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

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option : Système d'Information.

Sujet :

ADAPTATION ET MISE EN ŒUVRE D'UN PROTOCOLE AUTO-ORGANISABLE DANS UN SYSTEME DE PRODUCTION

Présenté par : Mr. RAÏAH Ahmed **Promoteur :** Mme. MELLAH Hakima
Mr. TIGUEMOUNINE Arezki

Organisme d'accueil : C . E . R . I . S . T.

Soutenue le: 27 octobre 2007, devant le jury composé de :

Mme BENSTITI .S, Chargé de cours, USDB	**Président**
Mlle BOUSTIA .N, MA, USDB	**Examinateur**
Mme ARKAM .M, Vacataire, USDB	**Examinateur**

-2006/2007-

Remerciement

Nous remercions Dieu le tout puissant qui nous a donné la volonté, le courage et la patience sans lesquels notre travail n'aurait pas abouti.

Nous remercions nos chèrs parents pour leur soutien fidèle et sans faille.

Nous remercions chaleureusement notre promotrice Mme Mellah Hakima pour ces précieuses orientations et ses conseils judicieux.

Nous remercions les membres du jury d'avoir accepté d'évaluer la qualité et la valeur de notre travail.

Nous remercions aussi Melle Ourari Soumia, et Melle Halilali Imen et Mr Kouider Ahmed, pour l'aide précieuse qu'ils nous ont apporté.

Nous remercions tout le personnel du département de l'informatique ainsi que tous ceux et celles qui de prés ou d'où loin ont contribué à la réalisation de ce modeste travail.

Ahmed et Arezki

Dédicace

Je dédie ce travail à tous ceux que j'aime

A mon père et ma mère, à qui je dois tout

A mon frère Mezian

A mes soeurs Lila et Dhahbia

A toute ma famille,

A tous mes amis en particulier Chahra, Imen

elhachemi, Reda, Lounes, Amine, Cherif, Mhamed,

Taha, Karim, Mustapha, Mounir, Sofiane, Nazim,

Farid, Yousef,

Ahmed

Dédicace

A mes chères mère et père, sœurs Naziha

Chahira Sonia et grands-mères, je dédie

ce travail.

A toute la famille TIGUEMOUNINE et HASSENAOUI.

A tous mes amis : Ahmed, Kamel, Youcef,

Elhachemi, Karim, Mustapha, Hichem, Nassim, Fethi,

Thabet, Taha, Amine, Louness, Reda, Mounir.

Arezki

Résumé

Dans ce document, nous voulons mettre en épigraphe un protocole auto-organisable dans un environnement industriel simulé par un Système Multi-agents et présentant un ensemble de machines auxquelles des tâches sont attribuées. Dans un système de production, l'affectation des tâches aux machines génère un flux de circulation de l'information. Le flux ainsi généré peut être interrompu lors de l'affectation d'une tâche à une machine qui est en état de dysfonctionnements. De plus, le réseau de communication tâche-machine ou bien machine-machine peut présenter une panne quelconque et le résultat est que la tâche n'aboutira pas à sa destination. Le recours au protocole sus-cité devra faire émerger un nouveau flux permettant la tolérance aux pannes, ainsi qu'une prévention contre le blocage du système en entier en assurant une communication permanente reflétant ainsi la robustesse et la flexibilité du système.

***Mots-Clés** : système multi-agents, système de production, protocole auto-organisable, tolérance aux pannes.*

Abstract

In this document, we want to put in evidence a self-organizing protocol in an industrial environment simulated by a Multi-Agents System and presenting a whole of machines to which of tasks are assigned. In a production system, the affectation of tasks to machines generates an information flux circulation. The generated flux can be interrupted at the time of task affectation to a machine that is in a dysfunction state. Besides, the network of communication task-machine or machine-machine can present any breakdown and the result is that the task won't arrive to its destination. The recourse to the know-quoted protocol should have a new flux permitting tolerance to breakdowns emerged, as well as a prevention against the blockage of the system in whole while assuring a permanent communication reflecting so the robustness and the flexibility of the system.

***Key-words:** multi-agents system, production system, self-organizing protocol, breakdown tolerance.*

SOMMAIRE

Liste des Figures

Introduction générale et problématique

INTRODUCTION GENERALE

Actuellement, les entreprises doivent se placer dans une démarche constante d'évolution pour rester compétitives, afin de répondre aux attentes et aux besoins des clients. Ainsi, les systèmes de production imposent des contraintes de fiabilité très sévères et le moindre dysfonctionnement du système peut affecter le processus de fabrication. La maîtrise et la gestion des ressources du système de production sont incontournables pour supporter un processus de production performant.

Dans le pilotage temps réel de systèmes de production (SdP), le problème, critique et important à résoudre, est celui de la tolérance aux pannes. Le système de pilotage doit fournir des mécanismes de tolérance aux pannes très performants afin d'assurer le fonctionnement continu du SdP : détection, prévention et correction, etc.

Les systèmes multi-agents offrent un cadre de modélisation et de simulation des systèmes de production en proposant de représenter directement leurs éléments, leurs comportements et leurs interactions sous la forme d'entités informatiques disposant de leur propre autonomie. De plus une modélisation par un système multi-agents permet à un SdP d'avoir des aspects de robustesse, proactivité, réactivité et de flexibilité, ce qui permet au système de pilotage d'un SdP d'être performant et réagissant à tous les aléas pouvant survenir.

De nombreux travaux ont été effectués, proposant des approches multi-agents utilisant des protocoles d'interaction connus (ContactNet, TwinBase, Réseau d'Accointances, etc....), la plupart de ces approches présentent un aspect centralisé de la modélisation des systèmes de production, ce qui résout le problème d'affectation des tâches aux ressources.

Ce que nous pouvons reprocher à ces protocoles c'est qu'ils ne sont pas tolérants aux pannes, et toute production de dysfonctionnement risque de coûter cher à l'entreprise, sinon un arrêt total du fonctionnement du SdP.

Lors de la communication inter-agents, un problème peut surgir soit au niveau de l'agent soit au niveau du lien de la communication ; afin de remédier à cette situation, nous avons fait recours à un protocole auto organisable offrant la possibilité au SMA de changer sa structure de communication (organisation) et ainsi se réorganiser de lui-même sans aucune intervention externe.

Pour cela nous allons expliquer le fonctionnement d'un SdP, ceci d'une manière générale. Donner notre approche multi-agents, que nous jugeons la plus adéquate pour la modélisation d'un système de production. Expliquer comment se fait la communication inter-agents en appliquant le protocole auto organisable que nous définissons dans le premier chapitre. Le protocole que nous utiliserons permettra à notre approche multi-agents proposée de changer son organisation dés qu'un problème surgit, ceci lui donnera la possibilité d'accomplir pleinement les fonctions du pilotage (ordonnancement dynamique, exécution et surveillance).

PARTIE 1

L'auto-organisation dans les systèmes multi-agents et les systèmes de production

CHAPITRE I

L'auto-organisation dans les systèmes multi-agents

1. Introduction

L'intelligence artificielle (IA) est apparue en 1956, elle a été largement développée aux Etats-Unis, puis au milieu des années 70 elle apparut en Europe et en Asie. C'est la science dont le but est de faire faire, par une machine, des tâches que l'homme accomplit en utilisant son intelligence. L'intelligence artificielle distribuée (IAD), est une branche de l'IA qui consiste en la résolution distribuée des problèmes, s'intéresse à la manière de diviser un problème en un ensemble d'entités distribuées et coopérantes et à la manière de partager la connaissance du problème afin d'en obtenir la solution.

L'origine d'agent provient d'une part de l'intelligence artificielle distribuée (IAD), et d'autre part de la vie artificielle. Les premières applications de la vie artificielle sont apparues quasiment en même temps que l'informatique, avec Von Neumann et ses automates cellulaires, ou encore Mc Culloch et ses neurones formels. [Mat 06]

Ce sont donc à partir de ces premières réalisations que nous avons vu apparaître l'idée de système multi agents (SMA).

Les recherches menées actuellement dans le domaine multi-agent s'intéressent particulièrement aux systèmes dotés d'une très forte dynamique et constitués de nombreux agents. Comme les créations, suppressions et modifications d'agents y sont très fréquentes, ainsi que la pression exercée par l'environnement dans lequel le système est immergé, il devient très difficile voire impossible de prévoir toutes les situations pouvant survenir durant le fonctionnement du système, et ceci dès la conception. Or chaque agent n'a qu'une vue partielle, ou peut être erroné, du système dans lequel il est situé ; ce qui explique que des situations imprévues se produisent fréquemment. L'imprévu est donc inhérent à la vie du système et l'auto-organisation devient un moyen pour parvenir à surmonter les perturbations de manière autonome. Par autonomie nous sous-entendons, autonomie dans son comportement, dans sa prise de décision sans aucune intervention externe. Cela intègre le point de vue de Conte et Castelfranchi, qui définissent l'autonomie comme

étant "la capacité d'un agent à générer et poursuivre ses propre buts". [Con 95] [Cam 98]

Nous présenterons, dans ce chapitre, quelques concepts et définitions d'agent, de systèmes multi-agent, ensuite nous aborderons l'organisation et l'auto-organisation, pour, enfin, expliquer le protocole auto-organisable que nous détaillerons par la suite.

2. Définition d'agent

Le concept d'agent a été l'objet d'étude depuis plusieurs décennies dans différentes disciplines. Il a été utilisé non seulement dans des systèmes à base de connaissance, la robotique, le langage naturel, mais aussi dans des disciplines comme la philosophie et la psychologie.

On trouve une multitude de définition d'agents. Elles se rassemblent toutes, mais différent selon le type d'application pour lequel est conçu l'agent. S. RUSSEL et P. NORVIG [Rus 95], M. WOOLRIDGE et N.R. JENNINGS [Woo 95], ont tous donné de différentes définitions d'un agent ; nous retenons celle de J. FERBER « Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi agent, peut communiquer avec d'autre agent, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents. ». [Fer 95]

2.1. Caractéristiques des agents

a) **Situé :** l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement (systèmes de contrôle de processus, systèmes embarqués,…).

b) **Autonome :** l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres action ; ainsi que son état interne.

c) **Flexible :** l'agent dans ce cas est :

- ***capable de répondre à temps :*** l'agent doit être capable de percevoir son environnement et d'élaborer une réponse dans les temps requis.

- ***proactif :*** l'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au bon moment.
- ***social :*** l'agent doit être capable d'interagir avec les autres agents (logiciels et humains).

2.2. Types des agents

Selon ce que les agents peuvent, ou non: [Uni 05]

- Communiquer directement avec d'autres agents
- Agir dans un environnement
- Percevoir (éventuellement de manière limitée) son environnement
- Se reproduire
- Posséder des compétences et offrir des services

a) **Agent situé vs communiquant**

1. Agent situé :

- L'environnement possède une métrique,
- Les agents sont situés à une position dans l'environnement qui détermine ce qu'ils perçoivent;
- Ils peuvent se déplacer;
- Il n'y a pas communications directes entre agents, elles se font via l'environnement

2. Agent communiquant :

- Il n'y a pas d'environnement au sens physique du terme,
- Les agents n'ont pas d'ancrage physique,
- Ils communiquent via des informations qui circulent entre les agents.

b) **Agents cognitifs vs réactifs**

1. Agents cognitifs :

- Représentation explicite de l'environnement et des autres agents
- Peut tenir compte de son passé et dispose d'un but explicite
- Mode "social" d'organisation (planification, engagement)
- Petit nombre d'agents (10/20), hétérogènes à gros grain

• Les relations entre agents s'établissent en fonction des collaborations nécessaires à la résolution du problème.

2. Agents réactifs :

- Pas de représentation explicite de l'environnement
- Pas de mémoire de son histoire, ni de but explicite
- Comportement de type stimulus réponse
- Mode "biologique "d'organisation
- Grand nombre d'agents (>100), homogènes à grain fin

• La structure du système émerge des comportements et non d'une volonté d'organisation.

La figure (figure 1.1) suivante nous montre le modèle d'un agent réactif. [Rus 95]

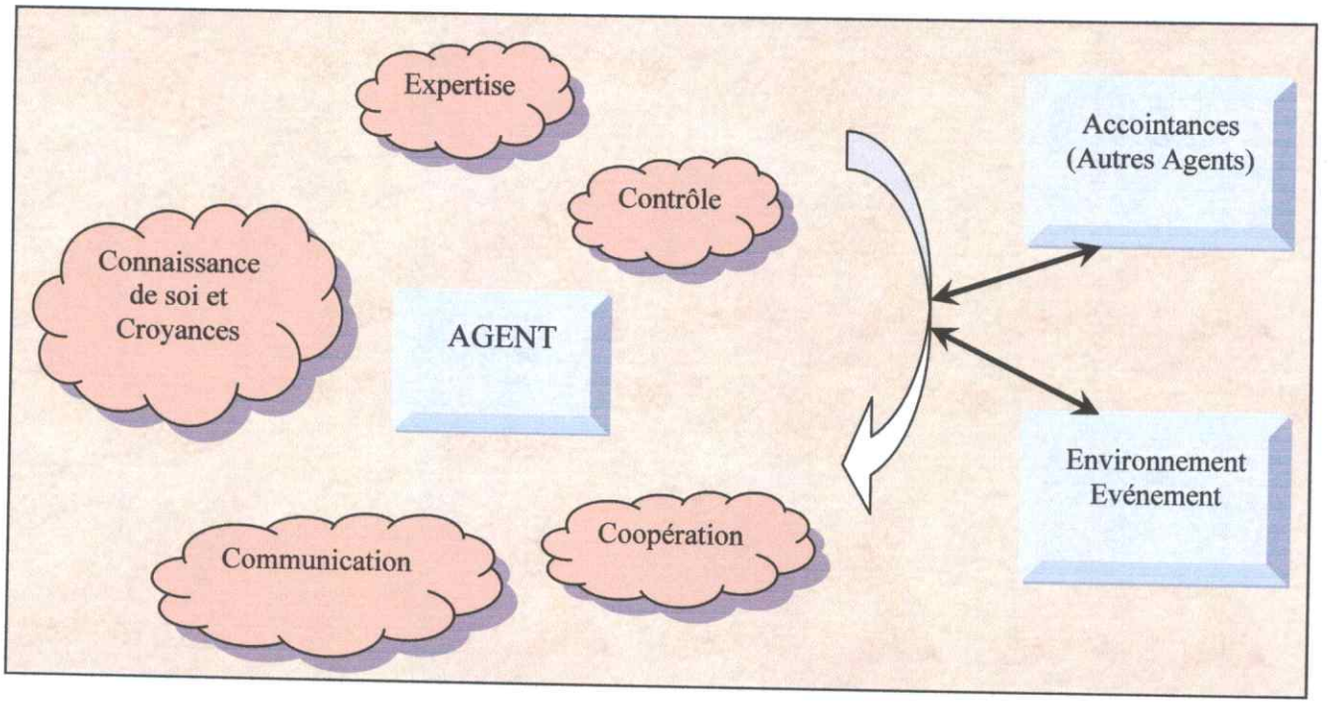


Figure 1.1. Modèle d'un agent réactif.

3. Système multi-agents

3.1. Présentation

Un système multi-agents (SMA) est un système distribué composé d'un ensemble d'agents. Contrairement aux systèmes de l'IA, qui simulent, dans une certaine mesure, les capacités du raisonnement humain, les SMA sont conçus idéalement comme un ensemble d'agents interagissant, le plus souvent, selon des modes de coopération, de concurrence ou de coexistence.

3.2. Définition

Un système multi agent est un système composé d'un grand nombre d'entités autonomes, nommées agent, qui évoluent dans un environnement commun, qui ont un comportement collectif qui permet d'atteindre une fonction désirée. [Wei 99]

3.3. Caractéristiques des SMA

Un SMA est généralement caractérisé par :

- Chaque agent a des informations ou des capacités de résolution de problèmes limitées (ainsi, chaque agent a un point de vue partiel).
- Il n'y a aucun contrôle global du système multi agent.
- Les donnés sont décentralisées.
- Le calcul est asynchrone.

Un SMA peut être :

- **Ouvert :** les agents y entrent et en sortent librement (ex : un café).
- **Fermé :** l'ensemble des agents reste le même (ex : un match de football)
- **Homogène :** tous les agents sont construits sur le même modèle (ex : une colonie de fourmis).
- **Hétérogène :** des agents de modèles différents, de granularités différentes (ex : l'organisation hospitalière).

