'articles à site S_1, selon des conditions négociées entre les deux sites, car ce dernier a plus de temps devant lui avant la rupture de stock du même article ($\simeq 9$ *semaines*).

Il faut signaler que l'on peut approximer la quantité nécessaire de stock devant satisfaire la consommation sur les premières semaines suivantes uniquement, mais pas à long terme, car l'approximation ne sera pas très représentative.

Maintenant, si pour un autre article r, le site S_1 possède un excédent au moment où le site S_2 est en difficulté. Ce dernier reçoit une quantité, à déterminer, de cet article.

```
 //Je suis l'Agent Prévention du site S_m
POUR chaque période T FAIRE
 POUR chaque pièce k de S_i FAIRE
  Calcul de T_Prev()
  SI P_{k,m,prev} < 0 ALORS
    Calcul de T_rupture()
    //je veux éviter la rupture de stock avant T_rupture
    Send (superviseur_m,P_{k,m,prev},demander,T_rupture)
  SINON
   SI P_{k,m,prev} > 0 ALORS
     // je peux offrir à partir de T_courant
     Send (superviseur_m,P_{k,m,prev},offrir,T_curant)
   FINSI
  FINSI
 FINPOUR
FINPOUR
```

FIG. 4.12 – *Script de l'agent prévention*

De cette façon on aura réalisé un équilibrage de ressources pour les articles k et r entre les sites S_1 et S_2.

4.7 Le Script de l'agent prévention

Comme nous l'avons expliqué auparavant, l'agent prévention associé à chaque site, surveille l'état de consommation de chaque article d'une façon continue, à des périodes fixes, étudie le rythme de ces consommations sur des intervalles, à déterminer, et sur la base de la méthode d'approximation présentée, prévoit les quantités exigées à des délais calculables qui correspondent à des T_{prev} qui sont les prochains délais de réapprovisionnement. Le script (Fig. 4.12.) montre l'interaction entre sites du système, qui est basée sur le principe de cette prévention :

Le bon fonctionnement de l'agent prévention constitue un bon élément d'aide à la décision pour la stratégie à adopter vis-à-vis des réapprovisionnements associés à chaque article.

Comme le problème posé est distribué, interactif et coopératif, dont les agents, les groupes et les structures intergroupes sont autonomes. L'objectif maintenant est d'utiliser cette démarche dans les scripts des agents et spécifier le modèle d'interaction entre agents du système afin de concrétiser la modélisation du problème, de l'implémenter et le valider sur réseau.

4.8 Conclusion

Dans ce chapitre, nous avons présenté notre démarche pour la conception et la réalisation d'un système d'équilibrage de ressources pouvant éviter à une entreprise étendue des situations anormales de sur-stockage et de sous-stockage de ses ressources. L'accent a été mis sur deux aspects:

1. L'aspect conceptuel qui a consisté à opter et structurer notre problème sous la forme d'un SMA, dont le mode d'interaction entre agents du système est purement coopératif. Ce qui permet au système d'arriver à une émergence de solution globale qui est l'équilibrage de ressources du système en tout.
2. Cette structuration du problème en SMA, nous a permis de prévoir des agents réactifs et autres cognitifs, dont l'agent de prévention qui est doté de capacité d'étudier la consommation antérieure du site et prévoir la consommation ultérieure, afin d'éviter des sur-stockage ou des sous-stockage.

Chapitre 5

Application à la e-maintenance d'une entreprise étendue

Après avoir énoncé les principes conceptuels de notre démarche pour résoudre le problème de déséquilibrage en matière de ressources dans un environnement réparti, et particulièrement coopératif, nous présentons dans ce chapitre, les éléments technologiques qui nous permettent de mettre en oeuvre cette démarche dans une e-plate-forme, à savoir comment concrétiser les concepts de nos deux modèles d'organisation et d'interaction SMA en pratique. Tout cela après avoir présenté les choix d'outils disponibles, à chaque étape.

5.1 Introduction

Avec l'avènement des Nouvelles Technologies de l'Information et de la Communication (TIC), la notion d'entreprise locale est passée vers celle d'entreprise étendue, distribuée géographiquement, mais connectée par le biais des flux d'informations qui circulent entre ses composants, via un réseau local dédié ou via Internet.

La maintenance de cette entreprise étendue n'échappe pas à la règle de cet impact, elle passe vers la e-maintenance qui n'est autre qu'un système distribué de maintenance. Mais structurellement, la e-maintenance est un système complexe, sa complexité revient à sa distribution d'un coté, mais surtout à l'hétérogénéité de ses composants et les différentes formes d'interaction entre eux de l'autre coté.

Pour la modéliser, différentes approches sont en cours de recherche un peu partout dans le monde. De notre part, nous avons apporté notre contribution par l'adaptation d'un modèle SMA à sa structuration et contribuer par conséquent à spécifier ses besoins.

Un SMA dans notre cas est un univers de sites de systèmes "intelligents" de production autonomes se déroulant en même temps, interagissant entre eux par coopération pour la réalisation d'un but commun qui est la maintenance de l'entreprise étendue, dont les sites sont connectés par une plate-forme de e-maintenance.

5.2 e-maintenance

5.2.1 e-maintenance et e-supervision

La maintenance est intimement liée à la supervision, dont les buts sont:

- La mise en oeuvre de la politique de conduite définie par la gestion prévisionnelle (aidant les opérateurs humains à prendre des décisions),
- La détection,
- Le diagnostic,
- La réactivité,
- Et la Sécurité.

Les trois tâches séquentielles détection, diagnostic et réactivité sont appelées surveillance. Ces fonctions peuvent être mises en oeuvre par :

- des logiciels
- des humains
- par les deux : opérateurs-logiciels

En généralisant ce qui précède à la e-Supervision, des politiques d'aide à la prise de décision s'imposent :

- Concevoir plusieurs Systèmes de Production (SP) locaux, chaque SP est associé à un site, complètement distribué sans interaction ni coordination entre eux.
- Centraliser le Système de supervision, dans ce cas, la prise de décision est centralisée, ce qui provoque l'effet du goulet d'étranglement.
- Répartir les prises de décision sur les différents sites, avec interaction et coopération entre eux pour la réalisation d'une tâche commune. Et, c'est sur ce point que va s'appuyer notre travail.

La Maintenance d'un site étant l'ensemble des opérations permettant de maintenir ou de rétablir un système de production durant son cycle de vie afin d'éviter sa dégradation.

Une politique de e-Maintenance globale pour l'ensemble du e-Système s'impose donc, elle doit prendre en considération les politiques spécifiques à chaque site de l'ensemble du système distribué de maintenance.

Le rôle de la e-maintenance est en forte relation avec toutes les étapes de la supervision, en particulier avec celle de la surveillance (détection, diagnostic et réactivité), puisqu'il s'agit :

- D'améliorer les performances du e-système de production par l'ajout de nouvelles techniques optimales. Pour cela, il faut s'assurer de la sûreté et de la conformité des moyens nouveaux,
- Ou du moins, de le maintenir dans son état actuel et éviter sa dégradation.

Ce e-système est généralement composé de n sites, chacun représente un site de production, ils sont dans la plupart des cas hétérogènes (sur les plans fonctionnel et structurel), tout en ayant des interactions et coopération entre eux. Et c'est pour toutes

ces raisons que ces types systèmes sont qualifiés de systèmes complexes.

La maintenance de ces sous-systèmes d'une façon centralisée est chose impossible en raison de leur hétérogénéité, pour différentes raisons :

- Ils sont différents par leurs structures d'implémentation et leurs fonctions,
- les politiques de maintenance respectives aux sites sont différentes. Cela est imposé par la nature du site de production, certains sites leur convient la maintenance systématique, d'autres la conditionnelle ou peut-être la prévisionnelle. La politique de maintenance du système distribué de production doit prendre en charge donc toutes ces différentes sous-politiques de maintenance.
- De par le flux d'informations (débit, types) qu'ils véhiculent.

5.2.2 e-maintenance et e-surveillance

Il s'agit de travaux s'attachant à l'interaction avec la commande des différents sites distribués représentant le e-Système de production et qui visent à surveiller le comportement des ressources (machine, outils, systèmes de transport et de manutention etc.) et éventuellement certaines parties de leurs commandes.

La spécification des besoins implique une analyse des défaillances et de leurs effets, c'est-à-dire en particulier de leur propagation. Leur criticité englobant [80] :

- la durée d'indisponibilité,
- la probabilité d'occurrence,
- la probabilité de non-détection,
- L'analyse des erreurs humaines est également nécessaire car elle peut être la cause première d'un mauvais fonctionnement du système.

Il faut aussi étudier les divers modes de marche et les diverses façons de reprendre un fonctionnement normal après défaillance.

Enfin, l'analyse des différentes politiques de maintenance (corrective, préventive) et qui restent très valables dans notre contexte de e-maintenance, ont un impact très important.

La conception des solutions doit répondre aux trois points essentiels de la e-Surveillance :

1. Détecter la défaillance, pour cela, il faut localiser le site défaillant et la nature de la défaillance.
2. Faire un diagnostic, qui peut être local ou dû à l'interaction avec le reste du e-Système
3. détailler les mécanismes de reprise (un site) ou de reconfiguration (sites distribués)

Ceci nécessite un bon modèle de fonctionnement et, également des modèles de pannes dans la majorité des cas. Cette étape est cruciale dans le cycle de vie de la démarche, c'est elle, et sur la base de sa conception que la suite des étapes de ce cycle se dessinera, elle concernera :

- l'architecture du e-système,

- l'interaction entre les composants,
- le type de coopération
- le rôle de chaque entité par rapport au but fixé localement,
- le rôle par rapport à son environnement,
- les limites de l'intelligence de l'entité,
- etc.

La détection est pratiquement toujours fondée sur une comparaison, en temps réel, du comportement effectif du système surveillé avec celui d'un modèle de bon comportement.

En productique, on se place en général à un niveau plus agrégé et l'on surveille l'apparition d'événements (défaillance, échéancier, évolution des paramètres, etc.).
La détection consiste à remarquer une incohérence entre le comportement du système physique et celui du modèle.
Suivant l'instant auquel est effectuée la comparaison, diverses approches peuvent être proposées.

- Si la comparaison est faite au moment où une commande va être envoyée, la notion de filtre de commande est utilisée.
- Si elle est faite au moment de la réception d'une nouvelle information, c'est la notion du filtrage-perception (mise à jour d'une base de connaissance en IA)
- Si elle est faite dans les deux cas c'est la notion de modèle de référence.

Le diagnostic, lui, est souvent fondé sur la recherche d'une cohérence entre le comportement (ou l'état) du système physique et celui d'un modèle de panne correspondant à des connaissances profondes.
Une autre approche du diagnostic consiste à formaliser les connaissances superficielles des opérateurs sous la forme d'un ensemble de règles.
Il est souvent possible de travailler en utilisant des techniques d'apprentissage, soit sous la forme de réseaux de neurones, soit en synthétisant un automate reconnaissant les scénarios significatifs (chroniques).
Les modèles de pannes peuvent correspondre à diverses visions [80]:

1. Modèles structurels, décrivant l'architecture des systèmes (vision statique)
2. Modèles comportementaux, décrivant des enchaînements d'événements et d'activités (sous la forme d'automates par exemple)
3. Modèles fonctionnels, décrivant les transformations de valeurs ou d'informations effectuées par les éléments du système (fonctions algébro-différentielles par exemple)
4. Modèles causaux, décrivant les événements observés qui sont des conséquences directes ou indirectes des pannes (les arcs d'un graphe causal peuvent être interprétés comme des implications logiques).

Une étude doit déterminer le modèle de panne à adopter mais, à priori, le modèle b) paraît le plus en adéquation avec notre approche de modélisation.

Pour les mécanismes de reprises; Côté e-maintenance, à première vue, ce volet est étroitement lié, au moins, à trois grands axes :

1. Les stratégies de reconfiguration du e-système
2. Les Politiques de tolérances aux pannes,
3. La politique de maintenance associé à chaque site composant le système.

5.3 La e-maintenance et le génie logiciel

Les systèmes multi-agents sont des systèmes logiciels complexes, nécessitant des méthodes d'ingénierie adéquates. Traditionnellement, on distingue en génie logiciel trois phases de construction de systèmes : l'analyse, la conception et le développement. Nous pensons que pour des systèmes tels que les systèmes multi-agents, ce n'est pas suffisant, et que pour tenir compte de l'intégralité du processus de construction des systèmes multi-agents, du début de l'analyse à l'exécution finale de l'application, nous devons distinguer une quatrième étape après l'étape de développement : le déploiement. Par exemple, les systèmes multi-agents sont généralement distribués dans un réseau d'ordinateurs, et la population d'agents d'une application est susceptible d'évoluer au cours de l'exécution. Nous sommes également conscients du fait que les frontières exactes entre les trois étapes classiques (analyse, conception, développement) font toujours l'objet de débats dans le domaine du génie logiciel. Nous pensons que ces étapes ne sont que des niveaux quantifiés au long d'un processus conceptuel continu. Nous proposons d'utiliser par la suite les définitions suivantes des quatre étapes :

1. Analyse : la détermination, séparation et description du type de problème et du domaine environnant. En pratique, il s'agit d'identifier le domaine d'application et le coeur du problème à résoudre.
2. Conception : la définition de l'architecture de la solution au problème, de manière déclarative. En pratique, il s'agit de spécifier un principe de solution, par exemple en utilisant UML ou Agent UML (AUML) qui est un langage en cours de spécification dont l'objectif est de développer une sémantique, un méta-modèle et une syntaxe abstraite commune et indépendante pour les méthodologies agent. Actuellement, le groupe de travail AUML n'a pas publié de spécifications, mais plusieurs publications sur des extensions d'UML ont étés publiés par les membres du groupe. Actuellement, leurs travaux se concentrent sur les interactions, plus spécifiquement sur les protocoles d'interaction, mais aussi sur d'autres aspects connexes comme les langages d'interaction, la coordination, les rôles des agents. D'autres travaux portent également sur les architectures, et les ontologies. Ces travaux sont prometteurs, mais ne se sont pas encore suffisamment mûrs pour pouvoir être utilisés dans de vrais projets. Actuellement, aucune spécification standardisée de AUML n'est publiée. Tout comme pour la FIPA, les premiers éléments des systèmes multi-agents qui sont étudiés sont les interactions, car celles-ci se prête relativement bien à la formalisation. Cependant, le danger associé à cette approche est que si l'on

sait facilement spécifier des interactions complexes utilisant des protocoles d'interaction complexes, on ne sait toujours pas spécifier, voire même construire, des agents sachant les employer.

3. Développement : la construction d'une solution fonctionnelle au problème. En pratique, il s'agit de coder la solution dans un langage de programmation, qui peut être général (par exemple Java) ou plus orienté multi-agents (par exemple AOP).
4. Déploiement : l'application de la solution au problème réel dans le domaine. En pratique, il s'agit de lancer le logiciel sur un réseau d'ordinateurs, et de maintenir (modifier ou remplacer des agents) ou d'étendre ses fonctionnalités (migration d'agents).

5.3.1 Le génie logiciel

Au début de l'ère informatique, le processus d'activité d'écriture du logiciel ne reposait que sur l'efficacité personnelle du programmeur laissé pratiquement seul devant la programmation d'un problème. De nos jours, la production d'un logiciel est le résultat de la coopération de plusieurs acteurs dont le programmeur qui dispose d'outils et de méthodes lui permettant de concevoir et d'écrire des logiciels.

Le terme logiciel, ne désigne pas seulement les programmes associés à telle application ou tel produit, il désigne en plus la documentation nécessaire:

- au développement,
- à l'installation,
- à l'utilisation,
- et à la maintenance de ce logiciel.

Pour de gros systèmes, le temps de réalisation peut être aussi long que le temps du développement des programmes eux-mêmes.

Le génie logiciel concerne l'ensemble des méthodes et règles relatives à la production rationnelle des logiciels [77]. L'activité de développement du logiciel, vu les coûts qu'elle implique, est devenue une activité économique et doit donc être planifiée et soumise à des normes sinon à des attitudes équivalentes à celles que l'on a dans l'industrie pour n'importe quel produit. Le mot-clef est désigne le mot "composant logiciel" qui tient à la fois de l'activité créatrice de l'humain et du composant industriel incluant une activité disciplinée et ordonnée basée pour certaines tâches sur des outils formalisés[77].

D'autre part le génie logiciel intervient lorsque le logiciel est trop grand pour que son développement puisse être confié à un seul individu ; ce qui n'est pas le cas pour des débutants, à qui il n'est pas confié l'élaboration de gros logiciels. Toutefois, il est possible de sensibiliser le développeur débutant à l'habitude d'élaborer un logiciel d'une manière systématique et rationnelle à l'aide d'outils simples.