3.4. Composition des SMA

Un système multi-agents est composé des éléments suivants [Vie 00] :

- **Un environnement E :** c'est l'espace où peuvent se déplacer les agents.
- **Un ensemble d'objets O :** Ces objets sont situés, c'est-à-dire que pour tout objet, il est possible, à un moment donné, d'associer une position dans E.
- **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. Cela

correspond à la capacité des agents de percevoir leur environnement, de manger, etc. Nous reviendrons sur ce point particulièrement important.

- **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.

4. L'environnement

L'environnement d'un agent désigne tout ce qui est extérieur à l'agent. On distingue l'environnement dit social, c'est-à-dire les agents qu'il connaît, et l'environnement, dit physique, constitué de ressources matérielles présentes dans le champ de perception de l'agent ou de ses propres effecteurs.

L'environnement peut être caractérisé en utilisant les termes fournis dans [Rus 95] que l'on retrouve aussi dans [Woo 00] et [Lin 01].

4.1. Environnement accessible

Dans un environnement accessible, le système peut obtenir une information complète, exacte et à jour sur l'état de son environnement. Dans un environnement inaccessible, seul une information partielle est disponible. Par exemple, un environnement tel que l'Internet n'est pas accessible car il est impossible de tout connaître à propos de lui. Un robot qui évolue dans un environnement possède des capteurs pour le percevoir et, en général, ces capteurs ne lui permettent pas de connaître tout de son environnement, qui est alors considéré comme inaccessible.

4.2. Environnement continu

Dans un environnement continu, le nombre d'actions et de perceptions possibles dans cet environnement est infini. Dans un environnement discret, le système possède des perceptions distinctes, clairement définies qui décrivent l'environnement. Par exemple, dans un environnement réel tel que l'Internet, le nombre d'actions qui peuvent être effectuées par les utilisateurs est illimité.

Mais, dans un environnement simulé tel qu'un écosystème, le nombre d'actions ou de perceptions qu'une entité virtuelle (comme une fourmi ou un robot) peut avoir est limité, l'environnement est alors discret.

4.3. Environnement déterministe

Dans un environnement déterministe, une action a un effet unique et certain. Si le système agit dans son environnement, il n'y a aucune incertitude sur l'effet de son action sur l'état de l'environnement. L'état suivant de l'environnement est complètement déterminé par l'état courant. Dans un environnement non déterministe, une action n'a pas un effet unique garanti. Par nature, le monde physique réel est un environnement non déterministe.

4.4. Environnement dynamique

L'état d'un environnement dynamique dépend des actions du système qui se trouve dans cet environnement mais aussi des actions d'autres processus. Aussi, les changements ne peuvent pas être prédits par le système. Un environnement statique ne peut changer sans que le système agisse. Par exemple, l'Internet est un environnement hautement dynamique, son changement n'est pas simplement lié aux actions d'un seul utilisateur (vu comme un système plongé dans cet environnement).

5. Communication

Un protocole de communication est à la base du comportement d'un système multi agent, son rôle est de fixer le contenu des messages échangés, et l'enchaînement de ces messages est géré par un protocole de conversation. Ce dernier s'appuie sur un protocole de communication, même si un protocole de communication n'est pas forcément lié à un protocole de conversation.

Un protocole de communication peut suivre différents standards, les plus connus sont : [Hal 06]

- **KQML (***Knowledge Query and Manipulation Language***).**
- **FIPA-ACL** (*Foundation for Intelligent Physical Agents*)- (*Agent Communication Language*).

5.1. Les protocoles d'interaction

M. Barbuceanu M. et Fox M.S. [Bar 95] ont défini l'interaction entre les agents : un des aspects d'interaction montre une conversation basée sur un échange partagé et conventionné de messages. Les conversations entre agents dans les SMA sont souvent structurées selon des schémas typiques appelés *protocoles d'interaction.*

Il existe différents types de protocoles d'interaction, on cite :

- Les protocoles de coordination,
- Les protocoles de coopération,
- Les protocoles de négociation, ainsi que
- Les mécanismes du commerce électronique.

Le problème d'allocation de tâches auquel fait face généralement un ensemble de résolveurs de problèmes, est géré par les protocoles de coopération. Des protocoles de coopération ont été élaboré pour pallier l'insuffisance (trous) que représente l'allocation de tâches, nous citons :

5.1.1. Contract Net

Dans ce protocole, les agents peuvent prendre deux rôles [Dav 83] : *gestionnaire* ou *contractant*. L'agent qui doit exécuter une tâche donnée (le *gestionnaire*), commence tout d'abord par décomposer cette tâche en plusieurs sous tâches. Il doit ensuite *annoncer* les différentes sous tâches au reste des agents de l'environnement. Les agents qui reçoivent une annonce de tâche à accomplir peuvent ensuite faire une proposition devant refléter leur capacité à remplir cette

tâche. Le gestionnaire rassemble ensuite toutes les propositions qu'il a reçues et alloue la tâche à l'agent ayant fait la meilleure proposition.

L'intérêt d'un tel protocole est qu'il réalise la coordination de tâches parmi les agents en assurant l'allocation la plus optimale possible.

5.1.2. Twin base

Un agent simple consultant le système est un agent intelligent plus quelques sources de connaissance du domaine dirigées par l'agent [Che 99]. Un petit nombre d'agents est organisé dans un système multi agents dans le but de la connaissance et l'information partagées. Dans ce genre des SMA, nous supposons que :

1. les agents sont reliés par un réseau fiable;
2. un agent individuel a sa propre base de connaissance ou un accès total à quelques systèmes locaux de connaissance, c.-à-d., systèmes experts, bases de connaissance, etc.... Par conséquent, il peut fournir des services experts dans quelques domaines d'application;
3. les agents sont disposés à coopérer l'un avec l'autre dans le but de partager la connaissance.

Dans le groupe coopératif, quand un agent reçoit une tâche qui dépasse ses capacités de résolution de problèmes, il invoque la recherche d'un but dans son système de connaissance local pour rapporter les données utilisateur voulues.

Si la tâche est au-delà de sa capacité de résolution, l'agent distribue cette tâche à d'autres agents. À cet égard, localiser efficacement un groupe coopératif devient une question critique, étant donné le coût entre le calcul et la communication. L'approche Twin base a été proposée pour permettre à un agent de diriger les activités d'autres agents pour un groupe d'agent directement localisé.

5.1.3. Réseau d'accointances

Le réseau d'accointance se présente sous la forme d'un tableau dynamique mémorisant les adresses des agents du réseau [Yam 01] ainsi que leurs

compétences. Il est possible d'ajouter ou de retirer des agents de ce tableau comme dans un simple carnet d'adresses. Un agent va ajouter un autre agent dans son carnet d'adresses lorsqu'il possède des choses en commun avec lui. On va, plus facilement, demander des services à des agents possédant des rôles similaires puisque ceux-ci possèdent des compétences proches des nôtres. L'intérêt des accointances est de minimiser la fonction énergie pour la recherche d'agents travaillant dans le même domaine.

6. Organisation et auto-organisation (AO)

6.1. Définition

Dans le sens commun du terme, l'organisation est à la fois la façon (la manière dont un ensemble est constitué en vue de son fonctionnement), et le fait (l'action d'organiser). Dans le premier sens on se réfère à un ordre, un régime, une structure ; dans le second sens, l'idée sous-jacente est celle d'un processus, d'une dynamique. Il y a deux aspects dans ce seul terme : l'action et le résultat de l'action. Les théories qui traitent de l'organisation sont issues de disciplines diverses telles que la psychologie sociale [Mor 77], l'économie [Min 79] et la sociologie [Sch 91]. Plus récemment les sciences cognitives [Ras 91] et les systèmes multi-agents se sont également intéressés à la notion d'organisation. A ce titre, on peut trouver des définitions variées, mais se rapprochant toutefois de celle du sens commun. Or nous pouvons nous accorder à voir qu'en S.M.A, l'organisation est un modèle permettant aux agents de coordonner leurs actions au cours de la résolution d'une ou de plusieurs tâches. Elle définit d'une part, une structure (ex., une hiérarchie) comprenant un ensemble de rôles qui doivent être attribués aux agents et un ensemble de chemins de communication entre ces rôles. Elle définit d'autre part un régime de contrôle (ex., une relation maître/esclave) qui dicte le comportement social des agents. Enfin, elle définit des processus de coordination qui déterminent la décomposition des tâches en sous-tâches, l'allocation des sous-tâches aux agents, et la réalisation des tâches dépendantes de façon cohérente. [Mal 99]

Le terme d'auto-organisation a initialement été défini par Farley et Clark du laboratoire Lincoln. Cem Ünsal cite cette définition dans sa thèse [Üns 93] : un système auto-organisateur est un système qui change sa structure de base en fonction de son expérience et de son environnement.
La figure suivante (figure 1.2) nous montre l'auto-organisation chez l'être humain. [Cam 98]

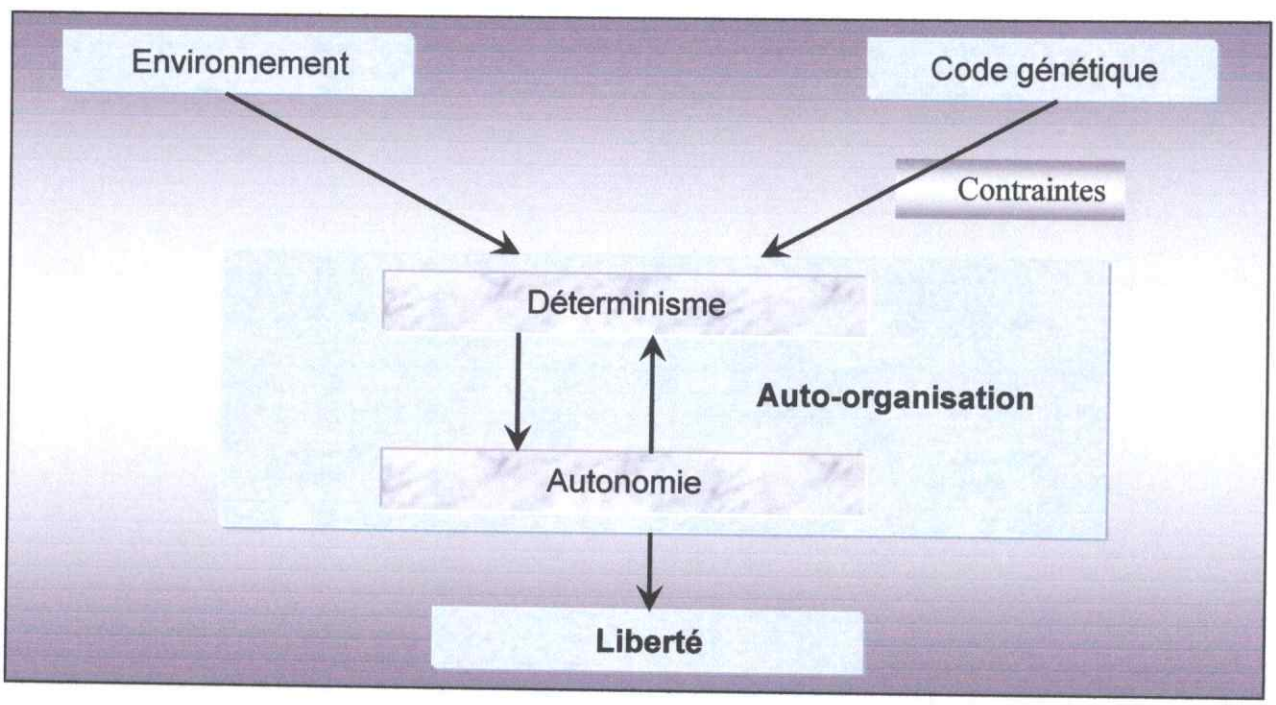


Figure 1.2. L'auto-organisation chez l'être humain.

Toru Ishida [Ish 92] donne une définition plus générale de l'auto-organisation ou OSD (*organisation self-design*). Elle permet à une organisation d'agents de s'adapter eux-mêmes aux situations dynamiquement changeantes.
Selon Young-pa So [So 93], les agents coopérant travaillent typiquement dans des environnements changeants. Il est donc crucial que les mécanismes de coordination qu'ils utilisent soient capables de s'adapter à ces changements. En particulier, l'organisation des agents doit évoluer dans le temps lorsque les circonstances le nécessitent. Dans le domaine de l'IAD, une telle réorganisation adaptative est appelée auto-organisation lorsqu'elle est réalisée par les membres de l'organisation.

Pour Pierre Glize et son équipe [Piq 96] [Cam 98], l'auto-organisation permet à un système de s'adapter de manière autonome aux conditions changeantes du milieu. Elle consiste plus précisément en une transformation de la topologie (c'est-à-dire des relations entre les agents) en tant que résultat du fonctionnement de ce même réseau dans le cadre de son couplage structurel avec l'environnement. Ils ajoutent que les systèmes auto-organisateurs appartiennent à la classe des systèmes autonomes (systèmes spécifiés par des mécanismes internes d'auto-organisation) et non pas hétéronomes (systèmes définis par des mécanismes extérieurs de contrôle). Cela implique que les règles d'organisation sont intérieures au système qui apparaît ainsi informationnellement clos. [Mal 99]

6.2. Caractéristiques des systèmes auto-organisés

Joël de Rosney [Ros 75] et Daniel Durand [Dur 79], proposent une double description pour un système auto-organisé : structurelle et fonctionnelle.

Au niveau structurel, ils dénombrent quatre éléments caractérisant un système :

- La frontière qui le sépare de son environnement, elle est plus au moins perméable. En informatique, elle peut désigner, par exemple, l'interface homme/machine.
- Les éléments qui peuvent être identifiés, dénombrés et classés. Ils sont plus au moins hétérogènes. En S.M.A, il s'agit des agents.
- Le réseau de relation, de transport et de communication qui véhicule soit des matières, soit de l'énergie, soit des informations. En S.M.A, ce réseau représente les liens de communication inter-agents.
- Les réservoirs dans lesquels sont stockés des matières, des produits, de l'énergie et l'information.

Au niveau fonctionnel, ils distinguent aussi quatre éléments :

- Des flux de nature diverses : de matières, de produits, d'énergie, d'information.... Ils circulent dans les divers réseaux et transitent dans les réservoirs du système. En informatique, il s'agit des signaux échangés entre entités.

- Des centres de décision qui reçoivent les informations et les transforment en actions. En S.M.A, ce sont en partie les compétences et les aptitudes des agents.
- Des boucles en rétroaction qui ont pour objet d'informer les déclencheurs de ce qui se passe en aval. Le mécanisme de rétroaction est en particulier utilisé pour mettre en œuvre les apprentissages par renforcement.
- Des délais de réponses qui permettent de procéder aux ajustements de temps nécessaires à la bonne marche du système. [Cam 98]

7. Le protocole auto-organisable

7.1. Introduction

Dans le chapitre 1, nous avons cité quelques protocoles d'interaction (*Contract Net, Twin base* et *réseau d'accointances*), et expliquer le mode de fonctionnement de ces protocoles, ainsi que les avantages et inconvénients de chacun. Dans le cadre de notre étude, nous allons adapter et mettre en œuvre un protocole dans le cas de gestion de la production, ce protocole doit présenter un aspect dynamique caractérisé essentiellement par un chamboulement fréquent de l'organisation d'une chaîne de production suite à des imprévues survenant à tout moment (panne, lien coupé,...). Malgré que les protocoles cités en dessus symbolisent une stratégie de haut niveau gouvernant les interactions entre les agents tout en permettant de faciliter leur dialogue, ils ne pallient pas les exigences imposées par notre environnement ; c'est pour cela que nous avons choisi un protocole auto-organisable pour mettre en œuvre le système.

7.2. Inspiration du protocole

Ce protocole a été inspiré biologiquement a partir d'une colonie de bactéries, Radhika Nagpal [Nag 00] a proposé un ensemble de primitives reflétant ce qui se passe réellement dans la vie d'une colonie de bactéries ; cette dernière est

caractérisée par un changement de pattern fréquent, et ce qui donne à cette colonie son aspect auto-organisable.

Dans [Hal 06] nous trouvons une interprétation informatique (évidemment dans le domaine des S.M.A) des concepts d'une colonie de bactéries. Ces primitives deviennent des processus constituants le protocole auto-organisable.