5.3.2 Facteurs de qualité du logiciel

Un utilisateur quelconque, lorsqu'il achète un produit comme un appareil électroménager ou une voiture, attend de son acquisition qu'elle possède un certain nombre

de qualités (fiabilité, durabilité, efficacité,...). D'après B.Meyer et G.Booch, il en est de même avec un logiciel qui doit posséder les critères suivants:

- La correction est la qualité qu'un logiciel a de respecter les spécifications qui ont été posées.
- La robustesse est la qualité qu'un logiciel a de fonctionner en se protégeant des conditions de dysfonctionnement.
- L'extensibilité est la qualité qu'un logiciel a d'accepter des modifications dans les spécifications et des adjonctions nouvelles.
- La réutilisabilité est la qualité qu'un logiciel a de pouvoir être intégré totalement ou partiellement sans réécriture dans un nouveau code.
- La compatibilité est la qualité qu'un logiciel a de pouvoir être utilisé avec d'autres logiciels sans autre effort de conversion des données par exemple.
- L'efficacité est la qualité qu'un logiciel a de bien utiliser les ressources.
- La portabilité est la qualité qu'un logiciel a d'être facilement transféré sur de nombreux matériels, et insérable dans des environnements logiciels différents.
- La vérificabilité est la qualité qu'un logiciel a de se plier à la détection des fautes, au traçage pendant les phases de validation et de test.
- L'intégrité est la qualité qu'un logiciel a de protéger son code et ses données contre des accès non prévus.
- La facilité d'utilisation est la qualité qu'un logiciel a de pouvoir être appris, utilisé, interfacé, de voir ses résultats rapidement compris, de pouvoir récupérer des erreurs courantes.
- La lisibilité est la qualité qu'un logiciel a d'être lu par un être humain.
- La modularité est la qualité qu'un logiciel a d'être décomposable en éléments indépendants les uns des autres et répondants à un certain nombre de critères et de principes.
- L'abstraction est la qualité qu'un logiciel a de s'attacher à décrire les opérations sur les données et à ne manipuler ces données qu'à travers ces opérations.

Nous considérons que, pour le cas d'une plate-forme sur réseau local ou dans le cas d'une e-plate-forme sur le Web, ou pour un progiciel dédié à une application distribuée, d'autres critères sont à prendre en considération concernant surtout le fonctionnement du logiciel, une fois il en oeuvre:

1. L'intégration : L'intégration de l'existant avec les nouvelles technologies pour préserver les applications " patrimoines " qui sont souvent indispensables au fonctionnement des systèmes (voir Fig. 5.1), et il serait très coûteux de les remplacer. Cependant, l'intégration de technologies hétérogènes s'avère généralement complexe et nécessite des plates-formes d'exécution réparties et souples .
2. L'interopérabilité : On peut trouver des solutions aux problèmes précédents mais ces solutions restent propriétaires et ne peuvent pas s'appliquer dans tous les

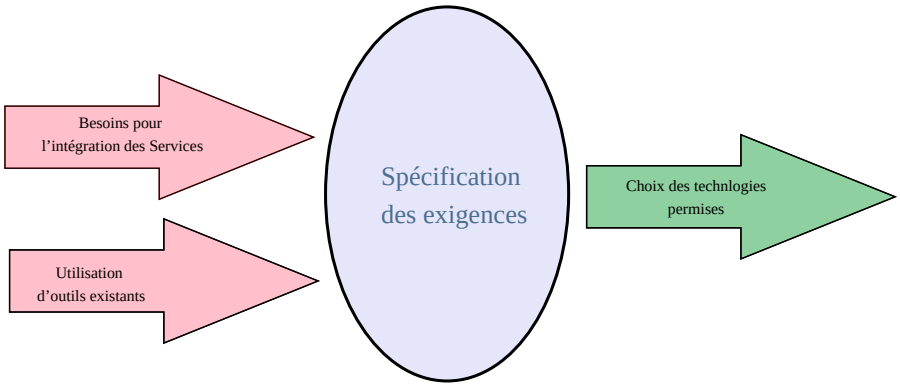

FIG. 5.1 – *Integration d'outils existants.*

contextes comme pour la coopération entre des applications développées indépendamment. D'où la nécessité de définir des normes acceptées et suivies par tous les acteurs impliqués dans la coopération pour atteindre l'interopérabilité .

3. L'hétérogénéité : L'hétérogénéité des environnements distribués est dû à la grande diversité des technologies proposées par l'industrie de l'informatique Par exemple le réseau d'une grand compagnie bancaire peut être composé de divers types d'ordinateurs des super calculateur des ordinateurs départementaux ou encore des stations personnelles fonctionnant sur différents systèmes d'exploitation comme Unix, Linux, Windows, Mac-OS. Les réseau et leurs protocoles associés qui permettent d'interconnecter ces machines et de faire communiquer les applications distribuées sont de nature divers les réseaux.
4. La communication : Un service distribué est composé des différents éléments logiciels et matériels mis en ouvre dans sa réalisation :les interfaces d'interactions pour les utilisateurs, les logiciels de services (serveurs), les machines, les espaces de stockage, les réseaux les protocoles de communication/dialogue entre les machines mais aussi entre les logiciels. l'ensembles de ces logiciels dialoguent selon un protocole client/serveur : les clients sont des applications destinées aux utilisateurs finaux, les serveurs sont les applications gérantes informations et les ressources partagées. Comme ces applications sont distribuées sur différents sites, il est nécessaires de les faire communiquer afin qu'elles coopèrent pour la réalisation d'un travail commun.

5.3.3 Cycle de développement proposé

La figure 5.2 représente le cycle de vie classique d'un logiciel dédié à la supervision qui, pour le rendre opérationnel, il faut passer par le chemin descendant pour sa formulation jusqu'à son implémentation et pour le valider, le tester et le rendre exploitable; il faut passer par les étapes ascendantes.

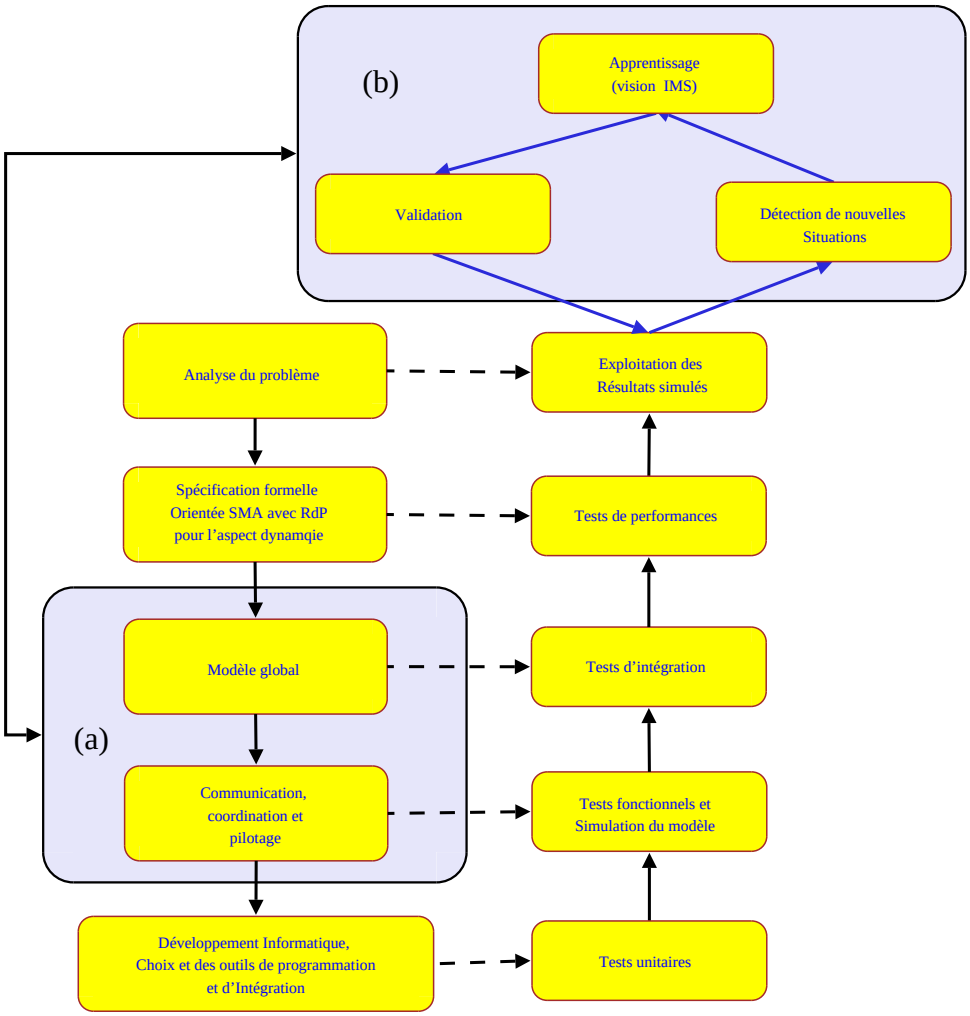

FIG. 5.2 – *Notre vision de cycle de développement orienté SMA [8]*

1. La spécification des besoins de la e-maintenance est caractérisée par :
   - L'aspect hétérogénéité des sites de production,
   - la nature de la distribution (complètement distribué, hiérarchique distribué),
   - le type d'interaction engendrant la coopération ou non.

   Cette spécification se concrétise généralement en deux grandes étapes :
   - informelle, obéissant à un cahier des charges,
   - formelle,équivalente sémantiquement à la première, mais qui tient compte des exigences des règles d'écriture du langage de spécification, tout en faisant abstraction à l'implémentation pour le moment.

Cette étape, même si elle se situe avant la phase conception, elle doit prendre en considération la structure du modèle dans lequel sera conçu ce e-système.

2. La modélisation générale de la e-maintenance, l'axe central du cycle de vie, qui structure au mieux l'architecture du e-système avec bien évidemment étude des différentes interactions et communication entre les éléments du e-système, pour cela, il faut :
   - Etude des différentes modélisations associés aux sites, et voir ce que peut apporter chacune des modélisations comme contribution dans le la modélisation du système global. Faire recours, par exemple, à la modélisation par réseau de Petri (RdP) pour les cas dynamiques.
   - Opter pour un type de modélisation en faisant toute l'étude portant sur :
     - L'architecture du e-système (hiérarchique, niveaux de hiérarchie, etc.).
     - Etude des composants du e-système.
     - Comment représenter chaque composant ?
     - L'interaction entre ces composants.
     - La coopération entre les composants.
     - Le comportement des composants et du e-système pour l'accomplissement des buts visés.
     - etc.
3. l'analyse et l'évaluation des performances : La vérification (ou peut être certification) de tout logiciel utilisé pour la commande est une tâche essentielle. Sa conception et son implémentation est certes d'ordre informatique, mais la façon dont il faut spécifier les fonctions réalisées par ce logiciel dépend fortement de l'application et cela entraînent des spécificités. Dans le cycle de vie classique en " V ", chaque étape (analyse, conception architecturale, etc.) comprend également la définition des procédures de test (e-système, sous-systèmes, unités, etc.) et de vérification. Deux approches majeures sont généralement utilisées dans cette étape de la supervision des e-systèmes de Production pour analyser et évaluer le logiciel produit :
   (a) l'une relevant de la simulation, plus proche du système réel permettant d'explorer les régimes transitoires et permanents mais demandant certaines précautions pour l'interprétation et la validation des résultats,
   (b) la seconde, analytique, nécessitant des simplifications physiques plus importantes, donnant des résultats exacts ou plus approchés au détriment d'une plus grande complexité et se limitant aux régimes permanents.
   Comme notre tâche dans le cycle cité se limite à la phase spécification formelle et modélisation générale, nous serons contraints à valider notre modèle par la première méthode, c'est-à-dire une simulation de scénarios pour des études de cas.
4. la maintenance au cours du temps. Cette tâche sera aisée si la conception du système est modulaire et ouvert et répond aux critères efficaces du génie logiciel. La réalisation du logiciel peut se faire :
   (a) Soit en différé (off-line) telle que la conception formelle du logiciel de commande, maintenances préventives, etc.
   (b) Soit en direct (on-line) tel que le cas de la réactivité.

L'adaptation de l'approche SMA à la e-maintenance n'est pas chose évidente, puisqu'il s'agit :

(a) de comprendre et cerner la politique et les techniques de maintenance associée à chaque site pour pouvoir la modéliser.

(b) partir d'une étude de cas qui va comprendre :

- étude du cahier des charges d'un cas posé,
- spécifier informellement les besoins associés au cas posé,
- spécifier formellement ces besoins en :
  - Se fixant comme architecture générale une structuration e-système de production, c'est-à-dire des agents distribués nommés, avec lesquels on peut communiquer (accointances).
  - Concevoir un métalangage de communication entre agents en fonction de cette architecture :
  - Syntaxiquement, basé communication (envoi et réception de messages entre agents basés sur la négociation et la coopération),
  - Sémantiquement basée modèle d'exécution parallèle.
- Simuler ce métalangage sur le cas posé, en faisant appel à des logiciels qu'on intègre dans le processus de réalisation (KADS, Ilog Solver, etc.).

En productique, on suppose que le système d'exploitation et les réseaux locaux sont corrects, ils ne sont pas au centre des préoccupations.

Le producticien développe des modèles qu'il implémente pour surveiller le procédé et extraire les informations qu'il trouve utiles. Il se focalise essentiellement sur la partie du système informatique correspondant à l'application, c'est-à-dire mettant en oeuvre les fonctions à remplir par le système de commande et de surveillance.

Le choix de la plate-forme sur lequel l'application doit être exécutée, ou du moins simulée et validée, est fondamental. Le fait que l'architecture du e-système est distribuée, deux solutions se présentent devant nous dans notre cas:

1. Implémentation du système sur une architecture parallèle.
2. Simuler l'exécution sur :
   - Un système d'exploitation multitâche à part entière (genre Linux), où chaque agent associé à sous système sera associé à un processus (au sens informatique du terme) durant son activation (passage de l'objet vers l'agent).
   - Windows-xx par multithreading où l'activation de chaque agent est simulé par un thread,
   - Une plate-forme sur un réseau local ou dédié, où chaque agent est associé à un poste.

5.4 Modélisation de la e-maintenance

5.5 La plate-forme PROTEUS

Le projet européen Proteus est animé par plusieurs partenaires industriels et universitaires allemands, belges, français et luxembourgeois. Ce projet propose de créer une plate-forme distribuée générique pour la e-maintenance. Cette plate-forme doit pouvoir s'interfacer avec n'importe quel outil de maintenance existant. Ce sont des outils tels que des logiciels de Gestion de la production (ERP), de gestion de maintenance asistée par ordinateur (GMAO), mais aussi des bases de données documentaires. Ces bases de données peuvent contenir toutes forme d'information ayant trait avec le contexte de la e-maintenance, en particulier la documentation liée à la pièce de rechange. Cette plate-forme doit également pouvoir se connecter à faire transiter des informations temps réel venant de différents capteurs situés dans ateliers de production. Ces capteurs peuvent être des capteurs de température, ou de toute autre forme de donnée utile. Ce peut être des caméras de surveillance. D'autre part, il doit aussi être possible pour un agent de maintenance de faire la visio-conférence avec d'autres agents de maintenance. Les agents de maintenance doivent pouvoir se connecter à la plate-forme Proteus au moyen de terminaux de différents types. Ces terminaux peuvent être des pockets-PC, des téléphones mobiles ou des PC traditionnels. Ils doivent pouvoir se connecter sur proteus de n'importe où, en utilisant Internet pour support.

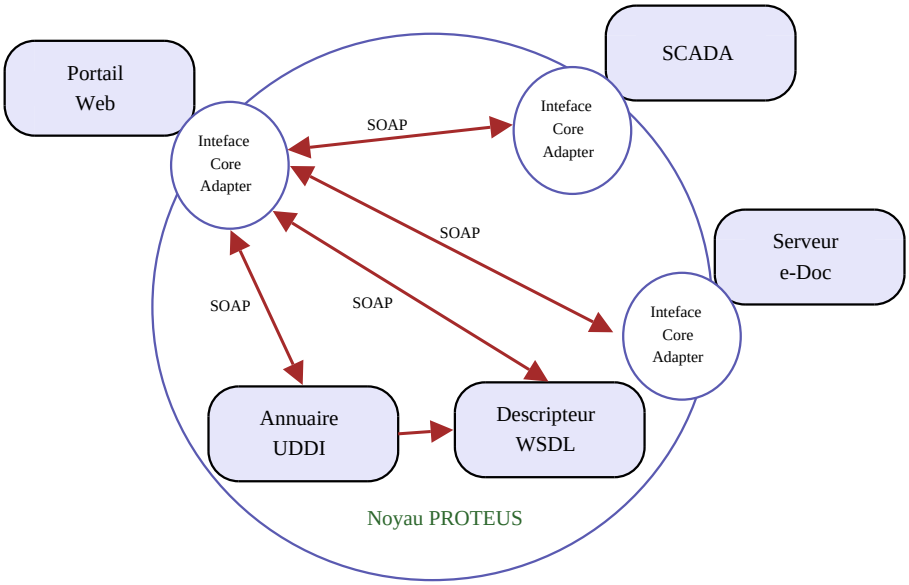

FIG. 5.3 – *Architecture technique de la plate-forme Proteus*

Par exemple, l'un des scénarios de la plate-forme Proteus se propose, pour un agent de maintenance équipé d'un pocket-PC devant son équipement défectueux, de pouvoir au moyen d'un navigateur demander des documents techniques relatifs à ce type de

matériel. Ces documents pouvant situer sur différentes bases de données documentaires existant bien avant que Proteus n'ai été mis en place, et situés sur des serveurs et sites distants accessibles par Internet.
Ce technicien doit également pouvoir avoir accès à l'évolution des températures et aux historiques de remontées d'alarmes sur cet équipement particulier. il doit pouvoir aussi, et en utilisant une même interface, visionner l'enregistrement de la caméra de surveillance durant les deux derniers quarts d'heure ayant précédés la panne. tous ces documents doivent pouvoir aussi être visionnés en même temps que les autres exports de maintenance sur ce type de matériel, avec lesquels il est en visio-conférence. Ces experts se trouvant respectivement à Besançon et Sétif, par exemple.