7.3. Les processus caractérisant le protocole

1) *Processus Calcul de Position*

Un agent informatique diffuse un message à ses voisins locaux, ce message contient un nom (signature du message), un ensemble de paramètres comme l'identité de l'agent émetteur source, le rôle de l'agent émetteur et une valeur Pos. L'agent qui reçoit le message, le diffuse dans son voisinage local avec la valeur incrémentée de « 1 », et ainsi de suite jusqu'à ce que le message soit propagé dans le système entier.

Chaque agent récepteur sauvegarde seulement la valeur minimale et l'identité de l'agent émetteur source. La valeur contenue dans le message augmente dés qu'on se déplace loin de la source originaire du message.

Donc cette valeur reflète la distance entre l'agent récepteur et l'agent émetteur (source d'envoi). L'agent doit émettre ce message pour maintenir sa connectivité avec les agents voisins, et adapter le message aux éventuels changements de topologies du réseau (lors des mises à jour des liens de communication entre les agents).

Ce processus fournit des informations sur le positionnement d'un agent parmi un ensemble d'agents. Les agents peuvent comparer leurs valeurs locales à un seuil pour déterminer s'ils sont dans une région relative à la source d'envoi ; ils peuvent aussi comparer les différents messages pour déterminer où ils sont positionnés par rapport à deux sources différentes.

2) *Processus Election*

Un algorithme d'élection fonctionne selon le fait que chaque agent peut initier un parcours parallèle du réseau, marqué avec sa valeur de fitness de l'initiateur. Lorsqu'un agent reçoit un message relatif à ce parcours, il le propage s'il n'a pas déjà

reçu une valeur de fitness plus importante et si sa propre valeur de fitness n'est pas plus grande. Dans le cas contraire, il stoppe ce parcours (ne le propage pas) et, si ce n'était pas déjà fait, lance son propre parcours.

Ce parcours peut de plus être acquitté. Ceci permet de prévenir son initiateur de la terminaison. Ce signal est appelé « signal inhibant ». Les agents sont inhibés seulement par les agents ayant la plus grande valeur de fitness.

Nous pouvons voir ce processus intéressant et robuste pour les comportements asynchrones (imprévus).Aussi, si l'agent leader détecte qu'un agent n'est plus capable d'assurer son rôle parce qu'il est isolé (tous ses liens de communication sont perdus) ou parce qu'il est en panne, alors l'agent leader a le privilège de le détruire.

Ce processus donne la capacité de s'adapter automatiquement si l'agent leader meurt ou si le message inhibant disparaît du réseau. Ainsi, la compétition peut reprendre.

3) *Processus Groupement*

Ce processus est identique à celui de l'Election, mais les agents envoient des messages d'inhibition aux agents de même rôle que les leurs. Donc un agent peut inhiber spatialement des agents, mais globalement la compétition continue jusqu'à ce que tous les agents soient inhibés ou sélectionnés.

4) *Processus Calcul du Chemin*

Lorsqu'un agent reçoit le même message plusieurs fois, il consulte les différentes valeurs du message et les compare. Nous avons déjà supposé que la valeur contenue dans le message augmente en se déplaçant loin de la source d'envoi, et ce pour déterminer la position de l'agent par rapport à cette source.

La comparaison faite sur la valeur et l'identité de l'agent émetteur permet de choisir le meilleur voisin, en choisissant la valeur minimale parmi les autres valeurs reçues du même message.

Ce processus est utilisé pour créer un chemin qui assure la circulation d'un message, de l'agent émetteur vers l'agent récepteur.

Contrairement au processus précédent qui était utile pour les déductions géométriques, ce processus est important pour la direction et la connectivité. Aussi

ce processus était une clé dans le travail de Coore [Woo 95] sur les modèles topologiques autos organisés, et depuis, a été exploité largement dans les systèmes informatiques.

5) *Processus Signal de Vie*

Par le biais d'un programme agent, il est possible de détecter l'absence d'un agent à travers une écoute locale. Chaque agent diffuse périodiquement un message de vie à ses voisins. Dans le cas d'un voisin manquant, l'agent change de comportement (il fait appel à un processus selon le cas).

La fréquence de messages de vie des agents voisins détermine la vitesse avec laquelle les fautes/pannes peuvent être détectées et le décalage des réponses dans le temps. Le temps de la réponse le plus rapide vient avec un coût de communication très élevé. Le contrôle local peut être considéré comme un processus qui déclenche l'auto-organisation.

Le traitement de cette procédure est indispensable. Chaque agent doit diffuser périodiquement un message à ses voisins pour signaler sa présence, appelé « Signal de vie ». Si un agent *ax* ne reçoit pas ce message après une certaine durée *d* de son voisin direct *ay*, il le considère en panne ou leur lien de communication est détruit, dans ce cas l'agent *ax* appelle d'autres procédures pour régler le problème.

6) *Processus Quorum d'Agent*

Le quorum 'sensing' est un mécanisme permettant de déterminer s'il y a un nombre d'agents suffisants. Un programme d'agent basé sur ce principe serait utile dans certaines situations où il est nécessaire de déterminer le nombre d'agents avant de commencer une activité donnée. Chaque message envoyé à un agent récepteur aura comme résultat d'exécution une ou un ensemble de tâches.

Chaque tâche peut être exécutée par un seul agent ou par un groupe d'agents. Dans le cas où une tâche nécessite plusieurs agents, il est utile de la décomposer en plusieurs sous tâches qui seront par la suite reparties entre les agents. Cependant un agent incapable d'exécuter une tâche, peut choisir une solution parmi les suivantes :

- Chercher l'agent ou le groupe d'agents le plus proche et leur affecter les sous-tâches.
- Créer des agents en nombre suffisant qui seront capables d'assurer les mêmes fonctionnalités que l'agent créateur.

7) *Processus Point de Contrôle et Consensus*

Certaines décisions exigent qu'un ensemble d'actions parallèles soient complétées avant de passer à la phase suivante (Principe de FORK / JOIN). Ces deux processus sont utiles dans le cas de la coordination et la synchronisation des comportements à condition que le quorum d'agents soit suffisant pour procéder, et que tous les agents soient d'accord pour le procédé.

La figure qui suit (figure 1.3) [Hal 06] est un schéma présentant les interactions entre les différents processus du protocole.

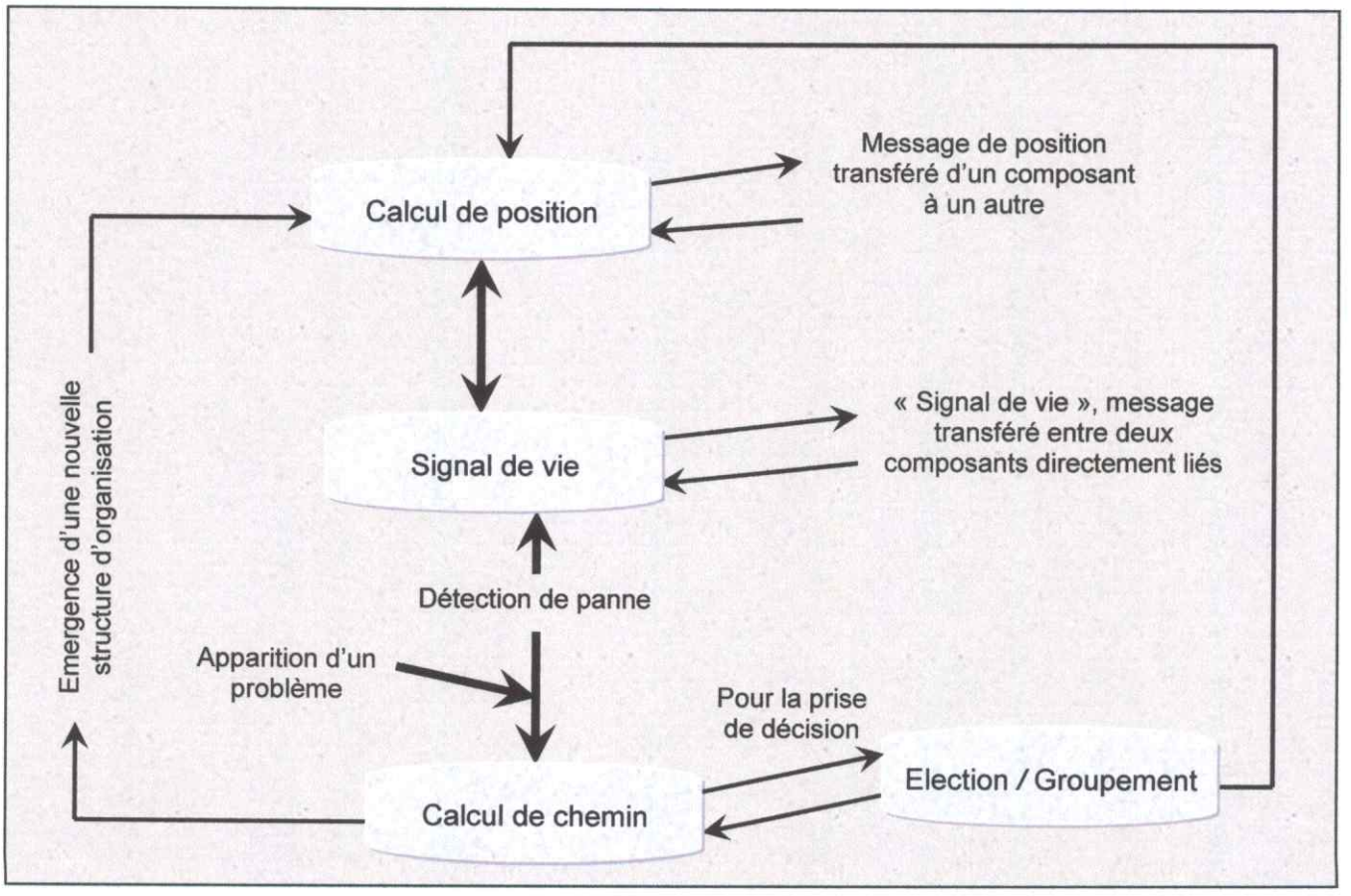


Figure 1.3. Diagramme d'interaction entre les différents processus lors des interactions entre les sources d'informations.

8. Conclusion

Nous avons présenté une étude de l'univers multi-agents, tout en essayant de clarifier la terminologie du domaine : IAD, agent, SMA, environnement... L'IAD et les SMA sont des thèmes de recherche en cours d'exploitation. Ils font intervenir plusieurs domaines de recherche tel que les systèmes répartis, l'industrie, l'IA.... La communication et l'interaction dans un SMA fait ressortir la notion d'organisation et réorganisation.

Dans ce chapitre, l'organisation et l'auto-organisation ont été les mots clés pour ensuite citer le protocole auto-organisable dédié pour un système multi-agents afin de lui donner plus de robustesse dans ses interactions.

Dans le chapitre suivant nous aborderons les systèmes de production auxquels nous donnons une interprétation SMA pour pouvoir par la suite utiliser le protocole susdit.

CHAPITRE II

Les systèmes de production

1. Introduction

Aujourd'hui, les entreprises doivent évoluer dans un environnement très incertain, changeant et dominé par une forte concurrence internationale. Actuellement, les entreprises doivent se placer dans une démarche constante d'évolution pour rester compétitives, afin de répondre aux attentes et aux besoins des clients. Pour rester compétitives, il faut s'adapter toujours plus rapidement et garantir leur réactivité. Afin d'atteindre cet objectif les entreprises doivent maîtriser la complexité de leurs systèmes de production (SdP).

Nous allons, dans ce chapitre, donner une définition des SdP, leurs caractéristiques, ainsi que le pilotage des systèmes de production.

2. Système de production

Un système de production est un ensemble de ressources réalisant une activité de production. La production est la transformation de ressources (machines et matières) conduisant à la création de biens ou de services. [Gia 88]

La transformation s'effectue par une succession d'opérations (tâches) qui utilisent des ressources (machines et opérateurs) et modifient les matières premières ou composants entrants dans le SdP afin de créer les produits finis sortants de ce système et destinés à être consommés par des clients. Les modifications peuvent porter sur la forme du produit, sa structure, son apparence, etc. La transformation subie par les produits leur procure de la valeur ajoutée. Les ressources appartenants au SdP mobilisées pour réaliser l'activité de production peuvent être des machines, des opérateurs, de l'énergie, des informations, des outillages...

Les caractéristiques d'un SdP sont [Dra 98] :

a) Flexibilité

La flexibilité d'un système de production se caractérise par sa capacité d'adaptation à la production des nouveaux produits pour lesquels le système n'a pas été étudié. Cela suppose une adaptation totale du système de production au produit

courant (de la distribution des flux discrets de composants aux opérations qu'effectuent les moyens de production sur le produit).

Plusieurs types de flexibilité ont été mis en évidence :

- *Flexibilité de produits* : offre la possibilité d'une reconfiguration du système pour la prise en compte d'un nouveau produit ou famille de produits permettant ainsi un gain de productivité ;
- *Flexibilité de mélange* : c'est la possibilité de produire simultanément un ensemble de produits ayant des caractéristiques de base communes ; cette flexibilité peut être mesurée par le nombre de produits différents qui peuvent être fabriqués simultanément ;
- *Flexibilité de quantité* : il s'agit de la capacité du système à faire face aux fluctuations de la quantité des produits à fabriquer en modifiant les rythmes, ainsi que les temps de passage et d'engagement des outils ;
- *Flexibilité de routage* : offre au système les moyens d'un aiguillage plus souple, de façon à servir les différents segments de procédés libres ou sous - engagés ;
- *Flexibilité d'ordre des opérations* : permet de changer l'ordre des opérations en cours de production (ce qui suppose l'existence d'une gamme principale et des gammes secondaires) ou de choisir la destination suivante après chaque opération ;
- *Flexibilité d'expansion* : autorise une extension et une modification de l'architecture du système et elle exige une modélisation ;
- *Flexibilité des ressources* : c'est la capacité des ressources à effectuer plusieurs tâches élémentaires et de permettre la reprogrammation.

b) <u>Réactivité</u>

La réactivité d'un système de production est définie comme l'aptitude à répondre (réagir) dans un temps requis aux changements de son environnement interne ou externe (aléa, situation nouvelle, perturbation, sollicitation, ...) par rapport au régime (fonctionnement) permanent (stable).

La réactivité d'un SdP impose une vision dynamique des événements qui se passent dans le système. Afin d'assurer cette propriété de réactivité du SdP, trois fonctions annexes s'avèrent nécessaires :

- *Une fonction d'observation* qui collecte les variables nécessaires au suivi, afin de connaître l'état courant du système (disponibilité et état des produits, disponibilité et état des moyens de production) ;
- *Une fonction de surveillance* qui détecte (suite au résultat d'une observation) et interprète les écarts et les changements entre le plan prévisionnel et le plan courant par anticipation ;
- *Une fonction de correction* qui tente à tout instant de corriger les écarts entre ces plans, ce qui implique un ordonnancement dynamique.

c) Proactivité

La proactivité d'un système de production se caractérise par ses capacités d'anticipation (prévoir et/ou provoquer) les changements d'état, d'apprentissage et d'enrichissement des connaissances (pour améliorer sa réactivité), d'adaptation ses règles de fonctionnement et par sa capacité de réorganisation reposant sur une architecture décentralisée et une délégation de responsabilité.

Un SdP proactif est avant tout un système réactif. La proaction sous-entend l'existence de la réaction. La réaction consiste dans l'application de règles fixées, en réponse aux événements, tandis que la proaction, en considérant la définition donnée ci-dessus, ajuste en quelque sorte son environnement et modifie les règles de fonctionnement afin de gérer et maîtriser les aléas néfastes à la performance industrielle.

d) Robustesse

La robustesse d'un système de production se définit par son aptitude à produire conformément aux résultats attendus. Cela suppose la garantie de l'obtention des performances souhaitées en présence d'incertitudes dans le système.

L'acquisition de ces quatre propriétés est liée à une réorganisation notable du SdP existant, notamment au niveau de la conduite du système par la prise en compte des nouvelles approches.

3. Décomposition d'un système de production

L'application de la théorie des systèmes [Hab 06] aux SdP suggère une décomposition de ces derniers en deux sous-systèmes :

3.1. Un sous-système d'information et de décision

Le sous-système d'information et de décision comprend une partie contrôle ou pilotage qui représente la partie intelligente du système, c'est-à-dire ; systèmes de commande, systèmes de supervision, acteurs humains (gestionnaires, responsables, opérateurs, etc.). La partie pilotage associe des points d'acquisition et de collecte de l'information, des processus décisionnels (analyse, traitement de l'information et évaluation pour générer des décisions) et des actions (points de transfert des ordres ou des actions de la partie pilotage aux autres parties du système).

3.2. Un sous-système physique de production

Le sous-système physique de production comprend deux parties :

- Une partie flux constituée par un flux de matière ou d'entités à transformer ou à assembler et qui représentent généralement matières premières, composants, produits en cours de fabrication et produits finis. Le flux formé par ces éléments peut se trouver sous différentes formes : pièces unitaires, groupements temporaires, assemblages, lots, séries.
- Une partie physique qui représente l'ensemble des moyens nécessaires à la réalisation des opérations sur le flux de matière. Elle comprend les machines, les espaces de stockage, les moyens de transfert, les robots, les manipulateurs, etc.

La figure suivante (figure 2.1) schématise la décomposition d'un SdP :

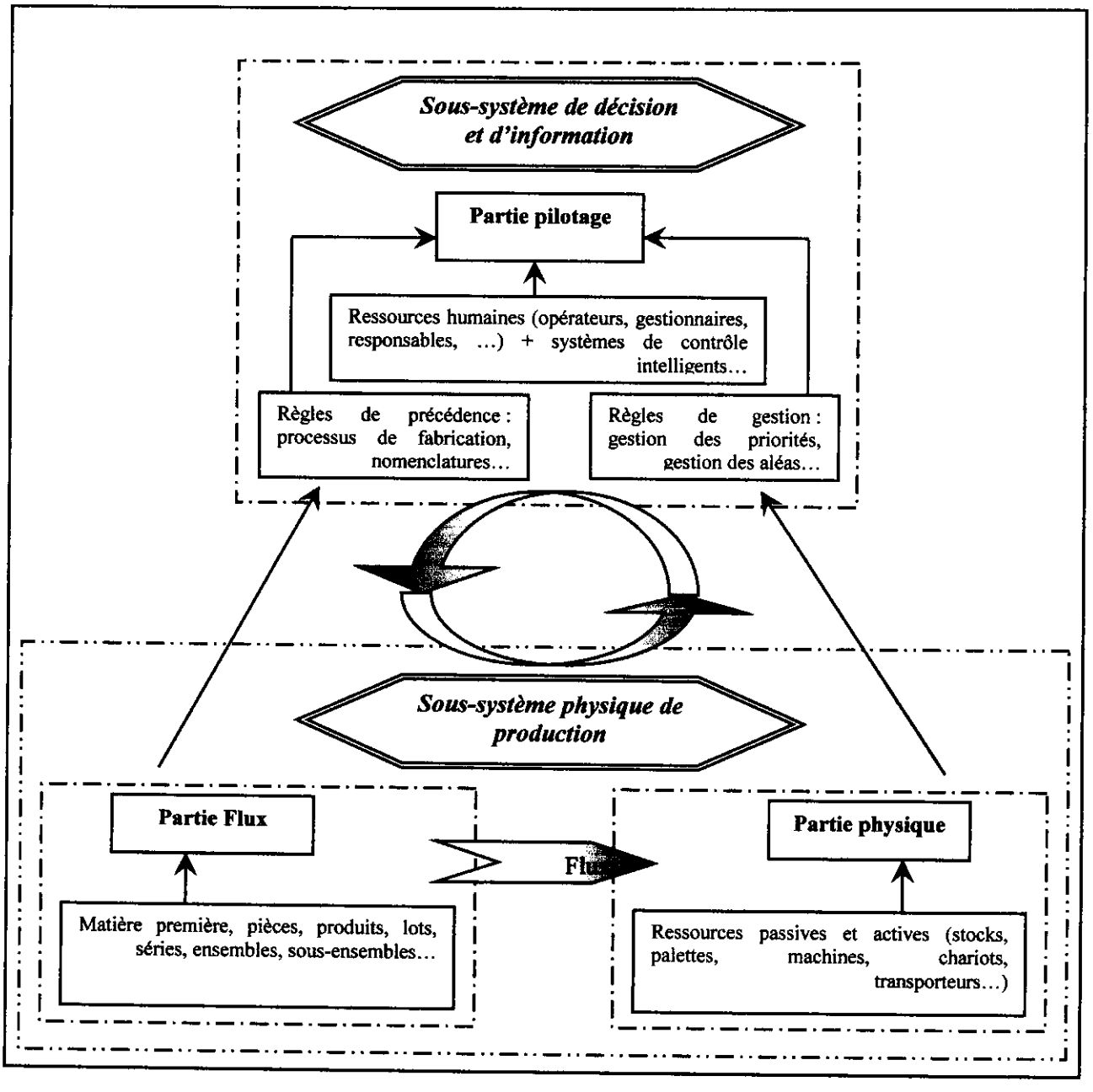


Figure 2.1. Décomposition d'un SdP.

4. Fonctionnement d'un système de production

Un système de production se compose d'un ensemble de ressources (machines), d'hommes (opérateur), de moyens de transport, de produits et de système de stockage. La matière première doit subir plusieurs transformations avant d'en sortir comme un produit fini.

Suivant les objectifs globaux du système de production l'opérateur élabore un plan de production afin de réaliser l'ensemble des produits sur les ressources composant le système.

Le plan se définit par l'affectation des tâches aux différentes ressources afin de réaliser un produit et fixer leurs ordres de passage (ordonnancement). La figure 2.2 illustre un exemple d'un système de production.

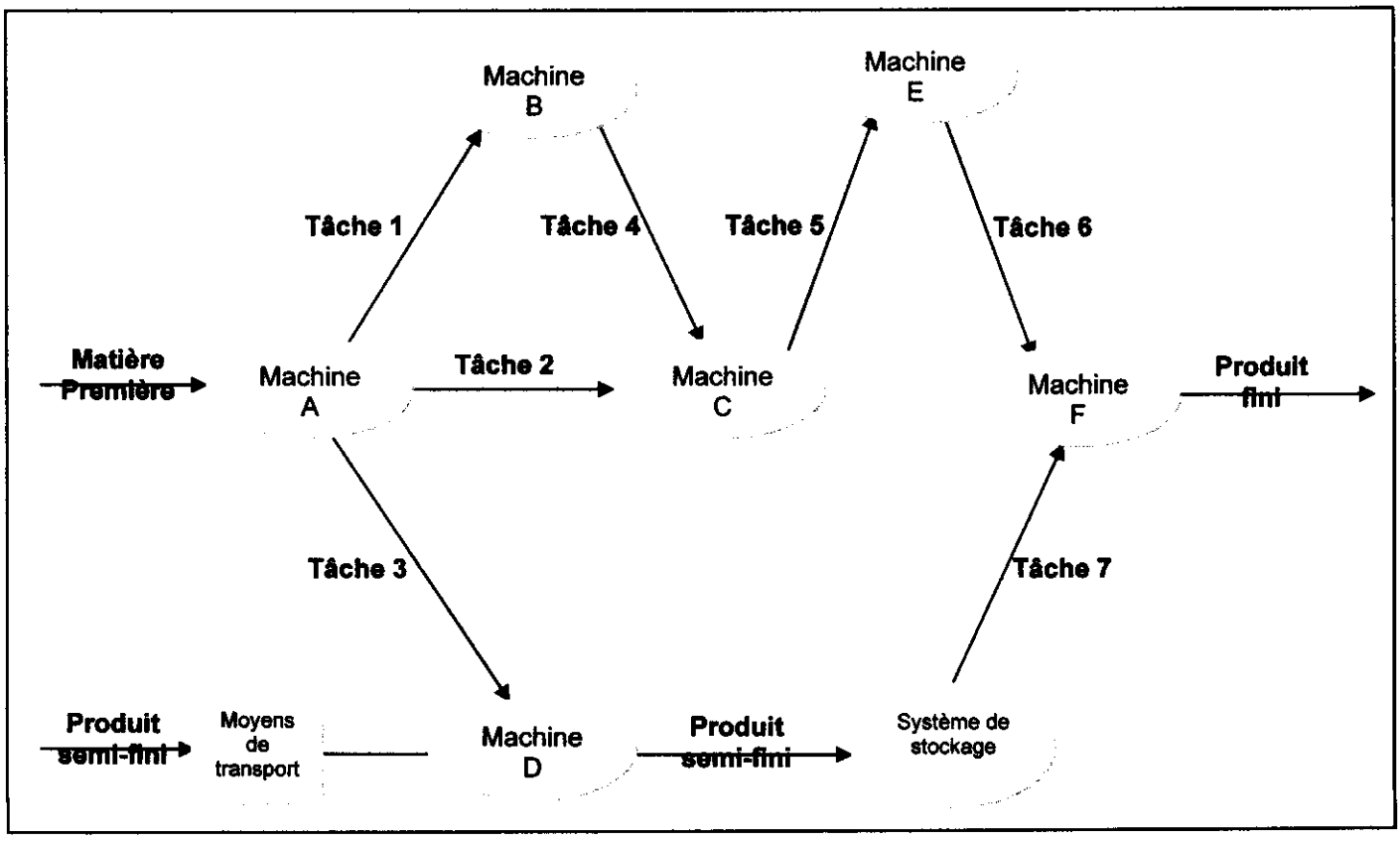


Figure 2.2. Exemple d'un système de production.

Pour gérer le déroulement des opérations sur les moyens de production la conduite doit respecter les contraintes techniques suivantes :

- une ressource exécute au plus une tâche à la fois (contrainte d'exclusion mutuelle) ;
- une tâche commencée sur une ressource doit être terminée sans interruption sur la même ressource (contrainte de non - préemption) ;
- l'exécution d'une tâche peut nécessiter un ensemble de tâches qui doivent être effectuées dans un ordre spécifique (contrainte de précédence).

L'affectation des tâches aux différentes ressources et leurs exécutions se traduisent par un échange de tâches, de produits (semi-finis ou matière première) et d'informations entre les différentes ressources du système. Nous trouvons ici un ensemble de flux régulés qui parcourent le système de production :

- ✓ Tout d'abord, le flux physique qui transforme la matière première et les composantes en produits finis.
- ✓ Puis le flux d'informations qui permet la circulation des informations nécessaires au contrôle et la prise de décision.

5. Application des systèmes multi-agents dans le SdP

Les agents offrent une nouvelle approche pour la modélisation des systèmes de production. Au lieu de modéliser les systèmes distribués à l'aide de programmes échangeant données et commandes, la technologie agent permet la création d'agents décideurs autonomes qui communiquent entre eux, négocient des sous objectifs et coordonnent leurs intentions de manière à atteindre les objectifs propres au système. [Bus 04]

Dans ce cadre plusieurs approches ont été citées pour modéliser un système de production, parmi elles nous pouvons citer :

Dans les travaux de [Roy 98], une plate-forme SMA est construite pour le pilotage d'ateliers. Différents types d'agents sont proposés (agents ressource, agent cellule et agent produit) qui représentent des entités physiques, des îlots virtuels ou bien des enchaînements d'opérations. Un agent superviseur a pour rôle de contrôler le

processus de production. Il est aidé d'un agent méta-objet afin d'insérer de nouveaux agents dans le système.

Dans [Kou 97], les auteurs exploitent également une architecture multi-agents dans laquelle chaque agent supervise une ressource de production. Un agent superviseur est chargé de contrôler l'ensemble du système de production via la communication avec les agents responsables d'une ressource de production. Ces derniers prennent des décisions quant aux règles de production à appliquer sur les ressources qu'ils supervisent. Cependant, l'agent superviseur peut intervenir et indiquer à chaque agent quelle règle appliquer afin d'atteindre les objectifs globaux du système de production. Parmi les limitations des deux approches, nous avons souligné :

- Centralisation du système au niveau de l'agent superviseur.
- Possibilité de saturation au niveau de l'agent superviseur (trop de messages provenant des autres agents du système).
- La panne de l'agent superviseur provoque l'arrêt de fonctionnement du système.
- Coût de communication très élevé (le temps que doit prendre un message pour parvenir à destination, sachant que ce message devra passer par l'agent leader du système multi-agents).

6. Le pilotage des systèmes de production

6.1. Définition

« Le pilotage est responsable de la réalisation effective de la production en prenant en compte l'ensemble des paramètres et contraintes de fabrication, et en réagissant aux aléas.

Piloter un système de production, c'est en fait accomplir un certain nombre de fonctionnalités comme l'ordonnancement (Scheduling), l'exécution (Dispatching,), la surveillance (Monitoring and Error Handling), la gestion des communications entre les différentes ressources du système, etc. ». [Sua 98]

6.2. Les fonctions d'un système de pilotage

Le pilotage temps réel fait appel, pour La réalisation du programme prévisionnel, à un certain nombre de fonctions [Her 98] l'ordonnancement, l'exécution et la surveillance.

6.3. L'ordonnancement dynamique (Scheduling)

L'ordonnancement dynamique consiste à programmer dans le temps, l'exécution des différentes opérations des ordres de fabrication, en présence de contraintes sur les ressources disponibles et compte tenu d'objectifs à satisfaire. C'est une série d'allocations dont chacune est un triplet (opération, où, quand). La gestion de ces allocations se fait dynamiquement au fur et à mesure de l'évolution événementielle des SdP. [Sab 98]

Les décisions à prendre pour la construction de l'ordonnancement temps réel sont :

- ✓ Les décisions d'affectation d'une ressource à une opération. Pour chaque opération, il s'agit de choisir la ressource qui peut exécuter l'opération, compte tenu de l'état du SdP.
- ✓ Les décisions d'attribution d'une ressource à une opération. il s'agit de choisir la date T à partir de laquelle il y'aura l'exécution effective de l'opération sur la ressource qui lui est affectée.

6.3.1. L'exécution (Dispatching)

L'exécutons génère des requêtes d'opérations aux contrôleurs d'équipements. Elle assure l'exécution des opérations élémentaires sur le système piloté. Elle interagit avec le système de surveillance.

6.3.2. La surveillance (Monitoring and Error Handling)

La notion de sûreté de fonctionnement n'est pas réaliste dans un système réel. Il faut s'attendre à ce que des défaillances puissent arriver, et donc pouvoir les

détecter et les traiter. La surveillance est l'une des tâches de première importance dans l'automatisation des SdP. Elle permet d'assurer une certaine tolérance aux pannes pouvant survenir en cours de fonctionnement du système de production se rapportant principalement :

- ❖ aux pannes d'origine matérielle qui peuvent concerner les moyens de production machines, unités de manutention, etc.
- ❖ aux pannes d'origine logicielle qui peuvent concerner les moyens de pilotage: élément de commande, réseau de communication, etc.

La surveillance regroupe plusieurs fonctions [Nou 94] :

6.3.2.1. Le suivi

Cette fonction collecte des informations permettant d'établir des tableaux de bord nécessaires à l'évaluation de l'état et des performances des ressources, des stocks et de la production par l'intermédiaire de capteurs.

Ces fonctions majeures sont:

- ❖ suivi des états des opérations, des pièces, et des ressources,
- ❖ filtrage et classification des informations provenant des contrôleurs d'équipements,
- ❖ communication des changements d'états du SdP,
- ❖ entretien des statistiques sur les ressources du SdP (taux d'usure des outils, dates d'initialisation, etc.).

6.3.2.2. Détection

La détection permet d'identifier les dysfonctionnements. Il existe diverses techniques de détection, parmi celles-ci, on peut citer:

- **L'identification** : cette technique identifie le procédé et exploite les résultats de l'identification pour détecter les erreurs qui peuvent se produire.
- **Le filtrage** : consiste à traiter des signaux qui apportent des informations sur l'état réel du système de production et sa divergence par rapport à l'état normal.
- **Chien de garde** : c'est une technique classique qui permet de temporiser les actions lancées au sein du SdP. Si la durée fixée est dépassée, une erreur est détectée.

6.3.2.3. Le diagnostic

Le diagnostic permet d'identifier et de localiser les causes qui ont provoqué l'apparition des symptômes (dysfonctionnements). L'une des techniques la plus utilisée de diagnostic est l'Intelligence Artificielle, en particulier les systèmes experts. Le système expert déclenche un raisonnement (moteur d'inférences) sur les connaissances décrivant l'environnement réel du SdP (base de connaissances), pour identifier et localiser les causes des symptômes détectés. Ces symptômes sont décrits dans la base des faits.

6.3.2.4. Traitement et reprise

Le traitement des pannes consiste à invoquer des actions correctives permettant soit d'anticiper l'occurrence de la défaillance, soit de corriger en agissant sur le procédé.

Les traitements possibles non exclusifs sont [Jai 97] :

- Réordonnancement: reprise des opérations.
- Arrêt de l'exécution d'une opération.
- Appel opérateur.