5.5.1 Outils de Communication dans les applications distribuées

Une plate-forme de e-maintenance a pour but de mettre à la disposition des utilisateurs des services sous la forme de web-services. cette plate-forme doit être distribuée, et pouvoir s'interfacer avec d'autres applications déja existantes sur d'autres sites. C'est pour cette raison qu'il faut étudier les middlewares existants déja (XML-RPC, CORBA, DCOM et Java-RMI), et déduire pourquoi l'utilisation de SOAP et non pas ces middlewares.

5.5.2 Les middlewares

Un middleware est une couche de logiciel située entre le système d'exploitation et les applications , permettant l'échange d'informations entre celles-ci . il constitue l'infrastructure d'un système informatique sur laquelle les logiciels viennent se connecter. Il apparaît de ce fait comme le logiciel central du système client /serveur, on peut généralement distinguer deux grandes classes:

- Le middleware général :
  Formant le substrat de la majorité des interaction client/serveur, il comprend les annuaires répartis, la synchronisation de l'heure sur le réseau, les appels de procédures distantsĚ Parmi les produits qui se trouvent dans cette catégorie, on trouve : TCP/IP et DCE.
- Le middleware propre à un service :
  Est nécessaire pour accomplir certaines types particulières des services client/serveur, on y trouve :
  - Le middleware spécifique de l'Internet , tel que http.
  - Le middleware propre au transactionnel avec transactionnel RPC, ODBC(open database connectivitty) de MicroSoft.
  - Le middleware propre aux base de données avec EDA/SQL.
  - Le middleware propre aux objets tels que CORBA (OMG).

Il est à remarquer que quel que soit le middelware utilisé, les applications distribuées ont besoin , pour leur exécution, de plates-formes reparties ses plates-formes nécessitent la définition des normes prenant en compte la communication, l'hétérogénéité, l'intégration et l'interopérabilité des applications distribuées.

5.5.3 Les Middelwares et le Web

Au fil du temps, il est devenu courant de construire des applications sous forme d'ensembles de composants distribués sur un réseau de machines, qui fonctionnent ensemble comme autant d'éléments d'un programme global. Lorsqu'il s'agissait d'applications distribuées, on avait l'habitude d'utiliser des technologies orientées objet ou composant comme DCOM (Distributed Component Object Model, Microsoft), CORBA (Common Object Request Broker Architecture, Object Management Group) ou RMI (Remote Method Invocation, Sun).

Ces technologies offraient une architecture fiable et évolutive qui permettait de répondre aux besoins croissants des applications. Si ces technologies à composants fonctionnent très bien dans un environnement intranet, leur utilisation sur Internet présente deux problèmes majeurs.

Tout d'abord, elles ne sont pas compatibles. Bien qu'elles fonctionnent toutes avec des objets, ces technologies ne s'accordent pas sur les détails, comme par exemple la gestion du cycle de vie, la prise en charge de constructeurs et le degré de prise en charge de l'héritage.

Ensuite, et surtout, elles utilisent exclusivement des communications de type RPC, ce qui conduit à des systèmes étroitement imbriqués construits autour d'appels explicites des méthodes objet.

Les applications Web fonctionnant sur navigateur sont, tout au contraire, flexibles et remarquablement interopérables. Elles communiquent via HTTP pour échanger des données MIME dans des formats divers et variés.

Les Web Services adaptent le modèle traditionnel de programmation Web pour qu'il soit utilisable dans tous types d'applications et non exclusivement dans celles fonctionnant sur navigateur. Ils échangent des messages SOAP à l'aide de HTTP et autres protocoles Internet.

Les Web Services sont fondés sur des normes comme HTTP, XML, SOAP et WSDL ; pour exposer les fonctionnalités d'une application sur Internet, ils sont donc indépendants du langage de programmation, de la plate-forme et du dispositif utilisé. L'infrastructure des Web Services ASP.NET, par exemple, offre une API simple pour les Web Services, qui fonctionne par mappage de messages SOAP sur des appels de méthodes.

Pour ce faire, elle propose un modèle de programmation très simple, qui consiste à mapper les échanges de messages SOAP sur des appels de méthodes spécifiques. Les clients des Web Services ASP.NET n'ont besoin de connaître ni la plate-forme, ni le modèle objet, ni le langage de programmation utilisé pour les construire. Les services eux-mêmes n'ont pas besoin d'avoir d'informations sur les clients qui leur envoient des messages. La seule exigence est que les deux parties s'accordent sur le format des messages SOAP produits et consommés, défini dans le contrat du service Web exprimé à l'aide de WSDL et du schéma XML (XSD).

Nous nous intéresserons à deux types de middlwares, le bus Corba et le SOAP, et nous en déduisons le pourquoi de l'utilisation de l'un au détriment de l'autre dans les plate-formes de e-maintenance.

5.5.4 Le bus Corba

Le bus CORBA propose un modèle client/serveur orienté objet d'abstraction et de coopération pour les applications réparties, chaque application peut exporter certaine de ces fonctionnalités (services) sous la forme d'objets CORBA .la notion client/serveur intervient uniquement lors de l'utilisation d'un objet : l'application implantant l'objet est le serveur, l'application utilisant l'objet est le client, bien entendu une application peut tout à fait être à la fois client et serveur.

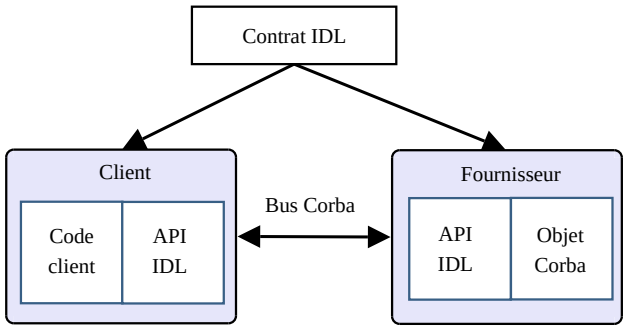

FIG. 5.4 – *Le middleware Bus Corba*

IDL (interface définition language) c'est le langage neutre par apport aux implémentation des clients et serveurs , il définit l'interface d'un composant et la description des service qu'il souhait proposé à ses clients et d'autre coté il définit aussi les interfaces de ces derniers. C'est ce qu'on appelle le contrat IDL, la coopération entre les fournisseurs et les utilisateurs de services, en séparant l'interface de l'implantation des objets et en masquant les divers problèmes liés à l'interopérabilité, l'hétérogénéité et lalocalisation de ceux-ci. Un contrat IDL spécifie les types manipulés par un ensemble d'applications réparties, c'est-à-dire les types d'objets (ou interfaces IDL) et les types de données échangés entre les objets. Le contrat IDL isole ainsi les clients et fournisseurs de l'infrastructure logicielle et matérielle les mettant en relation à travers le bus CORBA.

Les contrats IDL sont projetés en souches IDL (ou interface d'invocations statiques SII) dans l'environnement de programmation du client et en squelettes IDL (ou interface de squelettes statiques SSI) dans l'environnement de programmation du fournisseur. Le client invoque localement les souches pour accéder aux objets. Les souches IDL construisent des requêtes, qui vont être transportées par le bus, puis délivrées par celui-ci aux squelettes IDL qui les délégueront aux objets. Ainsi le langage OMG-IDL est la clé de voûte du bus d'objets répartis CORBA.

5.5.5 DCom

Au même titre que les autres middlewares dédiés aux applications distribuées, DCOM (Distributed Common Object Model) est présenté comme l'un des middlewares stan-

dards. DCOM est en fait un ensemble d'extensions basées sur Distributed Computing Environment RPC (DCE-RPC), et une évolution de COM et de COM+ de MicroSoft. Les composants de DCOM peuvent fonctionner dans des processus différents et donc un outil tout à fait adapté aux applications distribuées. La réutilisation des composants est favorisée au sein de DCOM.

Il est possible de faire une apparentée entre les composants de DCOM et les composants CORBA. En effet, pour DCOM, le concept de "client" est perçu comme étant une application invoquant les méthodes d'un composant DCOM fonctionnant sur un serveur . Le client est la partie du système qui rassemble les autres, il rend le système opérationnel.

Ainsi, le concept de "serveur" pour DCOM est une application qui rend accessibles des composants DCOM à des clients. Le serveur DCOM est en fait un exécutable ou une DLL (Librairie Windows Dynamique). Cet exécutable, ou librairie dynamique, contient les éléments DCOM rendus disponibles au client. Coté serveur, il existe aussi la référence counting". ce module complémentaire de la technologie DCOM permet de décider quand un serveur n'est plus utilisé, et ainsi de l'enlever de la mémoire.

Une "inteface" est un pointeur vers un groupe de fonctions. Ce pointeur peut être appelé via DCOM. L'interface est l'unique partie du système visible du client. C'est la spécification des méthodes qu'un serveur fournit pour une composante DCOM. Intimement lié à ce concept d'interface , la "classfactory" DCOM est un service qui retourne un pointeur à l'interface. Ce pointeur est retourné uniquement sur demande du système DCOM. Cette classfactory n'est pas visible par le client.

Une "class" est la définition d'une abstraction qui implémente une plusieurs interfaces.

DCOM se sert du "Marshaling". C'est une opération qui consiste à transformer et à transférer des données entre le client et le serveur. Par conséquent, un "object" est une instance d'une classe.

Windows fournit un Active Program Interface (API). Cet API permet aux clients d'initialiser un service et de demander un pointeur à l'interface d'un serveur. Comme nous, c'est le client qui rend le système opérationnel. Il initialise le serveur DCOM, fait les appels à l'API DCOM pour obtenir le pointeur sur l'interface. C'est aussi le client qui lance les méthodes distantes. C'est aussi lui qui ferme l'accès au système, lorsqu'il en a terminé avec les services.

Tout comme SOAP, DCOM peut fonctionner sur HTTP. Néanmoins, il présente quelques désavantages. En effet, DCOM n'est plus réellement un standard ouvet, car il est essentiellement développé pour les plate-formes WIndows. Il est néanmoins présenté comme étant multi-plate-formes. Toutefois, dans la pratique, il est difficile de l'adapter aux autres plate-formes que Windows. cela en fait donc un outil peu adapté aux environnements hétérogènes. Et il n'est pas possible d'utiliser DCOM depuis un client pour appeler autre chose qu'un programme serveur DCOM. Il n'est pas possible d'appeler un script CGI, par exemple, Alors que SOAP offre cette possibilité.

Bien qu'étant présenté comme l'un des middlewares incontournables dans certains cas, il est en train de perdre du terrain au profit de SOAP, même sur les plate-formes Windows.

5.5.6 Java-RMI

Java-RMI est une plate-forme à objets répartis, basée sur le langage Java. Le RMI (Remote Method Invocation) hérite donc des caractéristiques de ce langage, tel que le faitr qu'il soit orienté objet. java étant portable sur n'importe que système java-RMI est donc un outil parfaitement adapté aux développements pour plate-formes hétérogènes. RMI est proche du modèle de base de Corba, mais moins complet.

Les objets communiquent entre eux en utilisant des fonctions de communications. Ces fonctions sont en fait des programmes Java. les liens sont dynamiques et sont réalisés par un serveur de nommage. Le nommage reposant quant à lui sur des URL. tout comme dans Corba, les concepts de stubs et de skeleton existent aussi et ont la même signification. Les communications se font par sockets, ou invocation distantes. java-RMI possède son propre protocole de communication, le JRMP. Il est néanmoins possible de communiquer en utilisant le protocole HTTP.

Java-RMI hérite donc des qualités de Java, le plus important étant le portage. Il peut, tout comme SOAP et DCOM, fonctionner en utilisant le protocole HTTP. Néanmoins, si Java-RMI prend en compte l'hétérogénéité des plate-formes, il ne prend pas en compte celle des langages. Seuls les programmes Java peuvent être appelés par Java-RMI.

5.5.7 XML-RPC

RPC

Le principe du RPC (Remote Procedure Call) fonctionne de la manière suivante:

Le programme client contient une fonction locale portant le même nom que la fonction distante (le service). Cette fonction distante est implémentée de la même manière qu'une fonction classique sur le serveur distant. Les connexions réseau sont prises en charge par les stubs. Les stubs sont en fait des fonctions ayant en charge ces connexions. Il y a donc un stub client et un stub serveur en plus de la fonction distante et du programme client. L'interfaçage entre les deux se fait par un IDL (interface Description Langage). Un seul argument est autorisé en paramètre de la fonction et une seule valeur est autorisé en retour. Il est donc nécessaire d'utiliser des structures pour faire passer ou retourner plusieurs arguments.

C'est le format eXternal Data Representation (XDR) qui permet de définir les types utilisés pour l'échange de variables entre le client et le serveur, car les langages peuvent être différents entre le client et le serveur.

XML-RPC

Basé sur RPC et XML, XML-RPC est un langage de requête. C'est un standard libre permettant à différents systèmes de faire des appels de procédures distantes sur Internet. Ce langage repose sur le protocole HTTP pour le transport et la structure d'arbre XML pour l'encodage. XML-RPC est particulièrement adapté aux transmissions simples de structures complexes.

C'est XML-RPC qui a servi de base pour la création du middleware SOAP au W3C.

5.5.8 SOAP

Ce protocole récent qui a fait son apparition suite aux inconvénients des middelwares cités, nous disons dans ce paragraphe que c'est un middelware dont le code est écrit en XML et exécuté généralement au dessus de HTTP, donc indépendant des pare-feux. Nous expliquerons, plus en détail, dans un paragraphe ultérieur.

5.6 Les web-services

Les Web Services sont actuellement les principaux blocs de construction de l'informatique distribuée sur Internet. Les normes ouvertes et l'accent mis sur la communication et la collaboration entre les gens et les applications ont créé un environnement dans lequel les Web Services deviennent la plate-forme pour l'intégration d'applications. Les applications sont construites à l'aide de plusieurs Web Services provenant de sources différentes et fonctionnant ensemble, quel que soit leur emplacement et la manière dont elles ont été mises en oeuvre.

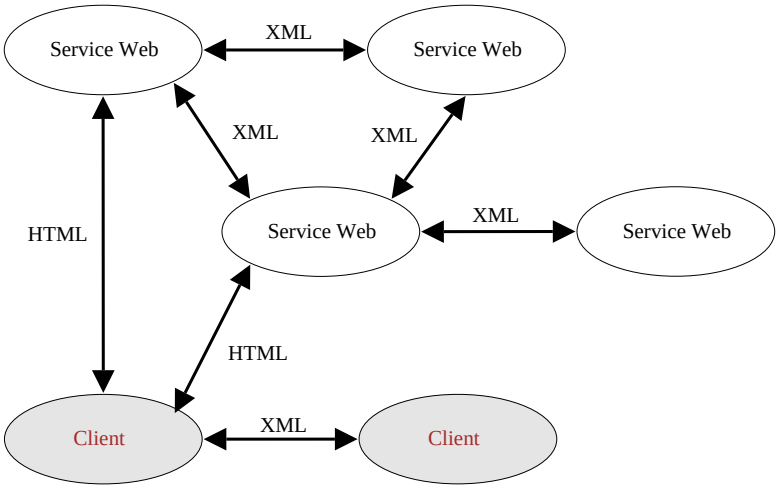

FIG. 5.5 – *Interaction Client/WebService*

Il existe probablement autant de définitions des Web Services que d'entreprises qui les créent, mais presque toutes ces définitions ont ceci en commun :

- Un service Web est une "unité logique applicative" accessible en utilisant les protocoles standard d'Internet
- Une "librairie" fournissant des données et des services à d'autres applications.
- Un objet métier qui peut être déployé et combiné sur Internet avec une faible dépendance vis-à-vis des technologies et des protocoles.
- Combine les meilleurs aspects du développement à base de composants et du Web.

Les services Web permettent d'interconnecter:

- différentes entreprises,
- différents matériels,
- différentes applications,
- différents clients,
- et distribuer et intégrer des logiques métiers.

Mais surtout, les Web Services, vu qu'ils sont basés sur XML, permettent d'aller vers un Web basé sur un aspect sémantique en plus l'aspect purement interactif.

La difference entre les Web service et les anciennes technologies est que le SOAP, qui est basé sur le XML, est sensiblement moins complexe que les approches précédentes, c'est pourquoi il est bien moins difficile d'obtenir une implémentation SOAP conforme aux normes.

Les Web Services ont un autre avantage significatif par rapport à d'autres solutions : ils fonctionnent avec des protocoles Web standard (XML, HTTP et TCP/IP).

Bon nombre d'entreprises ont déjà une infrastructure Web, ainsi le personnel possédant les connaissances et l'expérience de sa maintenance, c'est pourquoi le coût d'accès aux Web Services est très inférieur à celui des technologies précédentes.