- ❖ Arrêt en position de repli : la mise en position de repli permet l'intervention de l'opérateur. Il y'a complémentarité entre l'automatisation de la mise en position de repli et les actions de traitement laissées à l'opérateur.
- ❖ Arrêt en mode dégradé correspond à un mode de fonctionnement où le système de production est amputé de l'une de ses ressources jugée défaillante afin de limiter les temps d'arrêts du SdP et assurer une continuité de fonctionnement.

6.4. Typologie des architectures de pilotage

Il existe dans la littérature trois architectures de pilotage de systèmes de production [Oul 99] : l'architecture centralisée, l'architecture hiérarchisée, et l'architecture distribuée.

6.4.1. Architecture centralisée

L'architecture centralisée est très classique. Elle se caractérise par la l'existence d'un centre de décision unique, appelé superviseur, chargé de piloter l'ensemble des moyens de production, comme le démontre la figure suivante (figure 2.3). Ce dernier assure la coordination et la synchronisation des activités du SdP. Il est donc nécessaire qu'il dispose d'un système d'exploitation multitâche. La plupart des premiers systèmes de pilotage exploités dans les systèmes de production ont adopté cette architecture, tels que: COSIMA, MASCOT, SAGEM, PROFLEX, PILOTE

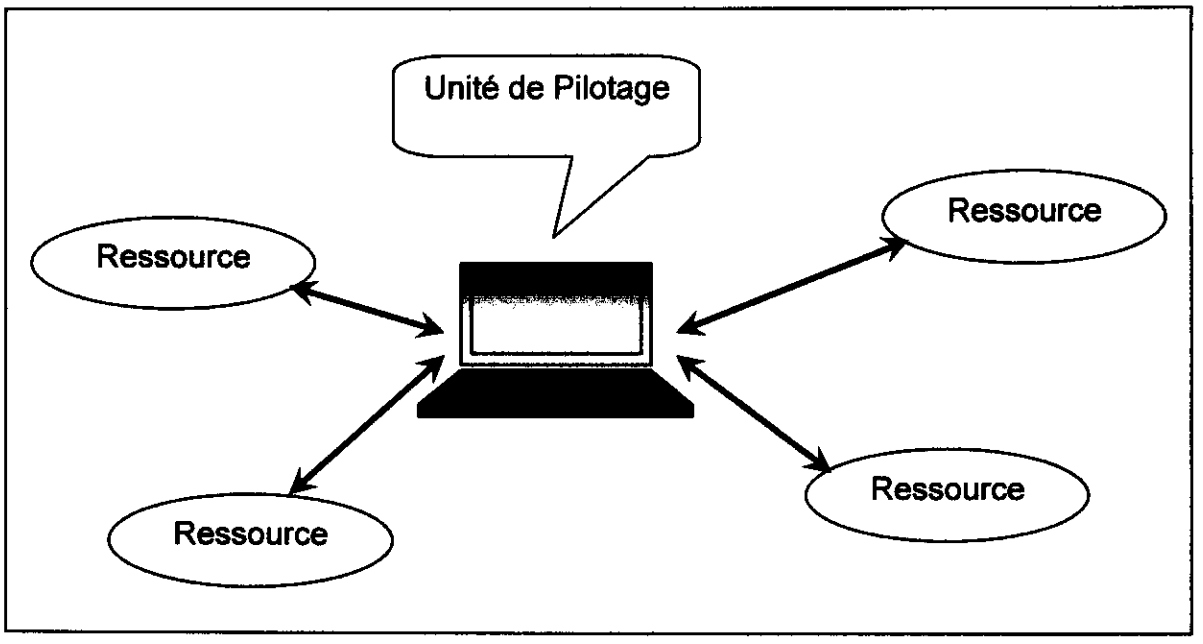


Figure 2.3. Architecture centralisée.

6.4.2. Architecture hiérarchisée

Dans l'architecture hiérarchisée, le pilotage du SdP est assuré par une hiérarchie d'unités de pilotage. Le niveau le plus bas de la hiérarchie représente les ressources. Chaque niveau coordonne les unités de pilotage de niveau inférieur, et ce jusqu'au niveau le plus bas. La relation, à un niveau donné, est donc de dépendance vis à vis du niveau supérieur et de dominance vis à vis du niveau inférieur. Chaque niveau élabore ses décisions en fonction des ordres reçus du niveau immédiatement supérieur, et des résultats transmis par le niveau inférieur. Ceci induit des relations de type maître/esclave entre les niveaux.

La figure suivante (figure 2.4) représente un schéma d'une architecture hiérarchisée.

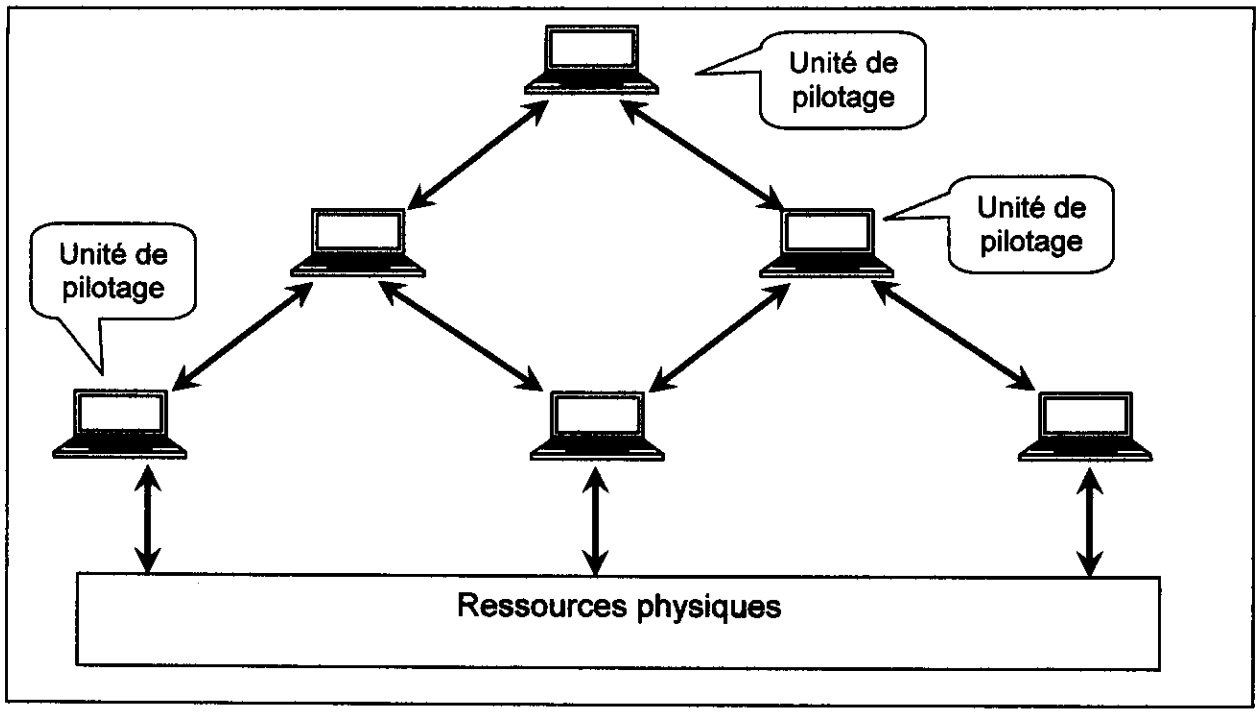


Figure 2.4. Architecture hiérarchisée.

6.4.3. Architecture distribuée

L'architecture distribuée est fondée sur une distribution totale des capacités décisionnelles parmi un ensemble de ressources pilotant le système de production. Chaque ressource est munie d'un calculateur, appelé unité de pilotage qui assure le pilotage local de sa ressource. Ces différentes unités de pilotage communiquent entre elles et coopèrent pour réaliser les activités de production (figure 2.5).

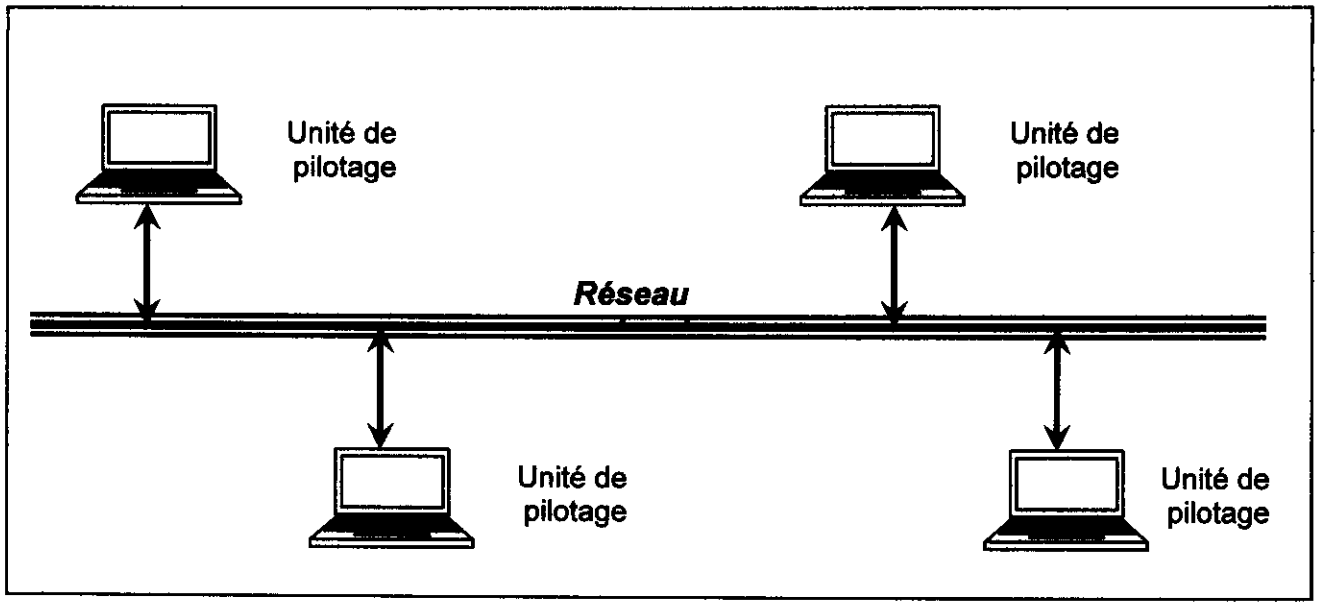


Figure 2.5. Architecture distribuée.

7. Conclusion

Dans ce chapitre, nous avons parlé des systèmes de production, composants et caractéristiques à savoir robustesse, proactivité, réactivité et flexibilité.

Ensuite nous avons défini le pilotage d'un SdP ; celui-ci se compose de trois fonctionnalités (l'ordonnancement dynamique, l'exécution et la surveillance) nécessaires et étroitement dépendantes les unes aux autres.

Enfin nous avons parlé des trois types d'architecture de pilotage, et nous pouvons constater que l'architecture distribuée est la mieux adapté et répondants aux exigences d'un pilotage temps réel.

Dans le chapitre qui suit, nous allons nous spécifier sur la fonction surveillance en appliquant le protocole auto-organisable sur l'approche multi-agents que nous allons proposer.

PARTIE 2

Conception et implémentation

CHAPITRE III

Conception

1. Introduction

Ce chapitre décrit l'architecture multi-agents proposée pour le pilotage temps réel de systèmes de production. Il expose, ainsi, l'architecture distribuée de pilotage, les modèles d'agents de l'architecture ainsi que les fonctions assignées à ces agents. Pour la coopération entre ces agents, un nouveau protocole de communication est présenté, dont la caractéristique principale est l'auto-organisation. Grâce à ce protocole, la coordination des actions des agents pour la réalisation des tâches n'est pas dédiée à un superviseur central, mais distribuée et émerge de l'interaction de ces agents. Nous allons, aussi, voir que la distribution de la surveillance garantie un niveau de fiabilité, de robustesse et de sécurité face aux pannes. Enfin nous nous intéresserons à la fonction surveillance et traitement des pannes du pilotage temps réel du système (panne machine, panne agent ressource pilote ou bien problème du lien de communication).

2. Approche multi-agents proposée

L'approche multi-agents que nous proposons est une approche basée sur la décentralisation du système de production, ce qui veut dire que le système multi-agents ne contient pas un même agent leader de façon permanente.

L'approche est basée sur trois types d'agents :

- Les agents ARP (Agent Ressource Pilote) qui pilotent un ensemble de ressources (machines) hétérogènes. Ces agents communiquent entre eux afin de réaliser le plan de production constitué par un ensemble de tâches. Chaque agent possède un niveau de connaissances suffisant pour une prise de décision autonome et pour assurer la gestion locale d'une ressource, en intégrant les fonctions du système de pilotage en temps réel, à savoir : l'ordonnancement, l'exécution et la surveillance. Chaque agent ressource est capable de réaliser ces fonctions de façon autonome. Ceci augmente la flexibilité et l'extensibilité de l'architecture, une forte tolérance aux pannes, la facilité d'intégration de nouveaux agents ainsi que la gestion des départs

d'agents, enfin la répartition du coût global de la communication sur l'ensemble des agents.

- Agent utilisateur (l'opérateur), cet agent est l'opérateur, son rôle est d'élaborer le plan de production de tout le système de production.
- Agent interface, cet agent joue le rôle d'interface du SdP (intermédiaire entre les agents ARP et l'opérateur).

2.1. Les connaissances des agents

2.1.1. Agent utilisateur

L'agent utilisateur représente l'opérateur, les connaissances de cet agent sont :

Des informations concernant les ressources :

- Identificateur de chaque ressource.
- Liste de toutes les ressources dans le SdP.
- Liste des tâches à réaliser sur chaque ressource.

Des informations sur les tâches :

- Identificateur de chaque tâche.
- Liste de tous les produits à réaliser.
- Liste de toutes les tâches nécessaires pour réaliser un produit.
- Durée d'exécution de chaque tâche.
- Priorité de chaque tâche.
- Liens de précédence.
- Etat de chaque tâche (en exécution, en attente, ...).

Des informations concernant les autres agents :

- Identificateur de chaque agent ARP.
- Liste de tous les agents ARP.
- Liste des tâches prise par chaque agent ARP.

2.1.2. Agent interface

Les connaissances de l'agent interface sont :

Concernant les tâches :

- Identificateur de chaque tâche.
- Priorité de chaque tâche.
- Liens de précédence.
- Durée d'exécution de chaque tâche.

Concernant les agents ARP:

- Identificateur de chaque agent ARP.

2.1.3. Agent ARP

Chaque agent du système possède un niveau de connaissances se rapportant aux informations suivantes :

Des informations décrivant la ressource :

- Identificateur de la ressource ;
- Type de la ressource ;
- Liste des tâches à réaliser sur la ressource.

Des informations concernant la tâche :

- Identificateur de la tâche ;
- Durée de la tâche ;
- Priorité de la tâche ;
- lien de précédence.

Des informations concernant les autres agents :

- Identificateur de l'agent ;
- Identificateur de la ressource pilotée par l'agent ;
- Tâches prises en charge par l'agent.

2.2. Principe de fonctionnement du système proposé

Suivant les produits à réaliser dans le SdP, l'agent utilisateur établi le plan de production qui se défini par l'affectation des tâches aux ressources afin de réaliser un

produit (ordonnancement), ce plan nous fournit des informations sur la tâche (identificateur, priorité, durée et lien de précédence).

L'agent utilisateur envoi ce plan à l'agent interface, qui a pour rôle d'intermédiaire entre l'agent utilisateur et les différents agents ARP du système.

L'agent interface envoi aux agents ARP les tâches à réaliser, chaque ARP assure la gestion locale de sa ressource en intégrant les fonctions de pilotage en temps réel, à savoir : l'ordonnancement, l'exécution et la surveillance, chaque agent peut réaliser ces fonctions de façon autonome.

1. **Ordonnancement :** l'agent établit le plan d'exécution propre à la ressource tout en respectant les priorités des tâches, et prend en considération ses liens de précédence ainsi que les durées d'exécution des tâches.
2. **Exécution :** l'agent ARP exécute les tâches affectées à sa ressource en suivant le plan d'exécution établi.
3. **Surveillance :** l'agent doit veiller au bon déroulement du processus de production, pour cela il fait le suivi, la détection, le diagnostic et le traitement et reprise en cas de panne, nous pouvons trouver ici un ré ordonnancement.
    - ***Réordonnancement*** **:** le ré ordonnancement est la réaffectation des tâches aux autres agents lorsqu'un aléa surgit dans le système ; dans notre cas par le recours au protocole auto-organisable.

Le diagramme (figure 3.1) suivant nous montre le rôle de chaque type d'agents.

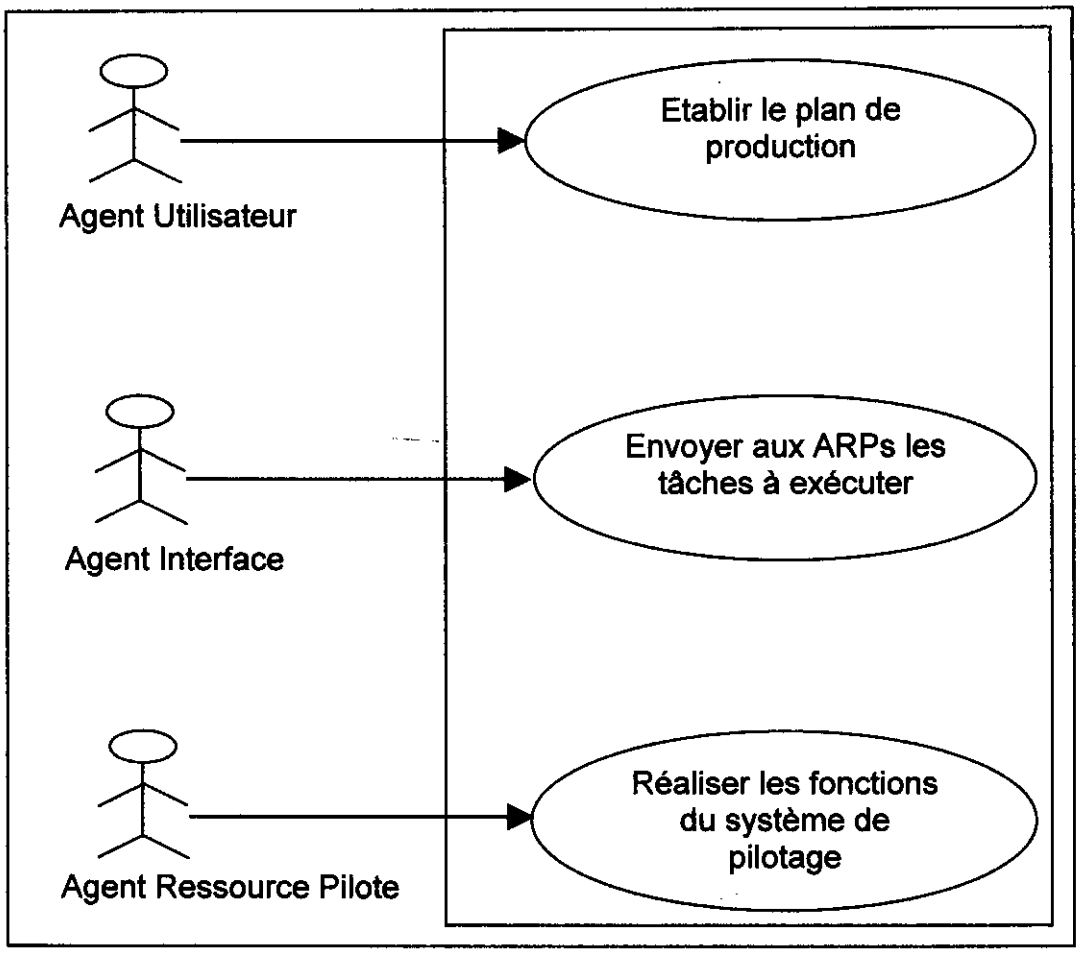


Figure 3.1. Diagramme de cas d'utilisation du système.

Le diagramme suivant (figure 3.2) nous désigne les différents états dans lesquels peut se trouver un agent ARP.

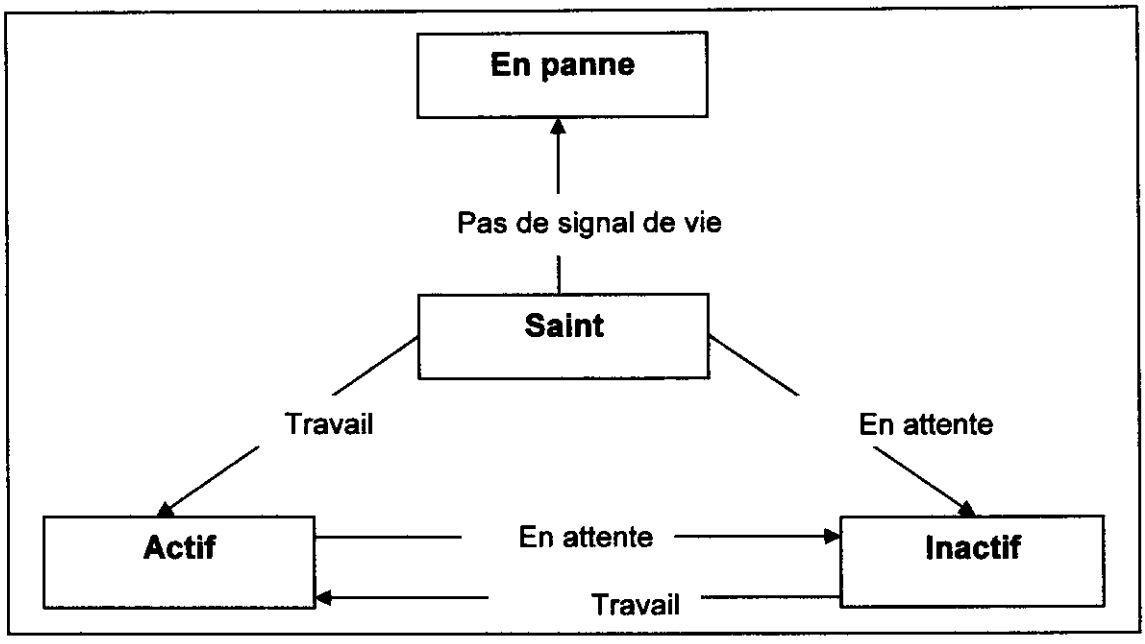


Figure 3.2. Diagramme état / transition de l'agent ARP.

La figure qui suit (figure 3.3) nous donne un aperçu sur le modèle d'un agent ARP.

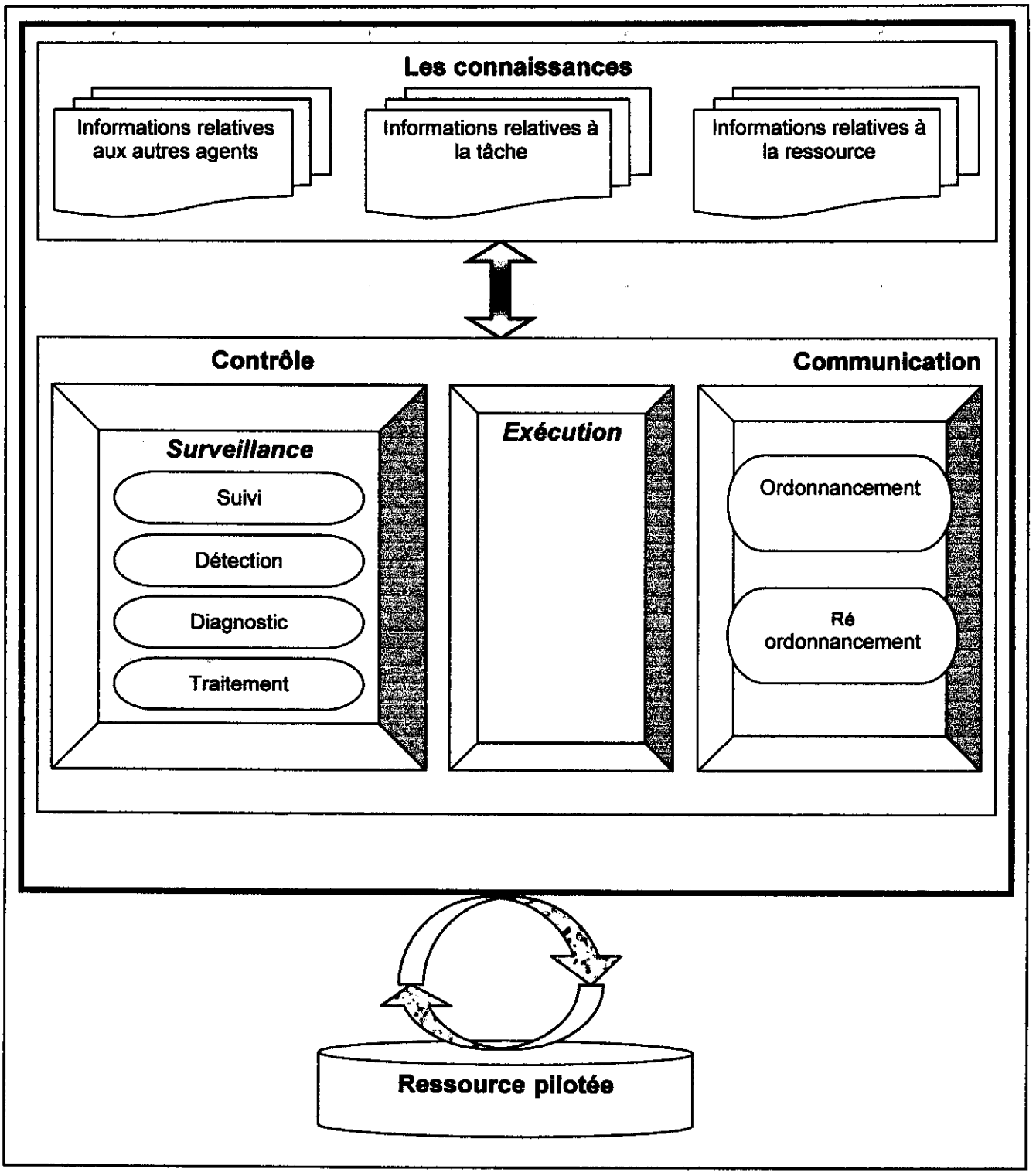


Figure 3.3. Modèle de l'agent ARP.

La figure suivante (figure 3.4) nous montre les différentes interactions entre les fonctions du système de surveillance.

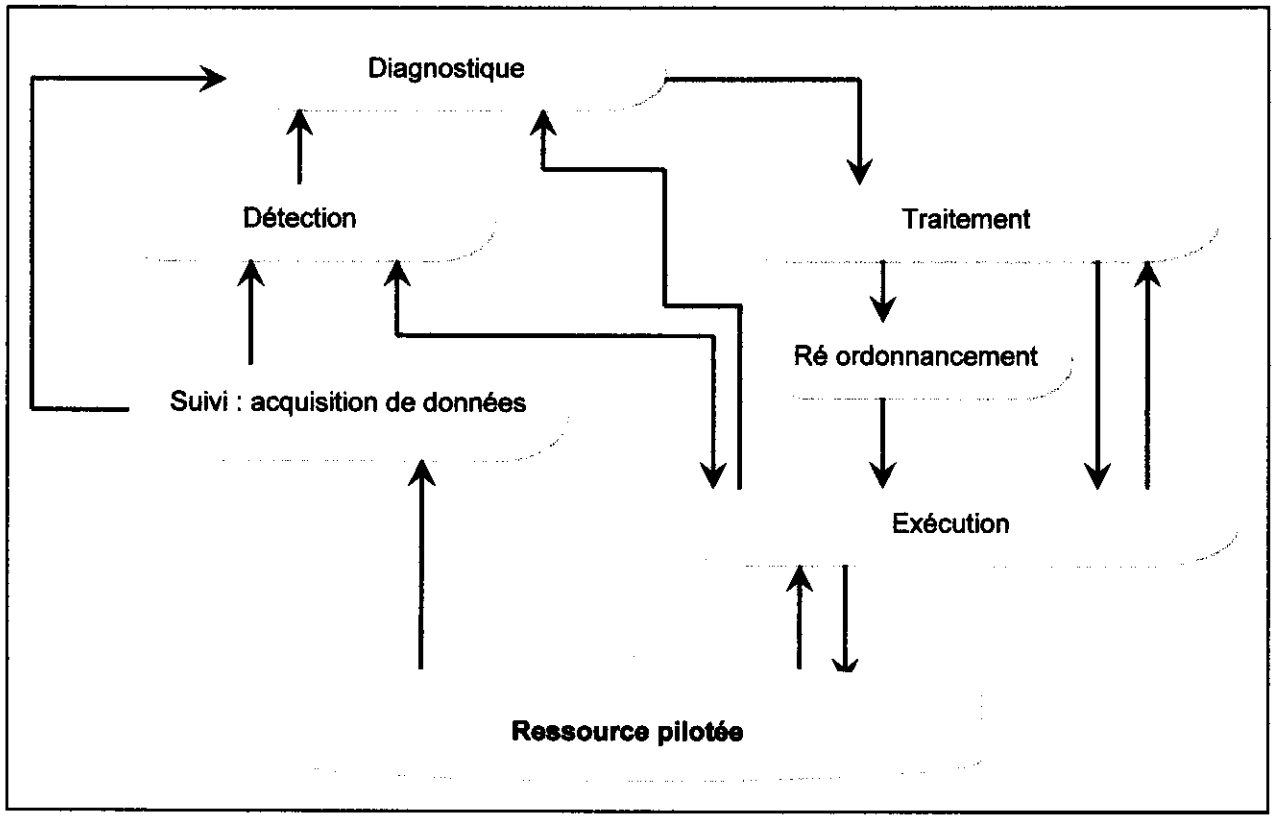


Figure 3.4. Les fonctions du système de surveillance.

Au démarrage du système, les différents agents ARP ainsi que l'agent interface exécutent deux processus du protocole auto-organisable, ces deux processus sont :

- **Processus calcul de position :** le processus calcul de position permet de calculer la distance entre les agents du système et un autre agent du même système considéré comme étant une source d'information. Apres l'exécution de ce processus, chaque agent du système aura une connaissance de tous ses voisins et leurs rôles.
- **Processus signal de vie :** le processus signal de vie permet de savoir si un agent est en panne ou un lien de communication est détruit, cette détection se réalise lorsqu'un agent ne reçoit pas un signal de vie de la part de son voisin directement lié.

Les deux processus cités ci-dessus sont les plus importants, ils permettent la détection d'un dysfonctionnement du système multi-agents mis en place; les autres processus seront exécutés par les agents ARP pour le traitement des pannes pouvant survenir.

Nous rappelons que l'ensemble des processus du protocole est :

- → Processus signal de vie.
- → Processus calcul de position.
- → Processus calcul de chemin.
- → Processus élection.
- → Processus groupement.
- → Processus Quorum d'Agents.
- → Processus Point de contrôle et Consensus.

Les deux derniers processus ne seront pas utilisés par la suite.

Le diagramme suivant (figure 3.5) nous indique comment se fait la collaboration entre les différents types d'agents du système.