5.6.1 Architecture des Web Services

L'architecture des Web services permet de développer des services Web qui encapsulent tous les niveaux de fonctionnalité métier. En d'autres termes, un service Web peut être très simple, par exemple renvoyer la température actuelle, ou bien constituer une application complexe. Cette architecture autorise également la combinaison de plusieurs services Web pour la création d'une nouvelle fonctionnalité. L'architecture des services Web a trois rôles distincts :

- un fournisseur,
- un demandeur,
- et un agent.

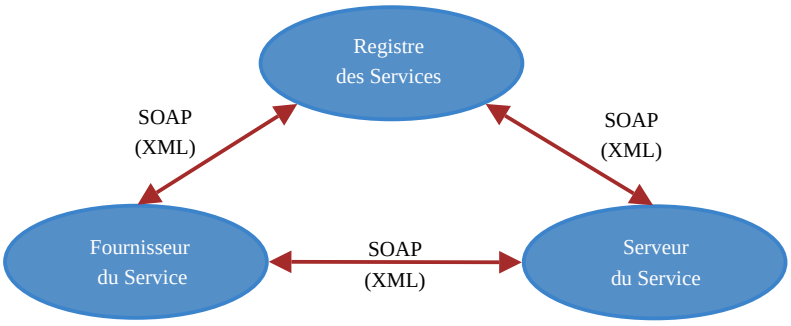

FIG. 5.6 – *Architecture d'un Web Service*

Le fournisseur crée le service Web et le met à la disposition des clients qui souhaitent l'utiliser. Un demandeur est une application client qui utilise le service Web. Le service Web demandé peut également être client d'autres services Web. L'agent, qui peut être par exemple un registre de service, permet au fournisseur et au demandeur d'un service Web de communiquer.

Les trois rôles de fournisseur, de demandeur et d'agent interagissent les uns avec les autres par l'intermédiaire des opérations de publication, de recherche et de liaison. Un fournisseur informe l'agent de l'existence du service Web en utilisant l'interface de publication de cet agent pour permettre aux clients d'accéder au service.

Les informations publiées décrivent le service et spécifient son emplacement. Le demandeur consulte l'agent pour localiser un service Web publié. Grâce aux informations sur le service Web obtenues par l'agent, le demandeur peut lier, ou appeler, le service Web. Ce diagramme résume la manière dont le fournisseur, le demandeur et l'agent interagissent les uns avec les autres.

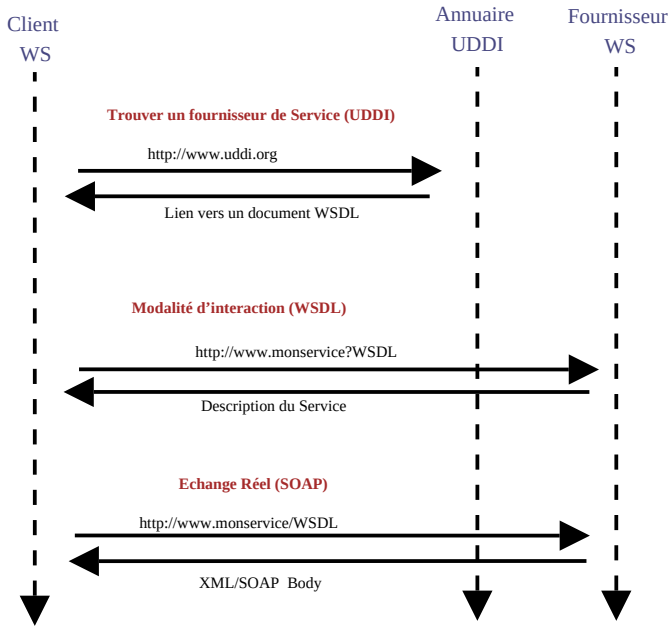

FIG. 5.7 – *Un Web Service en action*

5.6.2 Retour sur Le SOAP

SOAP (Simple Object Access Protocole) est un protocole de messagerie indépendant du transport. Il définit un ensemble de règles pour structurer des messages qui peuvent être utilisés dans de simples transmissions unidirectionnelles, mais il est particulièrement utile pour exécuter des dialogues requête-réponse RPC (Remote Procedure Call).

Le SOAP n'est pas lié à un protocole particulier mais le protocole HTTP est très populaire actuellement, les utilisateurs de SOAP l'utilisent généralement . Il n'est pas

non plus lié à un système d'exploitation ni à un langage de programmation, donc, théoriquement, les clients et serveurs de ces dialogues peuvent tourner sur n'importe quelle plate-forme et être écrits dans n'importe quel langage du moment qu'ils puissent formuler et comprendre des messages SOAP. En tant que tel, il s'agit d'un important composant de base pour développer des applications distribuées qui exploitent des fonctionnalités publiées comme services par des intranets ou internet.

Structure de base de SOAP

Un message du SOAP est un document XML qui consiste en une enveloppe du SOAP obligatoire, un en-tête du SOAP facultatif, et un corps du SOAP obligatoire (Fig. 5.8).

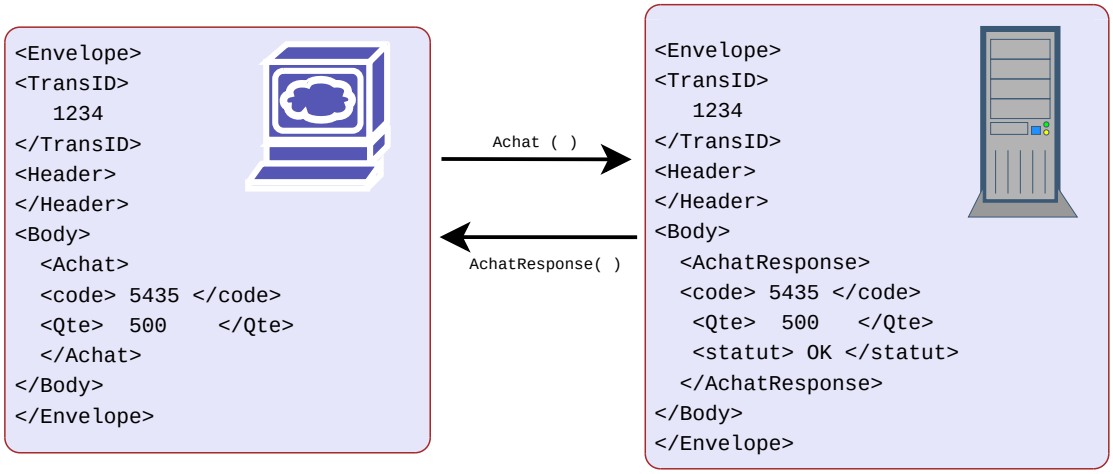

FIG. 5.8 – *Structure d'un message SOAP*

L'enveloppe: L'Enveloppe est l'élément du sommet du document XML qui représente le message.

- Le nom de l'élément est " Enveloppe "
- L'élément doit être présent dans un message du SOAP

L'élément peut contenir des déclarations du namespace aussi bien qu'attributs supplémentaires. Si présent, les tels attributs supplémentaires doivent être namespace -qualifiés. De la même façon, l'élément peut contenir des éléments des sous supplémentaires. Si présents ces éléments doivent être namespace - qualifiés et doivent suivre l'élément du Corps du SOAP.

L'en-tête: dont les caractéristiques sont les suivantes:

- Le nom de l'élément est " En-tête ".
- L'élément peut être présent dans un message du SOAP. Si présent, l'élément doit être le premier élément de l'enfant immédiat d'un élément de l'Enveloppe du SOAP.

- L'élément peut contenir un ensemble d'entrées de l'en-tête chacun être un élément de l'enfant immédiat de l'élément de l'En-tête du SOAP. Les éléments de l'enfant tout immédiats de l'élément de l'En-tête du SOAP doivent être namespace - qualifiés.

Le Corps: dont les caractéristiques sont les suivantes:

- Le nom de l'élément est "Corps".
- L'élément doit être présent dans un message du SOAP et doit être un élément de l'enfant immédiat d'un élément de l'Enveloppe du SOAP. Il doit suivre l'élément de l'En-tête du SOAP directement si présent. Autrement ce doit être le premier élément de l'enfant immédiat de l'élément de l'Enveloppe du SOAP.
- L'élément peut contenir un ensemble d'entrées du corps chacun être un élément de l'enfant immédiat de l'élément du Corps du SOAP. Les éléments de l'enfant immédiats de l'élément du Corps du SOAP peuvent être namespace - qualifiés. Le SOAP définit l'élément de la Faute du SOAP qui est utilisé pour indiquer des messages de l'erreur.

Exemple

Imaginez que vous ayez la base de données très simple d'une entreprise, comprenant une table spécifiant le numéro de référence, le nom et le numéro de téléphone des employés. Vous voulez offrir un service qui permet à d'autres systèmes de votre compagnie de consulter ces données. Le service retourne un nom et un numéro de téléphone (un tableau de chaînes de caractères à deux éléments) pour un numéro de référence d'employé donné (un entier). Voici un prototype Java de ce service:

$$String[\,]\ \ getEmployeeDetails\ \ (\ int\ employeeNumber\ );$$

Face à ce genre de problème [78], l'approche du développeur SOAP est d'encapsuler la logique de requête de la base de données, destinée à ce service, dans une méthode (ou fonction) en C ou VB ou Java etc. et ensuite, de démarrer un process qui écoute les requêtes adressées à ce service (un process listener), ces requêtes étant formulées dans un format SOAP et contenant le nom du service et les paramètres requis. Comme mentionné, la couche transport peut être HTTP mais pourrait tout aussi bien être SMTP ou autre chose. Puis, le listener, typiquement écrit dans le même langage que la méthode du service pour plus de simplicité, décode la requête SOAP entrante et la transforme en un appel de la méthode. Il récupère le résultat de l'appel de la méthode, l'encode dans un message SOAP (la réponse) et le renvoie au demandeur. Conceptuellement, cela donne (Fig. 5.9).

Soap est actuellement le protocole standard de communications des Web Services. Lorsque l'on parle de SOAP comme un protocole de communications, la plupart des gens pensent à DCOM ou à CORBA et se mettent à se poser des questions du type

- Comment SOAP réalise-t-il l'activation d'un objet? ou encore

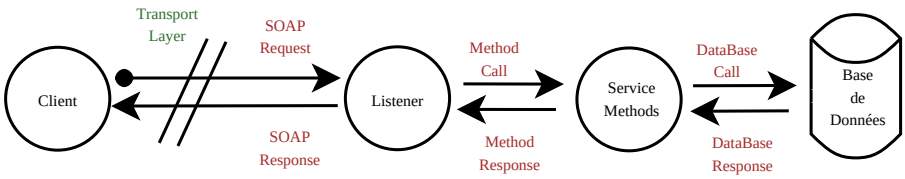

FIG. 5.9 – *Exemple d'utilisation de SOAP*

– Quel est le service de nom utilisé par SOAP ?

Si une implémentation de SOAP doit vraisemblablement s'intéresser à ce genre de choses, la norme SOAP ne les spécifie pas. SOAP est une spécification qui définit le format XML des messages (et c'est tout, pour ce qui est des parties obligatoires de la spécification). Si un fragment XML bien formaté se trouve intégré à des éléments SOAP, vous avez un message SOAP. Il existe d'autres parties de la spécification SOAP qui décrivent comment représenter les données d'un programme en XML et comment utiliser SOAP pour effectuer des appels de procédure distante (RPC).

Ces aspects facultatifs de la spécification servent à implémenter les applications de style RPC dans lesquelles un message SOAP contenant une fonction pouvant être appelée (ainsi que les paramètres à transmettre à cette fonction) sont envoyés par le client, et que le serveur renvoie un message contenant les résultats de la fonction exécutée.

La plupart des implémentations courantes de SOAP prennent en charge les applications RPC car les programmeurs qui ont l'habitude des applications COM ou CORBA comprennent le style RPC. SOAP prend également en charge les applications de style document dans lesquelles le message SOAP n'est qu'un wrapper (couverture) autour d'un document XML. Les applications SOAP de style de document sont très souples et nombreux sont les nouveaux Web Services qui tirent profit de cette souplesse pour créer des services qu'il serait difficile de mettre en oeuvre avec RPC.

La dernière partie facultative de la spécification SOAP définit l'aspect d'un message HTTP contenant un message SOAP. Cette liaison HTTP est importante car HTTP est pris en charge par presque tous les systèmes d'exploitation courants (ainsi que par de nombreux systèmes d'exploitation moins courants).

La liaison HTTP est facultative, mais presque toutes les implémentations SOAP la prennent en charge car c'est le seul protocole standardisé pour SOAP. C'est la raison pour laquelle il existe une idée fausse selon laquelle SOAP nécessite HTTP. Certaines implémentations prennent en charge les transports MSMQ, MQ Series, SMTP ou TCP/IP, mais la plupart des Web Services actuels utilisent HTTP parce qu'il est très répandu. HTTP étant l'un des principaux protocoles du Web, la plupart des entreprises sont dotées d'une infrastructure réseau qui prend en charge HTTP et ont le personnel qui en comprend la gestion.

Les infrastructures de sécurité, de surveillance et d'équilibrage de la charge pour HTTP sont déjà disponibles aujourd'hui. L'une des sources majeures de confusion lors du démarrage avec SOAP est la différence qui existe entre la spécification SOAP et ses nombreuses implémentations.

La plupart des utilisateurs SOAP n'écrivent pas les messages SOAP directement mais utilisent une boîte à outils SOAP pour créer et analyser les messages SOAP. Ces boîtes à outils traduisent généralement les appels de fonction depuis un langage donné en un message SOAP. Par exemple, Microsoft SOAP Toolkit 2.0 traduit les appels de fonction COM vers SOAP et Apache Toolkit traduit les appels de fonction JAVA vers SOAP.

Les types d'appels de fonction et les types de données des paramètres pris en charge varient en fonction de chaque implémentation SOAP, de sorte qu'une fonction qui fonctionne avec une boîte à outils peut ne pas fonctionner avec une autre. Il ne s'agit pas d'une limitation de SOAP mais plutôt de l'implémentation particulière que vous utilisez.

SOAP possède une fonction qui est de loin la plus extraordinaire : il est pris en charge par des plates-formes matérielles et logicielles différentes. Cela signifie que SOAP peut être utilisé pour relier des systèmes séparés au sein et en dehors de votre entreprise.

De nombreuses tentatives ont été faites par le passé pour trouver un protocole de communications commun qui pourrait servir à l'intégration des systèmes, mais aucun d'eux n'a été aussi répandu que SOAP. Pourquoi cela? Parce que SOAP est beaucoup plus petit et beaucoup plus simple à implémenter que beaucoup des protocoles précédents.

DCE et CORBA, par exemple, demandaient des années à implémenter, c'est pourquoi seules quelques implémentations ont été réalisées. En revanche, SOAP peut utiliser des bibliothèques HTTP et des parseurs XML existants pour exécuter les tâches les plus lourdes, de sorte qu'une implémentation SOAP peut être terminée en quelques mois. C'est pourquoi il existe aujourd'hui une pléade d'implémentations SOAP. Il est évident que SOAP ne fait pas tout ce que font DCE ou CORBA, mais c'est la simplicité des échanges de fonctions qui rend SOAP si facilement utilisable. La popularité d'HTTP et la simplicité de SOAP en font une base idéale pour l'implémentation de Web Services pouvant être appelés depuis pratiquement n'importe quel environnement.

SOAP et l'intoperabilite

Pour résoudre les problèmes d'interopérabilité entre composants logiciels, de nombreux protocoles ont été conçus ces dernières années. Jusqu'à la création de SOAP, trois protocoles étaient plus particulièrement utilisés :

- RMI (Remote Method Invocation). Ce protocole est très simple et très efficace, mais ne fonctionne que dans un environnement Java de bout en bout.
- CORBA/IIOP (Common Object Request Broker Architecture). Ce protocole, fruit du travail de l'OMG, permet de s'affranchir des architectures et des langages utilisés, objectif repris par .NET, à l'aide d'un langage neutre de définition d'interface IDL (Interface Definition Language) et d'un protocole commun de transport et de sérialisation des données. Malheureusement, l'OMG, en ciblant un public de professionnels de l'informatique très exigeant, a fourni une spécification très dense et une architecture au final très difficile à déployer hors quelques bons cas simples.
- COM (Component Object Model) et DCOM (Distributed Component Object Model). Ces protocoles ont été écrits par Microsoft pour faciliter la communication entre composants logiciels de Windows. Un portage UNIX a été réalisé par Software

AG, mais en pratique ce protocole reste largement restreint au monde Windows et dans le contexte des Intranets.

Pourquoi SOAP et non pas autres middlewares?

On remarque que chaque protocole imposait une barrière à une véritable interopérabilité, qu'elle soit la plate-forme (Windows ou Java pour DCOM et RMI) ou tout simplement la complexité avec CORBA. SOAP se différencie de ces protocoles par sa grande simplicité d'usage et sa neutralité. Les concepteurs de la norme se sont visiblement mis à la place des développeurs des services avec pour objectif de faciliter leur travail en supprimant les fonctionnalités les plus contraignantes de ses prédécesseurs. XML a été choisi comme format d'échange, ce qui permet d'utiliser la norme XML Schema pour la sérialisation des données. Les types autorisés sont limités à des cas volontairement simples (essentiellement chaînes de caractères, entiers, flottants et listes de ces derniers) mais suffisants pour reconstruire la plupart des objets. A la différence de ces derniers protocoles dont le format est binaire, les données transportées sont lisibles telles quelles par le développeur et leur manipulation par le client en est grandement facilitée. La structure choisie pour SOAP correspond à celle d'une enveloppe dans laquelle le message, les données, sont encapsulés, selon la figure suivante.