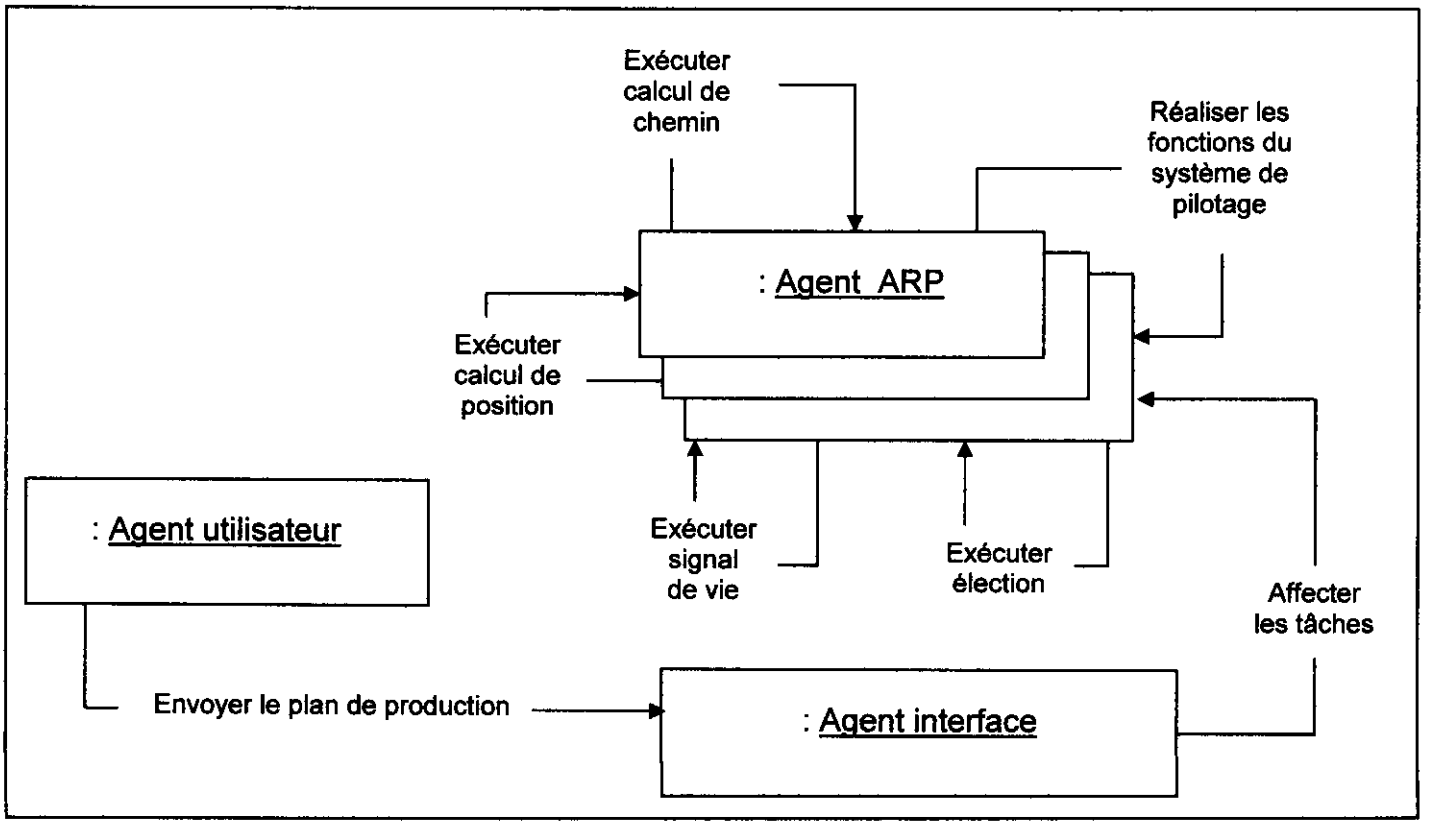


Figure 3.5. Diagramme de collaboration du fonctionnement du système.

2.3. La communication

Les agents doivent disposer d'un langage de communication commun pour pouvoir échanger des informations et coopérer pour la résolution d'un problème. Ce langage est un ensemble de primitives (messages) connues par chaque agent. Ces primitives constituent le protocole de communication qui permet de structurer et d'uniformiser les interactions inter-agent.

Dans le tableau suivant nous trouvons toutes les primitives de base du protocole de communication que nous avons adopté :

Primitives	**Définition des paramètres**	**Rôle**	**Appartenance**
Position (a_{src}, rôle, pos_{src})	**a_{src} :** identité de l'agent qui envoi le message. **rôle :** le rôle de l'agent asrc. **pos_{src} :** la position des voisins incrémentés par l'agent source.	Permet à un agent de calculer sa position par rapport à un autre agent.	Processus calcul de position.
Leader (chef_glo, fitness)	**chef_glo :** contient chef_loc : identité de l'agent émetteur. **fitness :** valeur initialisé à zéro puis incrémentée après l'exécution d'une tâche.	Cette primitive nous permet de sélectionner parmi le SMA un agent leader.	Processus élection.
Groupe (chef_glo, fitness)	//	Cette primitive nous permet de sélectionner parmi le SMA un groupe d'agents ayant des caractéristiques similaires.	Processus groupement.
Remplace	**a_{panne} :** identité de	Cette primitive permet de	Processus calcul

(a_{panne})	l'agent en panne.	prendre les tâches de l'agent en panne et de le remplacer.	de chemin.
Demande (a_{src}, a_{emet}, vois_inf)	**a_{src} :** identité de l'agent source. **a_{emet} :** identité de l'agent émetteur. **vois_inf :** ens des voisins directs de chaque agent ayant reçu ou allant recevoir le msg.	Cette primitive permet de demander à un agent de faire l'intermédiaire entre l'agent source et l'agent émetteur.	Processus calcul de chemin.
Acquitter (réponse, pos_{src})	**réponse :** oui *ou* non. **pos_{src} :** indique la position de chaque agent par rapport à l'agent source.	Acquitter suit toujours la primitive Demande, cette primitive permet à un agent d'envoyer une réponse à l'agent qui lui a envoyé le message Demande.	Processus calcul de chemin.
Vie (a)	**a :** identité de l'agent émetteur.	Cette primitive permet à un agent émetteur d'informer régulièrement ses voisins qu'il est en vie.	Processus signal de vie.
Explore (a_{src}, a_{dest})	**a_{src} :** l'agent source. **a_{dest} :** l'agent destinataire.	Le but de cette primitive est de savoir si un agent voisin de l'agent source a déjà reçu un signal de vie de la part de l'agent destinataire.	Processus signal de vie.
Rep (réponse, pos)	**réponse :** si un agent ax a déjà reçu ou non un signal de vie de a_{dest}. **pos :** la position de ax (celui qui reçoit explore (..)) par rapport à a_{dest}.	Cette primitive est une réponse au message Explore.	Processus signal de vie.

Tableau 3.1. Description des primitives de communication.

2.4. La tolérance aux pannes dans le système multi-agents :

2.4.1. Détection et diagnostic des pannes :

Afin d'assurer un bon fonctionnement du système de production, il faut que le système soit tolérant aux pannes. La tolérance aux pannes est un aspect important caractérisant le fonctionnement d'un SdP, pour cela le système multi-agents doit détecter et traiter une grande partie de pannes pouvant survenir, afin d'éviter l'intervention de l'opérateur (à part les cas extrêmes).

De façon générale, nous avons énumérer les pannes pouvant se produire dans le système comme suit :

a) Panne hard :

La panne hard apparaît lors d'un dysfonctionnement d'une machine, c'est ce que nous appellerons par la suite une panne ressource.

La panne ressource est détectée et diagnostiquée par l'agent responsable de la ressource en panne.

b) Panne soft :

La panne soft apparaît dans deux cas :

- Lien de communication détruit.
- Panne agent.

Dans ce qui suit nous allons montrer comment le système avec l'architecture multi agents proposée va être tolérant aux pannes, et d'une façon autonome, va les gérer. Mais avant de dire comment les pannes sont traitées, on va expliquer comment celles-ci seront détectées par le biais du protocole auto-organisable utilisé.

La détection et le diagnostic de la panne soft sont assurés grâce au processus signal de vie du protocole utilisé. Le signal de vie est exécuté par les agents ARP au démarrage du système. Ce processus permet aux différents agents de diagnostiquer et de détecter les pannes soft (panne agent ou lien de communication détruit), de façon autonome sans intervention externe.

La figure suivante (figure 3.6) est un diagramme de séquence de notre système.

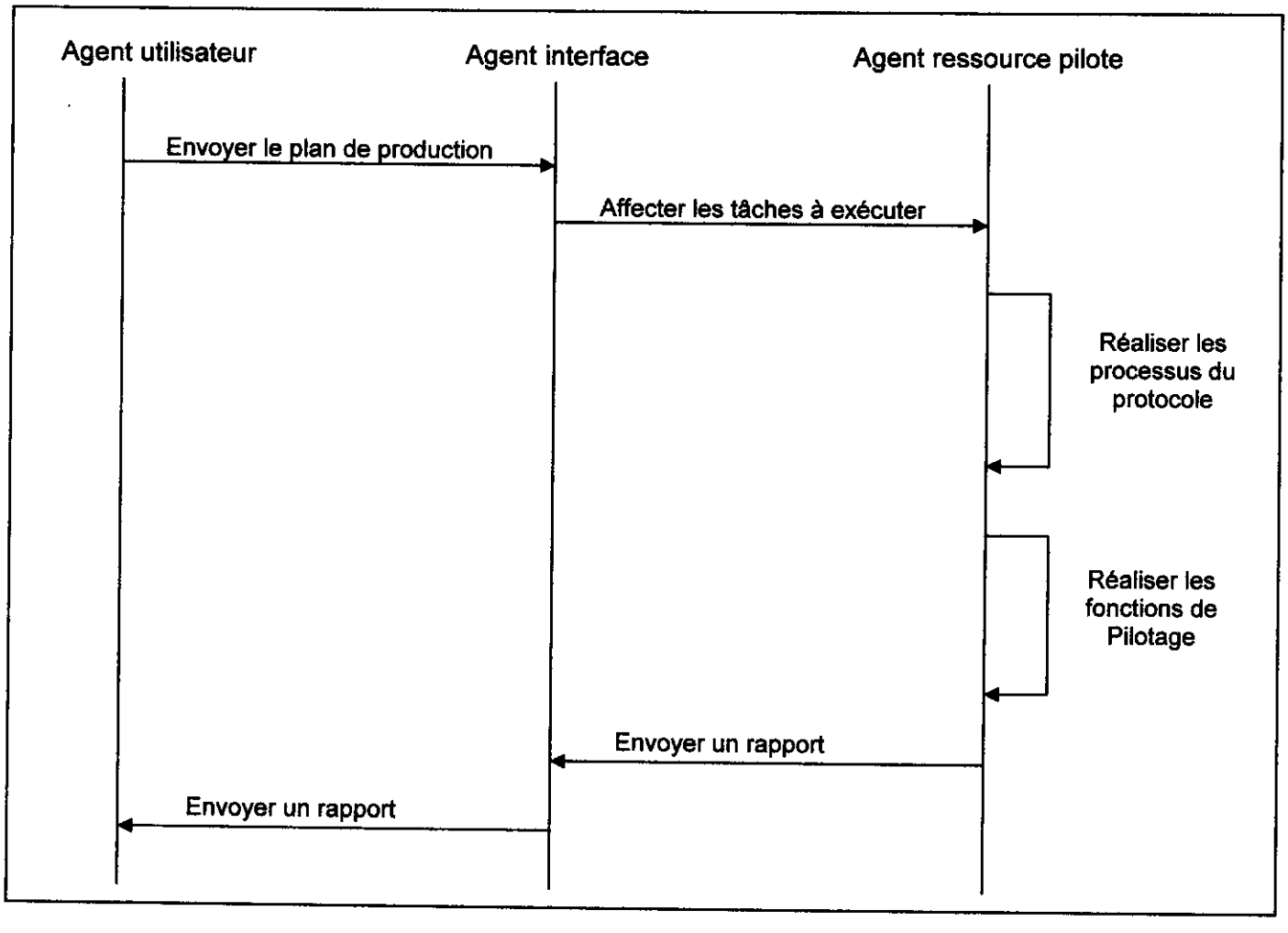


Figure 3.6. Diagramme de séquence du fonctionnement du système.

2.4.2. Traitement des pannes :

Pour traiter les pannes citées précédemment, les différents agents ARP font appel au protocole auto-organisable, qui permet aux différents agents ARP du système de traiter les pannes de façon autonome. Dans ce qui suit nous allons vous montrer comment les agents ARP du système vont traiter les pannes sus-citées.

2.4.2.1. Panne ressource

L'agent responsable de cette ressource lance le processus calcul de Chemin sur lequel nous avons apporté un changement qui est que l'agent responsable de la ressource au lieu d'envoyer l'adresse de son voisin, il envoi sa propre adresse. Ceci pour prévenir les agents voisins que l'agent responsable de la ressource en panne, ne peut exécuter l'ensemble de tâches qui lui restent ; nous allons appeler ce processus que nous avons modifié ***processus calcul de chemin3***. Après l'exécution de ce processus, nous trouvons ici deux cas :

1. Au moins un agent ARP du système peut exécuter les tâches de l'agent responsable de la ressource en panne.
2. Aucun agent ne peut exécuter les tâches de l'agent en panne et dans ce cas on fait appel à l'opérateur.

Cela permet aux agents voisins d'exécuter les tâches de l'agent en question ou un ensemble de ses tâches.

Si les tâches de l'agent responsable de la ressource en panne sont toutes prises par les autres agents, cela résout le problème, dans le cas contraire, le système fonctionnera en mode dégradé (le système de production continuera à fonctionner sans la machine en panne) en attendant l'intervention de l'opérateur.

Algorithme proposé

Variables

Ap : agent dont la ressource est en panne.

Ap.tâche : l'ensemble des tâches de l'agent Ap.

Algorithme

```
DEBUT
    Ap lance processus calcul de chemin3 ;
    Si Ap.tâche=0 alors  // l'ensemble de tâches de l'agent dont la ressource est en panne
                         // sont toutes prises par les autres agents.
                    Début
                        Ap envoi rapport à l'opérateur ;
                    Fin
    Sinon  // il reste au moins une tâche qui n'est pas prise par les autres agents.
                    Début
                        Ap fait appel à l'opérateur ;
                    Fin
    Fsi
FIN.
```

Le diagramme d'activité correspondant (figure 3.7) est :

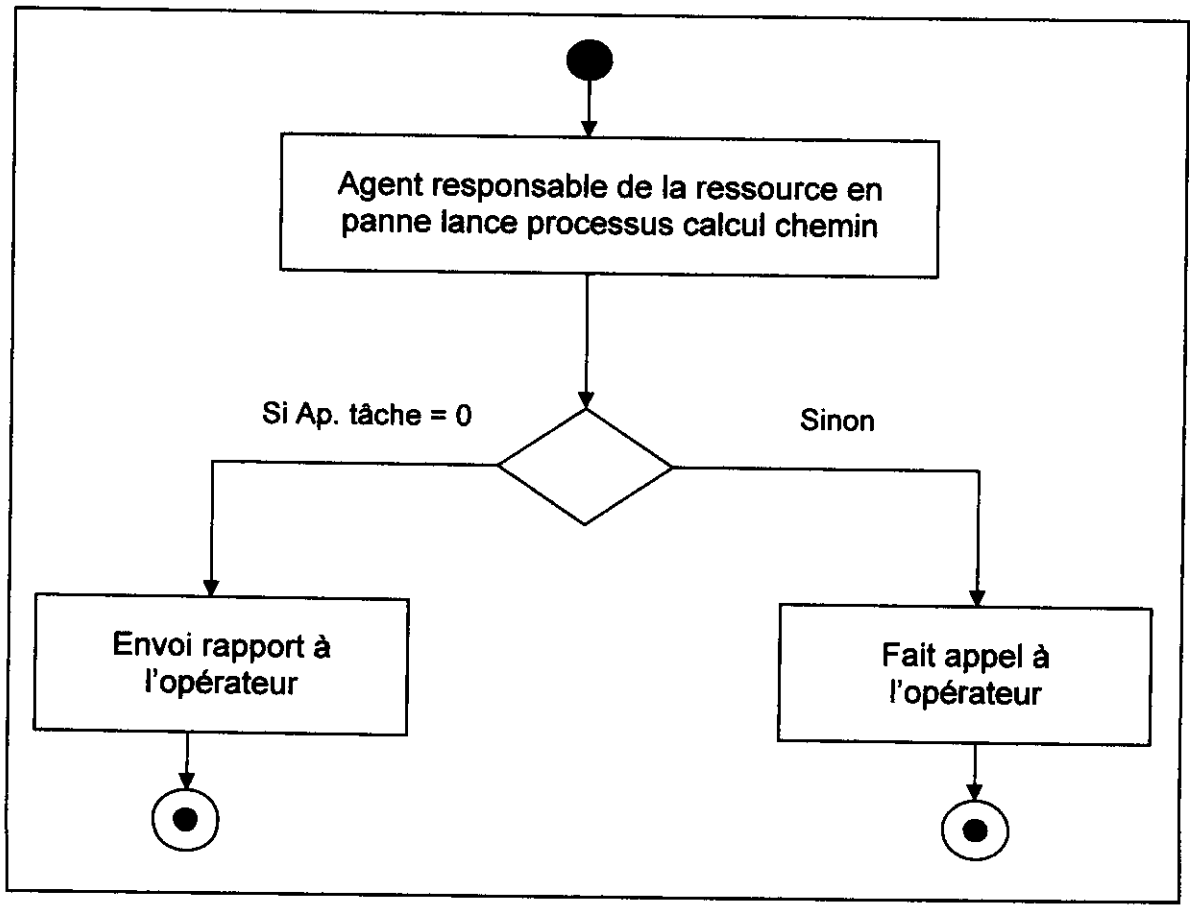


Figure 3.7. Diagramme d'activité de la panne ressource.