Autre point fort, le XML se prête bien à l'intégration dans un serveur Web. Ainsi, la norme de SOAP prévoie qu'il utilise le protocole HTTP pour transporter les données, bien que d'autres protocoles soient disponibles, dont SMTP, implémenté par Apache-SOAP. Comme l'écrit Microsoft dans la présentation des Web Services, SOAP sur HTTP n'est pas bloqué par les pare-feu (firewalls) des entreprises, ce qui advenait souvent avec CORBA, RMI et DCOM, rendant impossible une communication au-delà de l'Intranet. SOAP peut ainsi être servi efficacement par les serveurs d'application capables de gérer le XML, ce qui est le cas de IIS (Microsoft) et de J2EE. Le client retire également des avantages de SOAP. Le pré-requis sont peu nombreux et très répandus, à savoir HTTP et XML ; les nombreuses implémentations de SOAP déjà disponibles tendent à le démontrer. Selon une classification consacrée, le protocole SOAP appartient à la famille des " glues dynamiques ", c'est-à-dire que le déploiement (clients et serveurs) est léger et rapidement évolutif, à la différence de CORBA ou de RMI où l'installation de binaires et de descripteurs de déploiement lourds sont nécessaires pour interopérer. Pour qu'il reste simple, les concepteurs du protocole ont dû accepter des concessions importantes. SOAP n'inclut notamment pas dans sa spécification la gestion du cycle de vie des objets et les pointeurs vers des instances distantes d'objets, comme CORBA, COM ou RMI. La gestion des sessions et des transactions a été volontairement exclue de la norme.

5.7 Les Ressources distribuées

5.7.1 Le Cas de la pièce de Rechange

Nous rappelons que l'objectif de notre SMA au moment des son exécution est la réalisation d'un équilibrage de ressources, en particulier en matière de la pièce de rechange,

entre les différents sites du système. Pour ce faire, l'agent intelligent dit de prévention de chaque site détecte les situation anormales de sur-stockage et de sous-stockage, et en informe son superviseur qui entre en interaction, par principe de coopération, avec les autres agents superviseurs des autres sites qui composent le même groupe, afin d'assurer cet équilibrage. Cela se fait par le biais d'un agent facilitateur appelé le Coordonnateur des Accointances du Groupe (CAG), qui est, de son coté, en relation avec Coordonnateur Inter-Groupes (CIG) (voir chapitre 4).

5.8 Notre Modèle d'organisation

Le problème tel qu'il se présente n'obéit pas à un modèle d'organisation existant déjà. Nous avons développé le notre, sur la base de nos besoins, baptisé AGIRAT, il se base sur les éléments suivants :

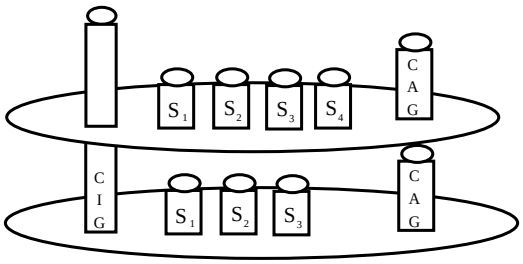

FIG. 5.10 – *Modèle d'organisation basé sur la notion du groupe*

- Agent : c'est l'entité conceptuelle de base de notre organisation, pour le moment un site est représenté, vis-à-vis de son environnement, par l'agent superviseur. Et le site lui-même peut avoir une structuration interne SMA ou autre. En conséquence, un agent est:
  - Autonome,
  - En interaction intra site avec les agents prévention et gestionnaire de stock et en interaction extra-site avec les autres agents superviseur du système,
  - Distribué,
  - possède un objectif,
  - Notre agent prévention est un agent cognitif car il est doté d'une forme d'intelligence ; il se base sur son passé pour prévenir son futur et agit au présent.
- Groupe: il représente l'organisation des agents dans un groupe selon un critère bien établi. De ce fait un agent peut appartenir à plusieurs groupes (Fig. **??**.).
- Interaction : Tout le système est basé sur l'interaction d'une entité vis à vis de l'autre, sans interaction, le système est statique.

- Rôle : Chaque agent à un rôle par rapport à son existence, mais exécute une ou plusieurs tâches à un instant $T_{current}$.
- Autorité (ou privilège) : l'autorité est la règle qui régit l'organisation, l'intérêt du groupe CAG doit passer avant celui de l'agent, et celui du CIG avant celui du CAG, sur cette base, une stratégie de l'ensemble peut être appliquée.

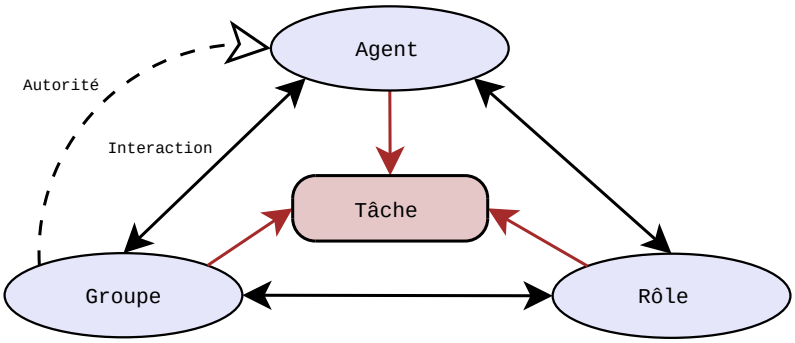

FIG. 5.11 – *Les concepts qui régissent notre modèle d'organisation*

- Et la tâche qui est la subdivision du rôle en séquences de tâches (en séquentiel ou en parallèle), c'est l'élément atomique par rapport aux SMA conceptuel. A l'implémentation, une tâche peut représenter un script s'exécutant sur un ou de plusieurs processus au sens informatique (Thread ou Processus Unix)

Pourqoui le critère " Autorité " dans notre modèle d'organisation?

En évaluant les coûts de chaque produit, le GAG associé à un ensemble de sites, et en dehors des critères classiques d'optimisation (prix unité, prix transport, Ĕ), doit prendre en considération les consignes directrices de l'entreprise étendue, et fait intervenir un coefficient dans l'évaluation de chaque produit sur des critères imposés d'avance selon une certaine stratégie :

$$Cout_produit = F(Prix - produit, Scoeff, ...)$$

Ce coefficient $Scoeff$ est imposé et varie dynamiquement selon la stratégie tracée de l'entreprise qui peut désigner par exemple une préférence de partenariat au détriment d'un certain gain sur le coût. Ce coefficient se répercute sur le coût proposé par le GAG au site Si demandeur.

Sur cette base, un site S_i n'est pas vraiment entièrement autonome, mais subit une forme d'"Autorité" qui est due à la stratégie de l'entreprise.

5.9 Notre modèle d'interaction

Dans le modèle d'interaction que nous proposons (Fig. 5.12.), nous nous basons sur un modèle similaire à celui des agences immobilières où, l'agence est intermédiaire et joue un rôle de négociation entre vendeurs et acheteurs, en évaluant toutes les propositions et présente, en principe, les meilleures occasions à ses clients, mais ne prend pas de décision à leur place, puisque le dernier mot revient aux concernés.

Dans notre cas, le CAG (Fig. 5.13) joue le rôle de cette agence, il reçoit les propositions des différents sites qui appartiennent au groupe et en même temps il guette les propositions d'offre ou d'acquisition de quantités d'articles. En cas de demande, il la propose au demandeur selon le script d'interaction suivant :

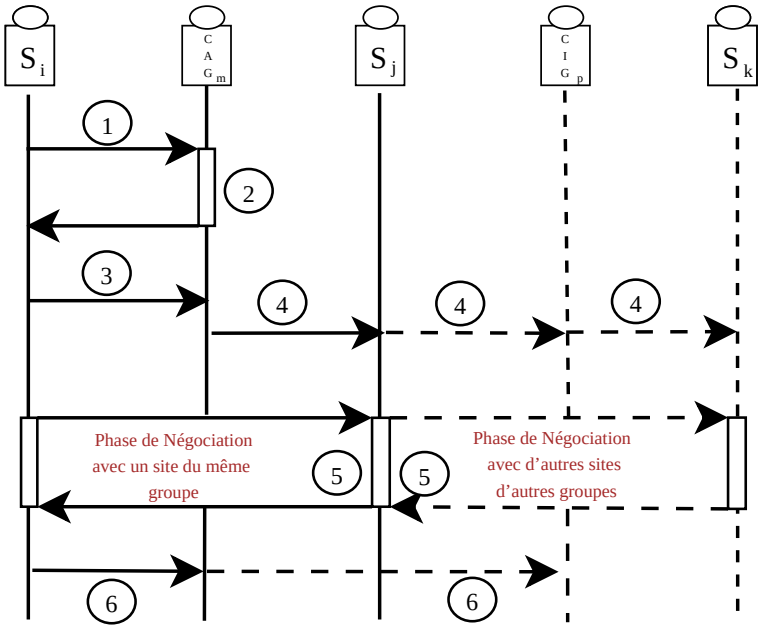

FIG. 5.12 – *Notre Modèle d'Interaction*

1. Le CAG_m reçoit périodiquement toutes les informations des sites faisant parti de son groupe ; elles concernent essentiellement les niveaux de stock ($P_{k,j,surplus}$) à offrir et les ($P_{k,j,,deficit}$) à demander.
2. Ce dernier évalue toutes les propositions et établit les meilleurs coûts pour chaque article k .
3. Le site S_i veut acquérir une quantité $P_{k,i,deficit}$ ou offrir une quantité $P_{k,i,surplus}$, émet son désir à son CAG_m.
4. Le CAG_m sait déjà le site qui propose le coût optimal, alors il lui envoie un message de proposition. Mais ne prend pas de décision, c'est au concerné de le faire. A signaler uniquement qu'en cas d'indisponibilité d'article au sein du groupe, le

```
Faire_ en_parallèle
  Pour chaque message reçu par GAG_m faire
    Si message est receive(S_m,P_{k,m,prev},acheter,T_{rupture}) alors
        evaluer l'offre
        insérer l'offre dans la liste des acheteurs du produit No k( )
        Si existe un contractant S_j ∈ GAG_m alors
          Send(S_j,P_{k,i,},achat,T_{rupture})
        Sinon
          Send(GIG,P_k,,achat,T_{rupture})
        Finsi
    Sinon
        Si message est receive(S_m,P_{k,m,prev},vendre,T_{courrant}) alors
          évaluer l'offre
          insérer l'offre dans la liste des vendeur du produit No k ( )
          Si existe un contractant Sj ∈ GAG_m alors
            Send(S_j,P_k,i,,vente,T_{courrant})
          Sinon
            Send(GIG,P_{k,},vente,T_{courant})
          Finsi
        Finsi
    Finsi
  Finpour
Fin_Faire_en_parallèle
```

FIG. 5.13 – *Le tableau de bord de l'agent CAG*

CAG_m associé au groupe doit faire appel à d'autres sites, S_k, d'autres groupes via l'agent Coordonnateur Inter-Groupes (CIG_p).

5. Négociation entre les sites (S_i et S_j) ou (S_i et S_k) sur le coût de la quantité d'article k et cela jusqu'à signature du contrat.
6. Dans tous les cas, le CAG_m doit être informé de l'issue des négociations et en cas de désaccord, le CAG_m propose l'offre d'ordre $(n-1)$ avec un autre site S_r avec $r \neq i$ et tout en ré-appliquant toutes les étapes de (1 à 6) de nouveau.

Cette opération se renouvelle jusqu'à signature et concrétisation d'un contrat.

5.9.1 Etude des cas exceptionnels dans notre modèle d'interaction

Un modèle d'interaction, une fois en oeuvre, dépend généralement de la plate-forme réseau, local ou Internet. par conséquent, il n'est pas à l'abri de situations exceptionnels. Dans ce cadre, nous avons entrepris une étude avec validation sur réseau [10] qui a permis de dégager un certains nombre de cas exceptionnels:

- un agent ne répond pas,
- un agent répond hors délai,

- un agent n'accomplit pas ce qu'il est censé faire,
- un agent tombe en panne,
- etc.

Nous utiliserons le jargon commun utilisé dans les SMA, à savoir administrateur, pour celui qui diffuse une requête sur les agents du système et les contractants ceux qui s'engagent avec cet administrateurs à répondre à cette requête.

5.9.2 Les cas d'exception

On analyse dans un premier temps les protocoles et les cas exceptionnels lorsqu'il qu'il y a un seul administrateur à un instant donné, et on généralisera cette étude à un nombre quelconque d'administrateurs simultanés.

Avec un seul administrateur

Un agent peut être à la fois administrateur et offrant, notre protocole d'interaction permet en effet à un administrateur qui a besoin de faire effectuer une tâche par d'autres agents plus compétents ou disponibles par le biais d'un appel d'offre. Lorsque ce dernier a reçu tous les acquittements, qu'il soit porteur d'une proposition positive ou négative, cet administrateur sait qu'il détient toutes les propositions et qu'il peut commencer son évaluation. Prenons ici l'exemple classique des robots: ou un robot ayant trouvé un filon joue le rôle d'un administrateur. Lorsque ce dernier reçoit tous les acquittements, il doit choisir l'agent le plus proche du filon.

Exceptions et solutions

L'administrateur doit attendre toutes les réponses à son appel d'offre avant d'évaluer les propositions dans le but de mieux tenir compte de l'ensemble des potentialités du système en n'oubliant aucune offre, mais au prix de deux inconvénients majeurs : Tous les agents doivent répondre, même ceux qui ne sont pas à priori intéressés par l'appel d'offre. De ce fait, le réseau risque d'être saturé par une trop grande quantité de messages ne contenant que très peu d'information et portant uniquement la mention "pas intéressé".

Un agent ne répond pas: Un agent en mauvais fonctionnement peut totalement bloquer le système, de plus les agents sont emmenés à se retrouver dans des situations où leurs intérêts peuvent être contradictoires, ils sont alors en situations de conflits. Ces situations proviennent essentiellement d'un problème d'accès à des ressources insuffisantes, buts incompatibles, compétences insuffisantes, etc. Il suffit qu'il ne puisse pas envoyer une réponse pour que l'administrateur l'attende indéfiniment. Dans la (Fig. 5.14.(a)), Ag2 est en mauvais état de fonctionnement, l'administrateur ne doit pas l'attendre indéfiniment.

Un agent tombe en panne: La défaillance d'un réseau au niveau d'un lien peut conduire à une partition du réseau (Fig. 5.14.(b)). Une panne affectant un agent peut provoquer l'arrêt total de ce dernier alors cet agent ne peut pas recevoir l'appel d'offre de l'administrateur et automatiquement il ne répond pas et il se trouve donc dans

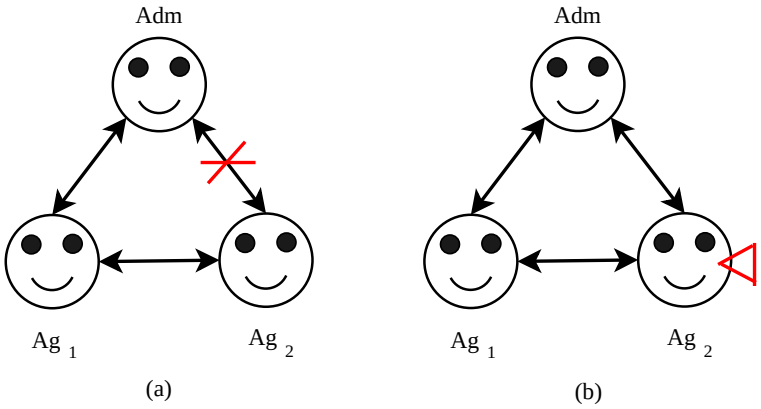

FIG. 5.14 – *Cas d'agent qui ne répond pas*

la même situation précédente et que l'agent administrateur l'attende indéfiniment. Pour résoudre ces problèmes il ne faut pas demander aux agents de retourner de messages s'ils ne sont pas intéressés ou évidemment s'ils sont dans l'incapacité de le faire. Une autre solution consiste à ce que tous les agents soient mémorisés (avec ses propriétés qui expriment ses capacités) dans une structure de donnée globale appelée la table d'accointances conservée par un agent unique (Fig. 5.15 (a)). Si un agent intègre ou quitte le réseau ou encore il voit ses compétences modifiées, il en informera le réseau en diffusant l'information, et lorsqu'un administrateur a besoin d'effectuer une tâche. Il appelle cette table d'accointances et à partir d'elle, il est possible d'obtenir une liste de tous les agents présents, ayant une compétence dans un domaine bien défini. Mais le fait que cet administrateur s'adresse à tous les agents du réseau ne doit pas l'handicaper du coté temps. Pour cela, il faut qu'il rejette les propositions après un certain délai, appelé délai d'expiration.

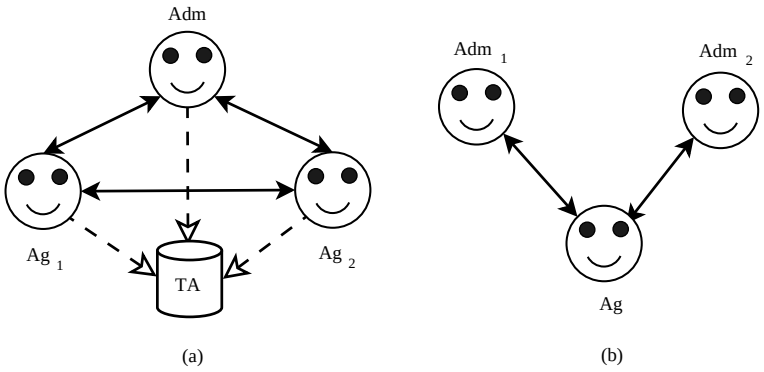

FIG. 5.15 – *(a) Compétences d'agents (b) Cas de plusieurs administrateurs*

Un agent répond hors délai: Les propositions qui arrivent après la date d'expiration

seront rejetées même si l'agent possède la compétence suffisante à réaliser la tâche demandée. La solution est qu'il faut bien calibrer ce temps de réaction de manière à ce que la plus grande partie des propositions aient le temps de retourner à l'administrateur. Si trop de demandes arrivent après la date limitée, on augmente le temps de réaction et si on attend trop après la réception des dernières réponses on diminue ce temps. De ce fait, si les agents répondent tous avec le même temps de réponse, le système convergera vers une valeur bien adaptée.

Un agent n'accomplit pas ce qu'il est censé faire: Lorsqu'un agent B indique qu'il accepte une tâche et qu'il dispose des compétences nécessaires pour l'accomplir et que, pour des raisons quelconques (B se trouve dans un mauvais fonctionnement : un composant de B tombe en panne, etc.), B voit ses compétences modifiées et n'ait plus la capacité d'accomplir la tâche T.
Afin de résoudre ce problème, il faut que B transmette son changement d'état et rompe le contrat avec l'administrateur qui mémorise que B ne sait plus faire T et relance l'appel d'offre à d'autres agents.

L'influence des sous contractants:

Lorsqu'un offrant X exécute une tâche T sous contrat avec un administrateur A, il peut être amené à décomposer cette tâche en sous tâches et ne pas être capable de réaliser certaines de ces sous-tâches lui-même. Alors il doit se comporter comme un administrateur et relance un appel d'offre pour l'ensemble des sous-tâches T_1,,T_n issues de la tâche T. Un offrant confronté à une telle situation peut n'affecter cet appel d'offre qu'après avoir reçu le contrat ou bien demander d'abord aux sous contractants de s'engager à effectuer ces tâches avant qu'il ne fasse lui-même des propositions à l'administrateur A. Évidement il s'agit d'un problème récursif et les sous contractants peuvent être amenés à faire appel à des sous-contractants et ainsi de suite.

Avec plusieurs administrateurs

C'est le cas où plusieurs administrateurs peuvent travailler en même temps et donc émettent leurs appels d'offre simultanément (Fig. 5.15 (b)). Les administrateurs de tâches n'ont pas généralement d'idées sur les connaissances locales des autres administrateurs. En effet qu'à cause de la localité du processus, les agents subissent trois types d'ignorances :

– L'ignorance temporelle: elle provient du fait que les offrants ne connaissent que les appels d'offre déjà reçus et non ceux qui sont sur le point d'arriver.

	A_1	A_2
Ag_1	75	60
Ag_2	60	20

TAB. 5.1 – *L'ignorance spatiale.*

- L'ignorance spatiale: provient de l'incapacité pour les administrateurs de savoir quels sont les autres appels d'offre en cours et pour les offrants de connaître les autres propositions. par exemple : Dans le tableau 5.1, à la réception des propositions, l'administrateur A_1 choisira normalement Ag_1, car sa proposition semble plus intéressante que celle de Ag_2, et de même, A_2 choisira Ag_1 puisque sa proposition dans ce cas est aussi bien meilleure que celle de Ag_2, de ce fait Ag_1 va être énormément chargé pendant que Ag_2 sera inactif.
- L'ignorance sur les charges: est due à ce que les offrants ne reconnaissent pas les charges de travail des autres agents. De ce fait, un agent qui travaille peut être amené à gagner devant un autre dont la charge est presque nulle. Pour remédier à ce problème, les agents doivent disposer d'informations portant sur les charges des autres, de manière à modifier leur proposition.

5.9.3 Critique de notre modèle d'interaction

Notre modèle d'interaction apporte une grande souplesse dans la manière de gérer l'accord entre les clients et les fournisseurs. Il ne s'agit cependant pas d'un modèle de résolution de problèmes mais seulement d'un protocole d'intercation et de répartition dynamique de travail.

Ses avantages sont:

- Notre modèle offre une grande dynamité. Il suffit qu'un agent soit capable de s'inscrire au réseau et de le quitter pour qu'il soit possible de prendre facilement en compte l'apparition de nouveaux agents et la disparition des anciens.
- La correspondance entre tâche et agent est élaborée à partir d'un accord bilatéral entre client et fournisseur, ce qui permet de tenir compte d'un grand nombre de paramètres tels que les compétences des agents, la charges de travail, le type de la tâche à effectuer, la description des opérations, le type des données à fournir, la durée d'attente maximale autorisée, le coût, l'urgence, etc.

Ses inconvénients sont:

- Si la mise en oeuvre est simple, l'exécution est lourde et produit un grand nombre de messages. Et de ce fait, Ce modèle n'est applicable qu'a de petites sociétés d'agents dans lesquelles seules des tâches de hauts niveaux sont traitées par appel d'offre.
- La simplicité de mise en oeuvre n'est qu'apparente et, qu'il faut que, soit le nombre d'appels d'offre soit faible, soit que l'on implémente une structure complexe de gestion des problèmes liés au parallélisme du processus.
- Enfin lorsque les tâches sont redécomposables, il est nécessaire que les agents aient une stratégie de décision qui leur permet de choisir entre un engagement rapide, tardif ou par embauche.

5.10 Aspects implémentatifs

Nous avons vu dans le paragraphe "Facteurs de qualité du logiciel" au début de ce chapitre qu'une politique d'integration de logiciels dans un travail en cours doit être définie avant de commencer tout développement de progiciel, particulièrement dans le domaine des e-plate-formes. Car développer tout à la base serait une tâche pas facile surtout s'il y a une bonne étude préalable de ce qu'il faut utiliser et de ce qu'il faut développer!

Plusieurs interrogations à qui il faut trouver des réponses?

5.10.1 L'utilisation de plate-forme dédiée SMA

Comme il est dit dans le chapitre 1, différentes plate-formes dédiées SMA sont proposées (PHOENIX, MadKit, SWARM, DARK, DIMA, etc.), et la plupart d'entre elles se veulent génériques et s'adapter à n'importent quelles applications.

Notre réponse est que l'utilisateur de ces plate-formes est contraint d'utiliser un langage dépendant de cette plate-forme (généralement Java) et il est contraint aussi d'utiliser le middleware imposé par cette même la plate-forme.

Pour notre part, nous avons fait des tests sur la plate-forme MadKit, et il en ressort que si le nombre d'agents en état d'activation dépasse 10 et avec de légers scriptes, la plate-forme ne répond plus.

Par ailleurs, le middleware utilisé est le socket. Ce dernier est valable sur un réseau local, mais ne peut pas outrepasser les pare-feux. Donc il n'est pas possible de l'utiliser sur une e-plate-forme.

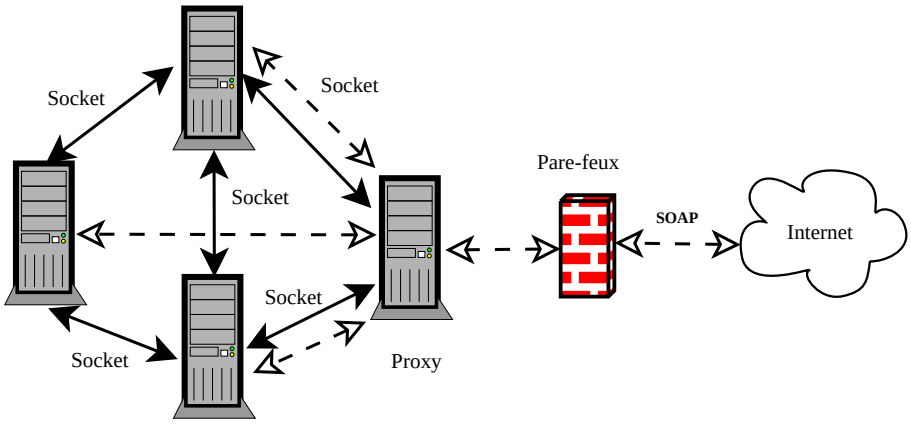

FIG. 5.16 – *Communication Socket vs SOAP*

Cette contrainte nous impose l'utilisation d'autres middlewares, et là le choix est presque évident, c'est le SOAP/XML.

5.10.2 Notre solution en Web-Services

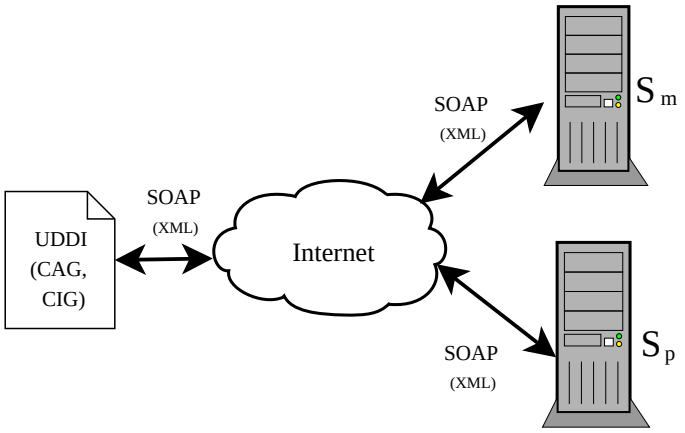

FIG. 5.17 – *Web-Service (S_m, CAG/CIG, S_p)*

Nous avons présenté le principe d'un web-services dans ce chapitre. Nous rappelons ici que ce principe se base sur trois acteurs logiciels distribués (serveur du service, annuaire des services et le demandeur du service) qui sont sur le web et communiquent entre eux en utilisant un protocole de communication de type HTTP ou HTTPS.

Nous rappelons aussi que notre modèle d'organisation SMA se base sur un agent facilitateur représentant d'un groupe appelé Coordonnateur des Accointances du groupe (CAG) dont le rôle est de regrouper des doléances, offre si sur-stockage ou demande si sous-stockage, des membres appartenant au même groupe.

Ce CAG dont le scripte est :

- Réceptionner les offres d'appels ( )
- Optimiser les coûts ( )
- Classer les offres ( )
- Répondre aux requêtes émanant des sites selon le meilleur côut ( )

Chaque site S_m demandeur d'un article est considéré comme demandeur de service auprès d'un autre site S_p acquéreur d'articles après avoir consulter l'annuaire du CAG qui affiche les meilleures offres donc meilleurs services.

Par opposition, chaque site S_m acquéreur d'un article est considéré comme demandeur de service auprès d'un autre site S_p demandeur d'articles après avoir consulter le même annuaire du CAG qui affiche les meilleures demandes donc meilleurs services.

Exceptionnellement, et comme montré dans le protocole d'interaction de notre solution proposée, si une requête de la part d'un site S_m ne trouve pas de réponse favorable dans l'annuaire du CAG, c'est le Coordonnateur Inter-groupes qui prendra en charge cette requête et remplace le CAG.

Reste maintenant à dire que pour pouvoir réaliser une communication entre ces agents logiciels distribués via le Web, le moyen le plus propice est l'utilisation du middleware SOAP qui, rappelons le, est encodé en XML et véhiculé via le protocole HTTP ou HTTPS.

Ce middleware, même s'il possède cette qualité de lent par rapport à d'autres middleware comme Bus-Corba, il a néanmoins cette qualité d'arriver au destinataire via le HTTP ou HTTPS.

En résumé de ce paragraphe, nous insistons sur le fait que cette architecture en Web-Service de notre solution (S_m, CAG, S_p) ou (S_m, CIG, S_p) est implémentable sur n'importe quelle plate-forme déployant les Web-Services (J2EE, .Net, etc.).

5.10.3 Implémentation d'un Agent

L'utilisation d'une plate-forme SMA présente certe des facilités d'implémentation, car tous les outils de base sont disponibles, mais présente l'inconvénient d'être dépendant de ces outils quant au développement d'applications industrielles de grande envergure fonctionnant sous réseau. C'est pour cette raison que nous sommes intéressés au développement de ces principes à la base, car ce choix ne nous impose, ni le langage, ni le middleware ni la plate-forme logicielle de base (Système d'exploitation).

Comme l'agent est une entité autonome capable d'agir dans son environnement, on peut l'assimiler à un objet doté d'une activité, cette dernière peut être concrétisée par un processus (au sens Unix) ou un thread.

L'ajout d'un processus à un objet lui permet d'avoir un contrôle sur lui-même.

Nous avons fait une projection des ces concepts en langage Java, sous Linux et en C# sous MSWindows. Sur cette base, un agent réactif est équivalent à :

- La mémoire d'un agent par une mémoire locale associée à un thread (processus léger), ce qui nous permettra de simuler la distribution des agents,
- Un script d'un agent par un objet Java, représentée structurellement par une classe Java (attributs + méthodes).
- L'activité de l'agent par celle d'un thread (ou plusieurs), ou un processus (au sens Unix), de façon à ce que toutes les tâches se déroulent en même temps (en multitâche).
- Une base de connaissance représentée, soit par un simple fichier, ou si les informations représentant les connaissances de l'agent sont énormes, faire recours à une base de Données Access, Oracle ou Informix), associée à des règles de recherche et de génération de l'information,
- Une Communication Bidirectionnelle ou multidirectionnelle (Communication entre agents), ou même en broadcast, en utilisant les Sockets, le bus OMG-Corba, Java-RMI, DCOM, XML-RPC ou comme dans notre cas de e-plate-forme le SOAP/XML.

Il faut signaler qu'avant que le message SOAP/XML est envoyé, les agents, obéissant au modèle d'organisation présenté et interagissant selon le modèle d'interaction proposé, chacun de son coté, déroule son script (voir chapitre 4) et obtient les résultats qu'ils faut

```
POST /book/servlet/Achat HTTP/1.1
Host: 216.2.13.156
User-Agent: Mozilla/5.0 (Windows; U; Win98; en-US; m18)
Gecko/20001108 Netscape/6.0
Host: Localhost:8080
Content-Type: Text/xml
SOAPMethodName: http://www.w3.com/soapAchat
<Envelope>
  <Body>
   <inv: Achat xmlns:inv="http://www.w3.com/soap">
    <MSG>Achat </MSG>
    <Code>789654</Code>
    <Qte>1000</Qte>
    <DateLimite>01/06/2005</DateLimite>
   </inv: Achat>
  </Body>
</Envelope>
```

FIG. 5.18 – *Le message SOAP d'un Achat d'un article*

encoder. La figure 5.20 montre le traitement de l'agent prévention associé à deux agents de notre SMA.

implémentation via un Réseau local

Nous avons présenté un modèle d'organisation SMA, associé à un modèle d'interaction entre agents du système, en tenant compte de l'implémentabilité des concepts avancés. Pour cela, une plate-forme de e-maintenance est, par excellence, le meilleur moyen pour tester ces concepts. Un jeu réel de données, d'articles appartenant à la classe C (voir chapitre 4), a été testé sur une application distribuée, sur des plate-formes logicielles hétérogènes (Linux et Windows). Elle a été écrite en langage Java, et validée sur réseau local, avec comme middleware les sockets.

implémentation via une e-plate-forme

Cette solution, si elle doit être implémentée sur une e-plate-forme, utilisant le protocole HTTP (Fig. 5.17.), elle doit utiliser comme support de communication et de diffusion de l'information un middleware qui peut infiltrer les pare-feux des réseaux locaux. Le seul moyen est l'utilisation d'un port ouvert sur ces pare-feux qui est celui de HTTP/HTTPS, sinon il n' y a pas d'internet pour les utilisateurs du réseau local.

Cette exigence requiert le développement, mais surtout l'adaptation de nos agents à la technologie XML, comme moyen de représentation de nos données, et le middleware SOAP comme moyen véhiculant les données via l'Internet au dessus du protocole HTTP, concernant l'offre et la demande des articles, entre agents du système.

```
POST /book/servlet/Vente HTTP/1.1
Host:216.2.13.156
User-Agent: Mozilla/5.0 (Windows; U; Win98; en-US; m18)
Gecko/20001108 Netscape/6.0
Host: Localhost:8080
Content-Type: Text/xml
SOAPMethodName: http://www.w3.com/soapVente
<Envelope>
  <Body>
   <inv: Vente xmlns:inv="http://www.w3.com/soap">
    <MSG>Vente </MSG>
    <Code>789654</Code>
    <Qte>500</Qte>
    <DateLimite>01/07/2005</DateLimite>
   </inv: Vente>
  </Body>
</Envelope>
```

FIG. 5.19 – *Le message SOAP d'une Vente d'un article*

Le message SOAP d'achat d'une quantité d'articles et encodé en XML est illustré par le code de la Figure 5.18, alors que celui de la vente est illustré par la Figure 5.19.

5.11 Langage de communication entre nos agents

Dans le chapitre 2, nous avons énoncé que les communications dans les systèmes multi-agents comme chez les humains. Ils sont à la base des interactions et de l'organisation. Une communication peut être définie comme une forme d'action locale d'un agent vers d'autres agents [28].

Un modèle de communication doit répondre aux interrogations suivantes:

- Le site S_m communique quoi?
- Le site S_m communique à qui?
- Quand Le site S_m communique t-il?
- Pourquoi Le site S_m communique t-il?
- Et Comment Le site S_m communique t-il?

Grâce au principe de la coordination, un système multi-agents peut réaliser ses tâches avec plus d'efficacité qu'un seul agent. Mais pour coordonner l'activité d'un ensemble hétérogène d'agents représentants des sites autonomes, il faut que les agents communiquent dans un langage compréhensible par tous les autres. On observe que dans un système ouvert un tel langage peut constituer une interface entre les agents.

L'utilisation d'un langage commun implique que tous les agents comprennent son

vocabulaire sous tous ses aspects concernant[69]:

- La syntaxe, qui précise le mode de structuration des symboles;
- La pragmatique pour pouvoir interpréter les symboles;
- L'ontologie pour pouvoir utiliser les mêmes mots d'un vocabulaire commun.

La compréhension du sens des symboles, ou à quoi les symboles font référence, demande un standard rigoureux de la sémantique et de la pragmatique. De plus il est nécessaire que les agents sachent bien utiliser le vocabulaire pour atteindre leurs buts, éviter les conflits, coopérer pour exécuter leurs tâches et modifier l'état mental d'un autre agent.

Avec l'avènement de la technologie XML, il y a une remise en cause des langages classiques dédiés SMA tels que ACL/KQML, principalement sur des e-plate-formes. Car ces derniers n'arrivent pas à véhiculer les donner sur des protocoles tels que HTTP/HTTPS.

Et justement c'est là l'argument principal qui nous a poussé à opter pour XML entant que moyen pour interpréter nos messages entre agents, et les envelopper dans SOAP afin des les envoyer aux destinataires.

Qu'est ce qu'un message?

Nous nous sommes bornés à notre cas précis, qui concerne l'offre ou la demande entre sites coopérants d'un Système-Multi-Agents.

En effet, un message est de la forme:

$$SEND\ \ (destinataire,\ code\_article,\ \{offre/demande\},\ date)$$

Ce message est encodé en XML (voir exemple 5.19 et 5.18), enveloppé en XML/SOAP et envoyé au destinataire, qui de son coté utilise la même technique pour répondre.

A partir de ce cas, plusieurs possibilités peuvent apparaître, un destinataire peut être:

- Le Coordonnateur des Accointances du Groupe CAG,
- Le Coordonnateur des Accointances du Groupe CIG,
- Un agent prévention AP
- Un agent Gestionnaire de stock AGS
- Ou n'importe quel autre site S_i.
- Broadcasting (Tout le monde), symbolisé par "$*$"
- Moi-même (Self), symbolisé par "."
- Agent destinataire indéterminé , symbolisé par "_" .

L'exemple 5.20 montre ce que doit être un message de coopération entre deux agents représentants de sites d'un même SMA qui entrent en coopération pour la réalisation d'un équilibrage de ressources entre eux.

Sachant que pour un site quelconque et d'après notre architecture SMA, les deux agents prévention et gestionnaire de stock ne peuvent communiquer qu'avec leur superviseur et qu'un agent superviseur représentant d'un site est seul habilité à représenter

son site et, par conséquent, il peut envoyer un message à un autre superviseur d'un autre site.

5.12 Conclusion

Dans ce chapitre, nous avons exposé nos deux modèles clés qui nous permettent de bien mettre en oeuvre notre SMA via le Web, à savoir:

- Le modèle d'organisation que nous avons baptisé AGIRAT (Agent, Groupe, Interaction, Rôle, Autorité et Tâche)
- Le modèle d'interaction, basé sur des agent facilitateurs, à savoir le CAG (Coordonnateur des Accointances du Groupe) et CIG (le Coordonnateur Inter-Groupe).

Ces deux modèles sont complémentaires, c'est-à-dire que la conception de l'un doit prendre en considération l'autre.

Ensuite, nous avons exposé les outils technologiques, en particulier les middlewares:

- Corba
- Java-RMI
- XML-RPC
- DCOM
- et SOAP

Nous avons montré pourquoi le choix de l'utilisation d'un SOAP/XML pour une plate-forme de e-maintenance. Et avons montré aussi comment utiliser ce middleware comme moyen pour interpréter nos messages entres agents de notre SMA durant l'activation du modèle d'interaction.

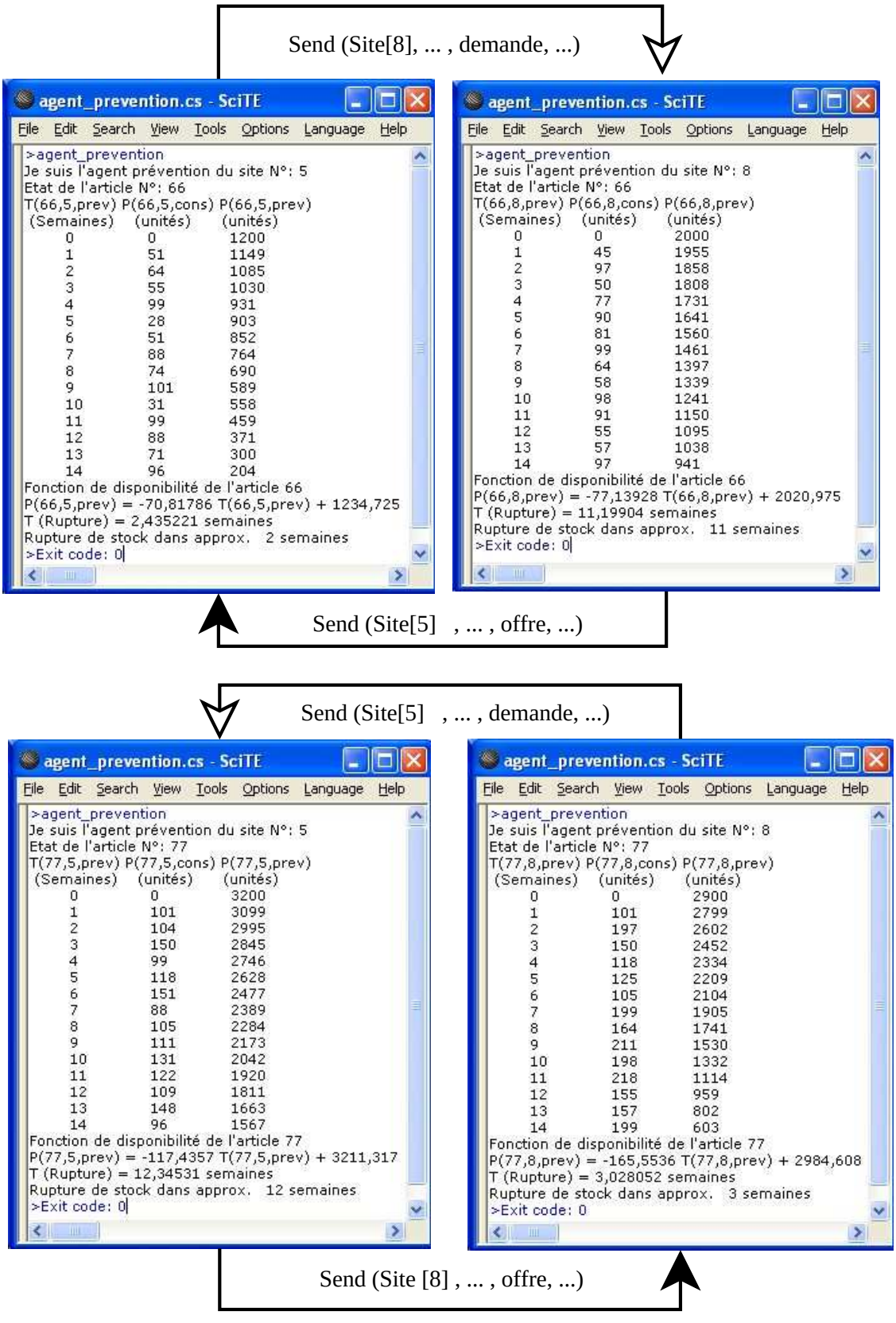

FIG. 5.20 – *Coopération réelle entre sites*

Conclusion générale

Nous avons présenté dans cette thèse les éléments nécessaires pour la modélisation, par l'approche des SMA, d'une gestion préventive et coopérative des ressources dans un environnement coopératif multi-sites. Nous avons présenté aussi les éléments d'implémentation du problème sur une e-plateforme générique de maintenance d'une entreprise étendue.

Compte tenu de l'importance des ressources pour le bon fonctionnement de ce genre d'entreprise, nous avons entrepris une étude, sur une gestion de stock de ressources et non pas une tenue de stock des ressources, dans un environnement distribué bien sur, mais surtout coopératif.

Cela a consisté en l'étude de tous les aspects qui nous permettent de faire une gestion "préventive" de la quantité de ressources dans chaque site qui composent l'entreprise et décider, au moment opportain, d'entrer en interaction via une e-plateforme générique, avec les autres sites, en offrant le sur-stockage ou en acquérant le besoin en sous-stockage qui permet au site de bien fonctionner et gérer ses ressources d'une façon optimale. Cela doit se faire, pour chaque ressource et pour tous les sites.

Cette façon de faire fait émerger une situation d'équilibrage de charge en matière de ressources, que nous avons appelé tout simplement "équilibrage de ressources", entre tous les sites de l'entreprise. Et permet à cette dernière à ne pas tomber dans des situations anormales de dépassement de seuils tolérés en sur-stockage ou en sous-stockage, pour chaque site et pour tous les articles.

Ce travail a commencé par une étude exhaustive de l'état de l'art de ce qui a été fait et se qui se fait actuellement dans des domaines scientifiques très connexes, à savoir:

1. Les ressources distribuées:
   - les ressources distribuées, sous tous leurs aspects: gestion, modélisation,etc.
   - éléments d'implémentation,
   - Equilibrage des Charges en matière de ressources,
   - et les interactions possibles entre ressources.
2. Outils de modélisation
   - Les Systèmes Multi-agents,
     - acteur et agent,
     - SMA,
     - Plateformes

   - Langages SMA dédiés
   - Plateformes SMA dédiés,
3. Les éléments logiciels ayant trait avec les systèmes distribués:
   - Plateformes logiciels dédiées,
   - possibilités des langages à gérer les systèmes distribués,
   - et les éléments de communication dans les langages.
4. Les Technologies Web utilisées dans le domaine, à savoir:
   - les e-plateformes,
   - les middlewares qui supportent le Web,
   - et la technologie SOAP/XML.

Un type de ressource que nous avons étudié est la pièce de rechange, qui est une ressources des plus importantes, d'une entreprise étendue.

Sa gestion consiste à se doter au préalable, pour chaque article, et pour chaque site, du maximum d'informations fiables sur cette ressorce afin de la classifier (A, B ou C), et lui adopter une des quatre stratégies de réapprovisionnement présentées dans le chapitre 4.

L'architecture multi-sites distribuée et coopérative de l'entreprise étendue requiert une modélisation structurelle et même comportementale par l'approche SMA. Avec comme base d'interaction, un modèle d'interaction que nous avons proposé et qui est inspiré des réseaux contractuels. Dans ce cadre, nous avons prévu, entre autres, et pour chaque site, un agent de prévention dont le comportement est la surveillance du site, à des périodes précises, afin de trouver une fonction représentative pouvant approximer les points critiques et optimaux de réapprovisionnement. Et ce, dans l'objectif d'assurer un équilibrage optimal de ressources entre tous les sites de cette entreprise.

C'est Sur cette base que s'est basé notre contribution, avec comme éléments "noyaux", les deux modèles connexes des Systèmes Multi-agents:

A. Le modèle d'organisation: Le problème tel qu'il se présente n'obéit pas à un modèle d'organisation existant déjà. Nous avons développé le notre, sur la base de nos besoins, baptisé AGIRAT, il se base sur les éléments suivants :

   - Agent : c'est l'entité conceptuelle de base de notre organisation. Pour chaque site, trois agents ont été prévus, agent prévention, agent gestionnaire de la base de connaissance et l'agent superviseur. Et dont le premier est un agent cognitif et les deux autres réactifs. L'agent superviseur est représentant du site vis-à-vis de l'environnement, c'est-à-dire les agents intervenant dans la e-plate-forme.
   - Groupe: il représente l'organisation des agents dans un groupe selon un critère bien établi. Dans notre cas, deux agents dédiés pour ce rôle, l'Agent des Accointances du groupe (CAG) et l'Agent Inter-Groupes (CIG). De ce fait un agent peut appartenir à plusieurs groupes et l'intersection est permise.
   - Interaction : Tout le système est basé sur l'interaction d'une entité vis à vis de l'autre, sans interaction, le système est statique. Notre modèle d'interaction

définit tous les scripts possibles qui permettent des réaliser cette interaction, qui est la clé de tout SMA. Notons dans ce cadre que l'interaction entre les éléments du système est provoquée par les différentes communications utilisées en utilisant un middleware dédié. Le langage utilisé n'est pas un langage classique FIPA tels que ACL ou KQML car nous avons utilisé XML comme langage sémantique de représentation de nos données et de messages aussi.

- Rôle : Chaque agent à un rôle par rapport à son existence, mais exécute une ou plusieurs tâches à un instant $T_{courant}$.
- Autorité (ou privilège) : l'autorité est la règle qui régit l'organisation, l'intérêt du groupe, car le l'intérêt du CAG doit passer avant celui de l'agent, et celui du CIG avant celui du CAG, sur cette base, une stratégie de l'ensemble doit être appliquée.
- Et la tâche qui est la subdivision du rôle en séquences de tâches (en séquentiel ou en parallèle), c'est l'élément atomique par rapport aux SMA conceptuel, au moment se son oeuvre. A l'implémentation, une tâche peut représenter un script s'exécutant sur un ou de plusieurs processus au sens informatique (Thread ou Processus Unix)

B. Le modèle d'interaction: Basé sur le principe d'agent facilitateur et inspiré du Contract-Net, il décide du champ de manoeuvre des agents et fait intervenir tous les éléments du modèle d'organisation.

Nous rappelons que cette contribution est en cours d'intégration dans le projet européen de recherche PROTEUS dédié à la réalisation d'une plate-forme de e-maintenance. Elle va nous permettre, pour la partie gestion coopérative de la pièce de rechange, d'implémenter le middleware SOAP au dessus du protocole HTTP entre les agents du système. exigence qui nous a permis de passer vers la technologie XML/SOAP. Elle a consisté à:

- Associer nos scripts d'agents aux plate-formes logiciels supportant cette technologie. Deux plate-formes ont été étudiées J2EE de Sun et .NET de MicroSoft avec les deux langages intimement liés le Java et le C#.
- Étudier les différents middlewares en optant pour le SOAP pour les avantages cités auparavant dans le chapitre 5 dont l'interopérabilité et son infiltration sur les ports du Web.
- Opter pour les Web-Services comme architecture sur le web car elles se basent actuellement sur le SOAP/XML.

En perspective de ce travail, un point essentiel est à prendre en considération:

- Une étude plus exhaustive pourra être faite sur une modélisation SMA, avec une association avec les Ants-Systems, pour la recherche optimale, en disponibilité et en coût, d'un ou de plusieurs articles sur le Web, dont le nombre de sites n'est pas connu d'avance, et où le problème n'est pas déterministe.

Bibliographie

[1] Agha G., Actor : *A Model of Concurrent Computation in Distributed Systems*, MIT Press, 1986.

[2] Matthieu Amiguet, *MOCA : Un modèle componentiel dynamique pour les systèmes multi-agents* Thèse de doctorat, Université de Neuchâtel, Suisse, 2003.

[3] Austin, J.L. *How To Do Things With Words.* Oxford University Press, 1962.

[4] Aronis K., Magou I., Dekker R., Tagara G., *Inventory control of spare parts using a Bayesian approach :A case study*, Europeen Journal of Operational Research, No 154, PP. 730-739, 2003.

[5] Attoui Ammar, *Les Systèmes Multi-agents et le Temps réel*, Eyrolles, Paris 1997.

[6] A.Barak, A. Shiloh, *A Distributed load-balancing Policy for a Multicomputer; Software Practice and Experience*, Vol.15, pp 901-913 Sep 85.

[7] Benaouda A., *Conception et implémentation d'un langage Orienté Acteur OCCAM+*, Mémoire de Magister, UFAS, 1999.

[8] Benaouda A., Zerhouni N, Barakat O., *Démarche pour modéliser et implémenter la e-maintenance des systèmes de production par l'approche système multi-agent*, JTEA02, Sousse, Tunisie, Vol. 2, PP.269–276, 2002.

[9] Benaouda A., Zerhouni N, Mostefai M., Barakat O., *Une modélisation Fonctionnelle des systèmes de production distribués par une approche coopérative des SMA (étude de cas)*, CGE'03, Alger, Algérie, 2004.

[10] Benaouda A., Zerhouni N, Mostefai M., Barakat O., *Les Cas d'Exception dans le Contract Net Agents : Etude Critique avec implémentation de Cas*, CIP'03, Alger, Algérie, 2004.

[11] Benaouda A., Zerhouni N, C. Varnier, *Spare part management for e-maintenance platform*, IEEE:MECHROB'2004, Session PROTEUS, Aachen, Germany, 2004.

[12] Benaouda A., N.Zerhouni, M.Mostefai, *A Preventive Intercation model applied to the management of spare parts.*, SEPADS, Salznurg, AUSTRA, February 13-15, 2005

[13] Benaouda A., N.Zerhouni, M.Mostefai, *A Preventive Interaction model applied to the management of spare parts.*, in Journal Transactions on Systems, World Scientific and Engineering Academy and Society, ISSN 1109-2777, Issue 1, Vol 4, pp 32-38, January, 2005

[14] Benaouda A. , N. Zerhouni, M. Mostefai, *A preventive interaction model in a cooperative environment: application to a platforme of e-maintenance*, International

Conference on advances in intelligent systems- theory and applications, in cooperation with IEEE Computer Society, 12-18 November, 2004, Luxemburg.

[15] Benaouda A., Benaouda N., *Simulation des architectures parallèles sur réseaux de PCs: Algorithmes, implémentation et Performances*, CIGE'04, Sétif, 2004.

[16] Benaouda N., Guyennet Y. , Benaouda A., *Simulation d'une plateforme de e-maintenance preventive utilisant les Services Web .Net*, SETIT, IEEE, Tunis, 28-30 Mars, 2005

[17] Olivier Boissier, *Systèmes Multi-Agents*, Cours SMA-DEA-CCSA, SMA/ENS Mines Saint-Etienne, 2001

[18] Bournez C., *Une architecture Multi-agents réflexive pour le contrôle de système de production distribués et hétérogène*, Thèse de doctorat, INSA, Lyon, France, 2001.

[19] Makram BOUZID, *Contribution à la modélisation de del'interaction agent/environnement: Modélisation stochastique et simulation parallèle* Thèse de doctorat, Université Henri Poincaré, Nancy 1, 2001.

[20] R.Brayant, R.Finkel, *A stable Distributed Scheduling Algorithm*, proc. Of2nd Int. Conf. On Distributed and computer systems, pp 314-323, Apr 81.

[21] J.P. Briot, Y. Demazeau (eds), *Principes et architectures des systèmes multi-agents*, Hermes Science, Lavoisier, 2001.

[22] Burns Alan, *Programming in OCCAM2*, Addison-Wesley Publishing Company,1988.

[23] Brafman R.I., M. Tennenholtz. *Modelling agents as qualitative decision makers*, Artificial Intelligence 94 (1-2), 1997.

[24] C. Castelfranchi, *Engineering Social Order. Int. Workshop Engineering Societies in the Agents World*,MIT Press, 1987. , ESAW 2000, LNAI 1972, p. 1-18, Springer-Verlag, 2000.

[25] Chelbi A., Ait-Kadi D., *Spare provisioning strategy for preventively replaced systems subjected to random failure*, International Journal production economis, No 74, PP. 183-189, 2001.

[26] K. Decker, K. Sycara, M. Williamson *Middle-Agents for the Internet.*, In IJCAI'97 International Joint Conference on Artificial Intelligence, Nagoya, Japan, 1997.

[27] D.J. Farber & al., *The distributed computing system*, in Proc. Compcon Spring 73, pp 31-34, 1973.

[28] Ferber J., *Les Systèmes Multi-agents: Vers une intelligence collective*, InterEditions, Paris, 1995.

[29] Ferber J., Gutknecht A., *A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems.* , International Conference on Multi-Agents Systems (ICMAS). IEEE CS Press, June 1998. ICMAS'98 .

[30] Ferber J. et M. Ghallab, *Problématique des univers multi-agents intelligents*, Actes des journées nationales PRC-GRECO Intelligence Artificielle, pages 295-320, mars 1988.

[31] Ferber J., *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*, Addison-Wesley, 1999.

[32] FIPA 97, *Foundation for Intelligent Physical Agents. FIPA 97 Specification. Part 2, Agent Communication Language*, http://www.fipa.org, 1997.

[33] Flora Adina Magda, *Agents et Systèmes Multi-agents*, Université "Polytehnica" de Bucarest, 2001

[34] Franklin S., A. Gasser, *Is it an agent, or just a program?: A taxonomy for autonomous agents*, Dans Muller,

[35] Frécon Louis, *Logique Modale et Communication*, Séminaire LEACM/INRETS, Lyon, 17 Mars1997.

[36] Gelbers L., Vannieuwenhuyse B., Waeyenberg G., Pintelon L., *La gestion de pièces de rechange et la sous-traitance: Quelques directives pratiques, Revue française de gestion industrielle*, Vol.22 No 3, PP. 107–115, 2003.

[37] H. Guyennet, E. Sanchis, F. Spies, *Network software for transputer distributed machines; 10th LASTED International Conference*, Actes publiés par ACTA, Press, pp 57-60, Innsbruck Autriche, Fév 92.

[38] T. R. GRUBER, K. Sycara, M. Williamson *A Translation Approach to Portable Ontology Specifications*, Knowledge Acquisition, 5 (2), pp. 199-200, 1993.

[39] Hewitt C., *Viewing Control Structures as Patterns of Message Passing*, MIT Press : 1975.

[40] Houck C, Agha G., *HAL: A High Level Actor Language and its Distributed implementation*, Dept of Computer Science, University of Illinois at Urbana Champaign, 1992.

[41] Houck C, *Run Time System Supported for Distributed Actor programs*, Thèse Master of science Computer University of Illinois at Urbana Champaign, 1992

[42] Huiskonen., *Maintenance spare parts logistics: Special characteristics and strategic choices*, International Journal of production economics, No 71, PP. 125–133, 2001.

[43] K. Hwang et al., *A Unix-Based Local Computer Network with Load Balancing*, IEEE, Apr 82.

[44] Huget Marc-Philippe , *une ingénierie de protocoles d'interaction pour les Systèmes Multi-agents* Thèse de doctorat, Université Paris IX, 2001.

[45] Chaïb-draa B., *Interaction between agents in routines, familiar and unfamiliar situations.*

[46] Chaib-Draa B., *Interaction between agents in routines, familiar and unfamiliar situations*, International Journal of intelligent and Cooperative informations System, 5(1), 1996

[47] Chevrier V., *Etude et mise en oeuvre du paradigme multi-agents : De ATOME à GTMAS*, Thèse de doctorat, Université Henri Poincaré, Nancy I, 1993.

[48] Cohen P.R, H.J. Levesque. *Teamwork*, Nous, vol. 25, 1991.

[49] Cohen P.R. et H.J. Levesque, *Communicative actions for artificial agents. In Proceedings of the International Conference on multi-agents Systems*, AAAI Press, Juin 1995.

[50] Cohen P.R. et H.J. Levesque, *Intention is choise with commitment*, Artificial Intelligence, No 42, pages 213-261, 1990.

[51] Dennett D.C. *The Intentional Stance*. The MIT Press, 1987.

[52] Drogoul A., *De la simulation multi-agents à la résolution collective de problèmes, Une étude de l'émergence d'organisation dans les systèmes multi-agents*, Thèse de doctorat de l'université de Paris VI, 1993.

[53] Drogoul A. et D. Fresneau, *Métaphore du fourragement et modèle d'exploitation collective de l'espace sans communication ni interaction pour des colonies de robots autonomes mobiles*, Actes des 6èmes Journées Francophones d'Intelligence Artificielle et Systèmes multi-agents, pages 99-114, novembre 1998.

[54] Durfee E. H., V. R. Gasser et D. D. Korkill, *Coherent cooperation among communicating problem solvers*, IEEE Transaction on Computers C-36, pages 1275-1291, 1987.

[55] Georgeff M.P., *A theory of action for multi-agent planning*, Dans Actes de AAAI-84, Austin, TX, 1984, 125-129. Hintikka, J. Knowledge and Belief. Cornell University Press, 1962.

[56] M. Huhns, M. Singh (eds.), *Readings in Agents*.

[57] D. Kayser, *La représentation des connaissances*, Hermes, Paris, 1997.

[58] Morgan Kaufmann, 1998. M. Wooldrige, *Reasoning about Rational Agents*, The MIT Press, 2000.

[59] Klein M., *Supporting conflict resolution in coopérative design systems*, IEEE Trans. Syst. man. Cybern. 21 (6), 1991, 1379-1390.

[60] Konolige K. A, *Deductive Model of Belief*, Pitman Publishing, 1986.

[61] Labrou Y. et T. Finin, *A Proposal for a new KQML Specification*, Technical Report CS-97-03, 1997.

[62] Labrou Y. et T. Finin, *Semantics for an Agent Communication Language. Agent Theories, Architectures and Languages IV. M.*

[63] Labrou Y. et T. Finin, *A semantics approach for KQML-a general purpose communication language for software agents*, 3rd International Conference on Information and Knowledge Management, Gaithersburg, MD, 1994.

[64] F. Lin, R. Keller, *Gradient Model: A demand-Driven Load Balancing Scheme*, IEEE, pp 329-336, 86.

[65] Maître B. et L. Lâsri, *Cooperating expert problem-solving in blackboard systems: ATOME case study*, In Proc. 1st MAAMAW, pages 251-263, U.K., 1989.

[66] Massotte P., Coutirier P. et Liu Y,

[67] Moore R.C., *A formal theory of knowledge and action. Dans Formal Theories of the Commonsense World*, ed. J.R. Hobbs si R.C. Moore, Ablex, 1985.

[68] T. W. Malone, K. Crowston *The interdisciplinary study of Coordination. ACM Computing Surveys*, Knowledge Acquisition, 5 (2), pp. 199-200, 1993.

[69] Mazouzi, *Ingénierie des protocoles d'interaction: Des systèmes distribués aux systèmes multi-agents*, Thèse de doctorat, Université Paris IX, 2001.

[70] Hervé Matthieu , *Modélisation conjointe de l'infrastructure et des processus pour l'administration pro-active de l'entreprise distribuée* Thèse de doctorat, Laboratoire de Productique et d'Informatique des Systèmes Manifacturiers, France, 2004.

[71] Masini G., Napoli A., Colnet D., Leonard D & Tombre K. *Les langages à Objets*, InterEditions 1989.

[72] Meyer Bertrand *Systematic Concurrent Object-Oriented Programming* Communications of the ACM, September, Vol. 36, No 9, 1993.

[73] M. Minsky, La société de l'esprit, InterEditions, Paris, 1988

[74] A.Barak, A. Shiloh, *A Distributed load-balancing Policy for a Multicomputer; Software Practice and Experience*, Vol.15, pp 901-913 Sep 85.

[75] Carle Patrice, *Un langage d'acteurs pour l'intelligence artificielle distribuée intégrant objets et agents par réflexivité compilatoire*, Thèse de doctorat, Paris VI, Avril 1992.

[76] H. Clark, B. McMillin, *DAWGS-A Distributed Compute Server Utilizing Idle Workstations*, Journal of Parallel and Distributed Computing, Vol 14,No 2, pp 175-186, Fév 92.

[77] RM di Scala, *www.rmdiscala.developpez.com*, univetsité Tours, France

[78] www.soap.user.com, *www.soap.user.com*

[79] Ramamritham, J.A. Stancovic, W. Zhao, *Distributed Scheduling f Tasks with Deadlines and Resource Requirements*, IEEE Transactions on Computers, Vol. 38, No 8. pp 1110-1123, Aug 89.

[80] Rapport de synthèse, *Systèmes de Production Sûrs de Fonctionnement (SPSF), Années 96,97,98 et 99 en France*, CNRS, France.

[81] Rao A.S., M.P. Georgeff, *Modeling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall (eds)*, Dans actes de Knowledge Représentation and Reasoning'91, Morgan Kaufman, 1991. p.473-484.

[82] Russell S.J., *Rationality and intelligence*, Artificial Intelligence, Vol. 94, 1997. p.57-77.

[83] Searle J.R, textit Speech Acts, Cambridge University Press, 1969.

[84] Shoham Y., *Agent-oriented programming*, Artificial Intelligence, Vol. 60, 1993. p.51-92.

[85] Singh M.P., *Semantical considerations on some primitives for agent specification*, M. Wooldridge, JP. Müller, and M. Tambe, editors, Intelligent Agents Volume II-Proceedings of the Workshop on Agent Theories, Architectures, and Languages (ATAL-95), Lecture Notes in Artificial Intelligence. Springer-Verlag, 1996.

[86] Singh M.P., *Towards a formal theory of communication for multi-agents systems*, In Proceeding of the IJCAI'91, 1991.

[87] R.G. Smith, *The Contract Net Protocol: High-Level Communication and Control in a distributed Problem Solver*, IEEE Transactions on Computers, Vol. C-29, No 12, pp 1104-1113, Dec 80.

[88] J.A.Stancovic, I.S. Sidhu, *An Adaptive Bidding Algorithm For Processes, Clusters and Distributed Groups*, Proc. 4th Int. Conf. Distributed Computers, pp 49-59, may 84.

[89] J. Stancovic, *An Application of Bayesian Decision Theory to Decentralized Control of Job Scheduling*, IEEE Transactions on Computers, Vol. C-34, No 2, Fev 85.

[90] C. Sibertin-Blanc *Pour une typologie des espaces d'interaction dans les SMA*, IRIT / Université Toulouse1, ...

[91] Vanderveken D., *The Logic of Speech acts*, Cambridge University Press, 1994.

[92] Weiss G. (ed.), *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, 2000.

[93] Wooldridge M. et N. R. Jennings, *Agent theories, architectures, and languages*, Dans Wooldridge, Jennings (ed), Intelligent Agents, Springer Verlag, 1995. p.1-22.

[94] Wooldridge, J. P. Muller and M. Tambe, Springer-Verlag, 1998.

[95] Wooldridge, Jennings (eds), *Intelligent Agents III. Agent Theories, Architectures, and Languages*, Springer Verlag, 1997. p.21-35.

[96] M. Wooldridge, *An Introduction to Multi-agent Systems*, John Wiley and Sons, 2002.

[97] M. Wooldridge and N.R. Jennings, *Intelligent agents: Theory and practice.*, The Knowledge Engineering Review, 10 (2): 115-152, 1995.

[98] Yonezawa A., Shibayama E., Briot JP., Honda Y et Takakda T. *Object Oriented Concurrent Processing Model ABCM/1 its and its Description Language ABCL/1 : An Object Oriented Concurrent Language*, MIT (The Massachusetts Institute of Technologie) Press, 1990.

[99] Yonezawa.A et Tokoro.., *Object-Oriented Concurrent Programming*, MIT Press, 1987.

[100] F. Zambonelli, N. R. Jennings, and M. Wooldridge, *Organizational Abstractions for the Analysis and Design of Multi-Agent Systems.*, In P. Ciancarini and M. Wooldridge editors, Agent-Oriented Software Engineering. Springer-Verlag Lecture Notes in AI Volume 1957, January 2001.

[101] Zlotkin G., J.S. Rosenschein, *Negociation and task sharing among autonomous agents in coopérative domains*, Dans Actes du 11eme IJCAI, Detroit, USA, 1989.

Résumé

Dans cette thèse, nous proposons un modèle d'interaction dédié Système Multi-Agent (SMA) qui répond à un modèle d'organisation basé sur les principes AGIRAT (Agent, Groupe, Interaction, Autorité et Tâche) et s'appliquant, mais dans un cadre coopératif, à la gestion de ressources matérielles d'une façon générale et, comme cas d'application, à la pièce de rechange sur une plateforme de e-maintenance. En effet, nous avons établi une architecture SMA interne au site qui prévoit, entre autres, un agent cognitif dédié, dont le rôle est de guetter, et même de prévenir, les cas de sous-stockage et de sur-stockage des articles qui peuvent arriver, et ce, sur la base de la consommation antérieure. L'architecture SMA globale de tous les sites en interaction avec le SMA interne du site, dans ce même cadre coopératif, assure l'émergence d'un équilibrage de charge en matière des ressources, que nous avons appelé " équilibrage de ressources ", et permet de quantifier la disponibilité d'articles. Cette solution évite à l'ensemble des sites des situations de dépassement de seuil de stockage. La mise en oeuvre de l'architecture que nous proposons requiert l'utilisation d'un langage de communication dépendant d'un middleware dédié SOAP/XML pouvant être un moyen de supporter cette solution comme une architecture Web-Service dans une e-plateforme.

Absctract

In this thesis, we propose a model of interaction corresponding to an organization model, and based on the AGIRAT principles (Agent, Group, Interaction, Authority and Task) and applying to the management of the spare part on a platform of e-maintenance, in a cooperative context. Indeed, an architecture internal SMA to the site foresees a dedicated cognitive agent, whose role is to watch, and even to warn, the cases of under-storage and on-storage of the articles that can arrive, and that, on the basis of the previous consumption. The architecture global SMA of all sites assures, in interaction with the internal SMA of the site, in this same cooperative setting, the emergence of a load balancing of measured availability of articles, capable to avoid to the set of the situations of overtaking of storage doorstep. The implementation of architecture that we propose requires the use of a language of communication depending on a middleware dedicated SOAP/XML which can be a means of supporting this solution like an architecture Web-Service in an e-platform.