2.4.2.2. Panne agent ARP

Il existe deux cas :

1. ***L'agent ARP en panne est le seul voisin d'un autre agent ARP :***

La figure 3.8 nous montre le cas où l'agent ARP en panne n'a qu'un seul voisin, nous trouvons ici deux sous cas :

- ✓ L'agent voisin peut exécuter tout seul l'ensemble des tâches laissées par l'agent en panne. Dans ce cas l'agent voisin a le privilège de lancer le processus d'Election. L'agent leader prendra en charge la suppression de l'agent en panne, après l'avoir supprimé, l'agent voisin lance le processus de calcul de position.
- ✓ L'agent voisin ne peut exécuter tout seul l'ensemble des tâches laissé par l'agent en panne, dans ce cas il doit chercher un autre agent qui pourra l'exécuter, en lançant le processus de calcul de chemin:
  - Si l'agent voisin trouve celui qui pourra exécuter l'ensemble des tâches, alors ce dernier déclenche le processus d'élection (l'agent leader supprime l'agent en panne) et le processus de calcul de position.
  - Sinon l'agent voisin déclenche le processus d'élection, et l'agent leader supprime l'agent en panne et crée un autre agent.

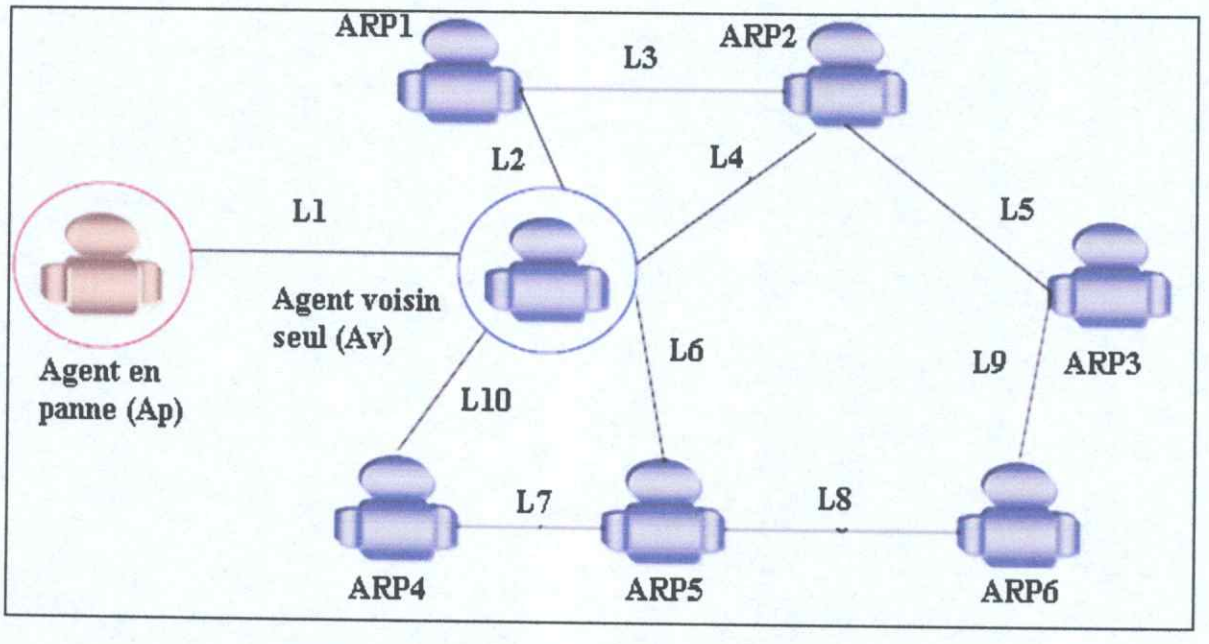


Figure 3.8. Un seul agent voisin d'un agent ARP en panne.

2. *L'agent ARP en panne est voisin de plusieurs autres agents :*

Dans ce cas nous avons un agent en panne mais relié avec plus d'un agent, comme le montre la figure 3.9, le système peut se trouver dans deux cas différents :

- Chaque agent peut exécuter tout seul une partie de l'ensemble des tâches de l'agent en panne : tous les voisins directs sont capables de récupérer un sous ensemble de tâches de l'ensemble des taches de l'agent en panne. Ces voisins lancent le processus d'Election (le leader supprime l'agent en panne), puis lancent le processus de calcul de position.
- Si aucun agent ne peut exécuter tout seul un sous-ensemble de tâches de l'ensemble des tâches de l'agent en panne :

Tous les agents voisins de l'agent en panne cherchent d'autres agents voisins en lançant le processus de calcul de chemin.

1) Si les agents voisins ne trouvent aucun de ces voisins capables d'exécuter les tâches de l'agent en panne alors : ils lancent le processus d'Election (le leader supprime l'agent en panne et crée un nouvel agent a_n qui sera relié à tous les anciens voisins de l'agent en panne).
2) Sinon (tous les voisins peuvent prendre un sous-ensemble des tâches)

- ✓ le Processus d'Election est lancé (le leader supprime l'agent en panne car il est remplacé par ses voisins ou par les voisins de ses voisins).
- ✓ Processus de calcul de position.

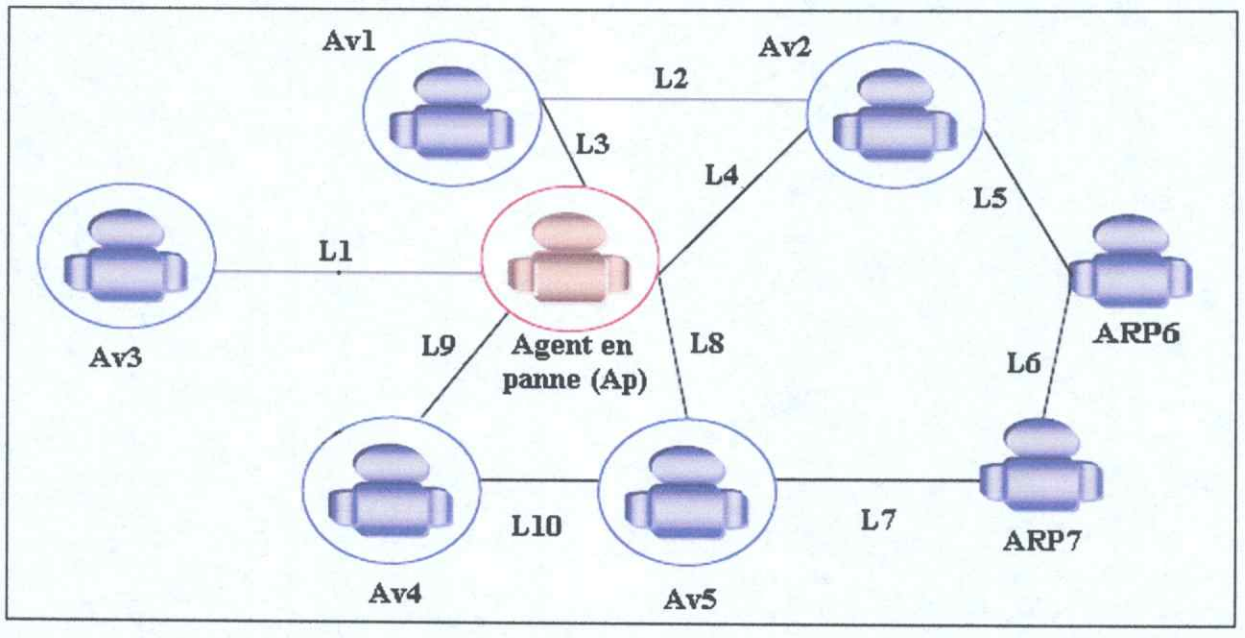


Figure 3.9. Plusieurs agents voisins d'un agent ARP en panne.

Algorithme proposé

Variables

Ap : agent en panne.

Av : agent voisin.

Ap.tâche : l'ensemble des tâches de l'agent Ap.

NbrAv : entier, nombre d'agents voisins directs de Ap.

Na : nouveau agent.

Aleader : agent leader.

Na.tâche : l'ensemble des tâches de l'agent Na.

EnsAv : ensemble des voisins directs de Ap.

Algorithme

```
DEBUT
    Si NbrAv=1 alors
        Début
            Si Av peut exécuter tout seul l'ensemble des tâches de Ap  alors
                Début
                    Av lance le processus d'élection ;
                    Aleader supprime Ap ;
                    Av lance le processus calcul de position ;
                    Aleader crée Na ;
                    Aleader envoi un rapport à l'opérateur ;
                Fin
            Sinon
                    Début
                        Av lance le processus calcul de chemin ;
                        Si Ap.tâche=0 alors // toutes les tâches ont été prise en compte
                                            // par les agents voisins d'Av.
                            Début
                                Av lance le processus d'élection ;
                                Aleader supprime Ap ;
                                Av lance le processus calcul de position ;
                                Aleader crée Na ;
                                Aleader envoi un rapport à l'opérateur ;
                            Fin
                        Sinon // il reste des tâches qui ne sont pas prises par les voisins.
                                Début
                                    Av lance le processus d'élection ;
                                    Aleader supprime Ap ;
                                    Aleader crée Na ;
                                    Na.tâche := Ap.tâche ;
                                Fin
                        Fsi
                          Na lance le processus calcul de position ;
                          Na lance le processus signal de vie ;
                    Fin
            Fsi
        Fin
    Sinon    // NbrAv>1
            Début
                Si Ap.tâche=0 alors  // toutes les tâches ont été prise par les agents voisins.
                        Début
                            EnsAv lancent le processus d'élection ;
                            Aleader supprime Ap ;
                            Aleader crée Na ;
```

```
                                    Aleader envoi un rapport à l'opérateur ;
                                Fin
                            Sinon
                                Début
                                    EnsAv lancent le processus calcul de chemin ;
                                    Si Ap.tâche=0 alors
                                            Début
                                                EnsAv lancent le processus d'élection ;
                                                Aleader supprime Ap ;
                                                Aleader crée Na ;
                                                Aleader envoi un rapport à l'opérateur ;
                                            Fin
                                    Sinon
                                            Début
                                                EnsAv lancent le processus d'élection ;
                                                Aleader supprime Ap ;
                                                Aleader crée Na ;
                                                Na.tâche := Ap.tâche ;
                                                Aleader envoi un rapport à l'opérateur ;
                                            Fin
                                    Fsi
                                    Na lance le processus calcul de position ;
                                    Na lance le processus signal de vie ;
                                Fin
                            Fsi
                        Fin
                    Fsi
                FIN.
```

Le diagramme d'activité correspondant (figure 3.10) est :

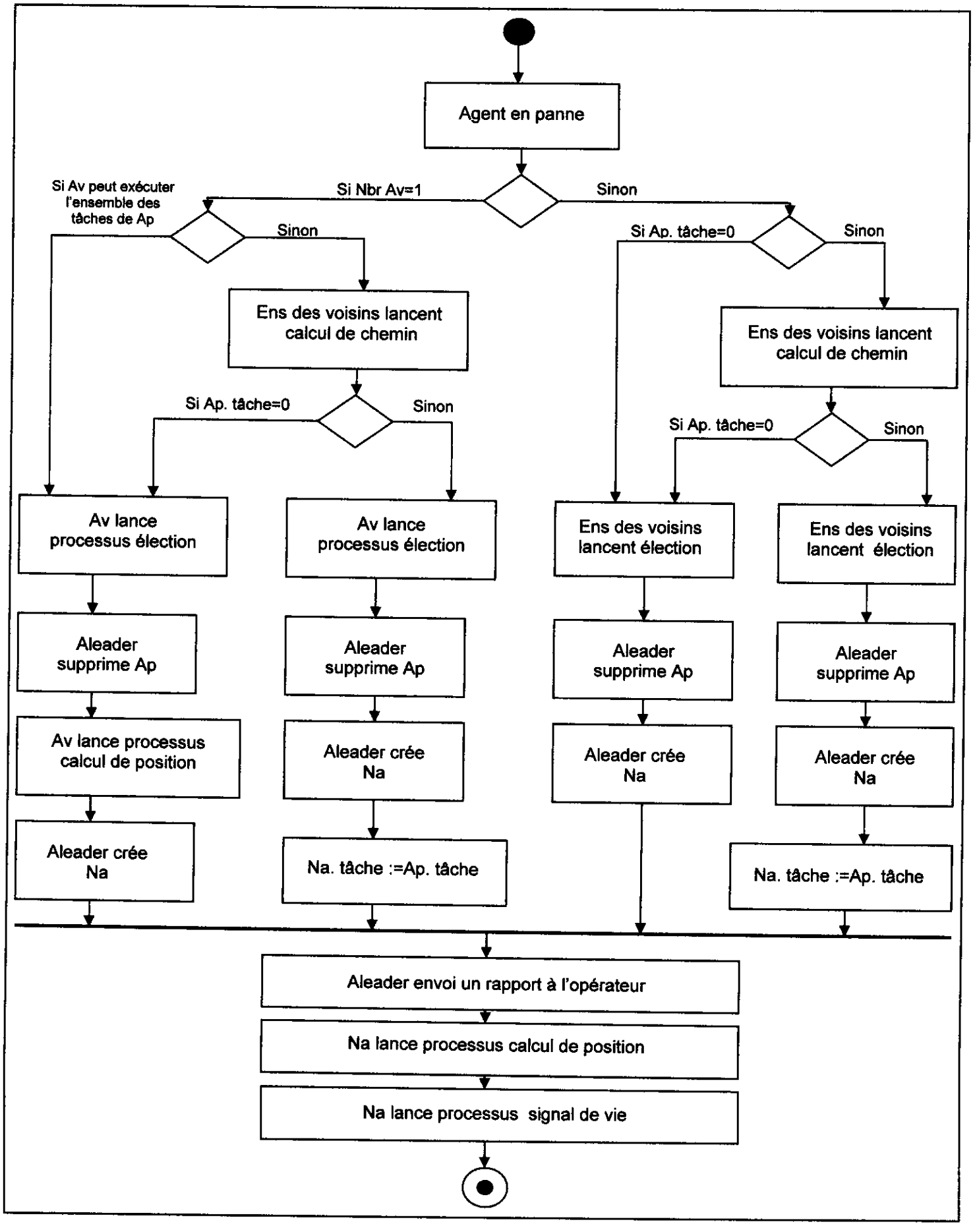

Agent en panne
Si Nbr Av=1
Sinon
Si Av peut exécuter l'ensemble des tâches de Ap
Sinon
Si Ap. tâche=0
Sinon
Ens des voisins lancent calcul de chemin
Ens des voisins lancent calcul de chemin
Si Ap. tâche=0
Sinon
Si Ap. tâche=0
Sinon
Av lance processus élection
Av lance processus élection
Ens des voisins lancent élection
Ens des voisins lancent élection
Aleader supprime Ap
Aleader supprime Ap
Aleader supprime Ap
Aleader supprime Ap
Av lance processus calcul de position
Aleader crée Na
Aleader crée Na
Aleader crée Na
Aleader crée Na
Na. tâche :=Ap. tâche
Na. tâche :=Ap. tâche
Aleader envoi un rapport à l'opérateur
Na lance processus calcul de position
Na lance processus signal de vie

Figure 3.10. Diagramme d'activité de la panne agent ARP.

2.4.2.3. Lien de communication perdu

Lorsqu'un lien de communication est perdu, il y'a deux agents qui sont perturbés, l'un d'eux cherche à rejoindre l'autre par un autre chemin en demandant dans son voisinage quel sont les agents qui peuvent servir d'intermédiaire pour arriver à l'agent en question.

Nous trouvons ici trois cas :

- Agents isolés ou système décomposé en deux sous système.
- Agent isolé voisin d'un agent appartenant à un sous système.
- Les deux agents appartiennent au même sous système.

1. ***Agent ARP isolé ou système décomposé en deux sous systèmes :***

   Dans ce cas (voire figure 3.11), lors de la destruction du lien de communication entre deux agents, le système multi-agent est décomposé en deux sous-systèmes et il n'existe pas un chemin reliant les deux agents.

   L'idée est que l'un des agents change son adresse du port puis envois un signal de vie à l'autre agent.

   1. Si l'autre agent répond par un signal de vie, pour dire qu'il est d'accord pour cette nouvelle adresse, alors les deux sous-systèmes seront reliés à nouveau et le SdP redeviendra fonctionnel.
   2. Sinon, l'autre agent ne répond pas par un signal de vie à toutes les adresses de port envoyées, alors la panne est physique et l'intervention de l'opérateur est nécessaire.

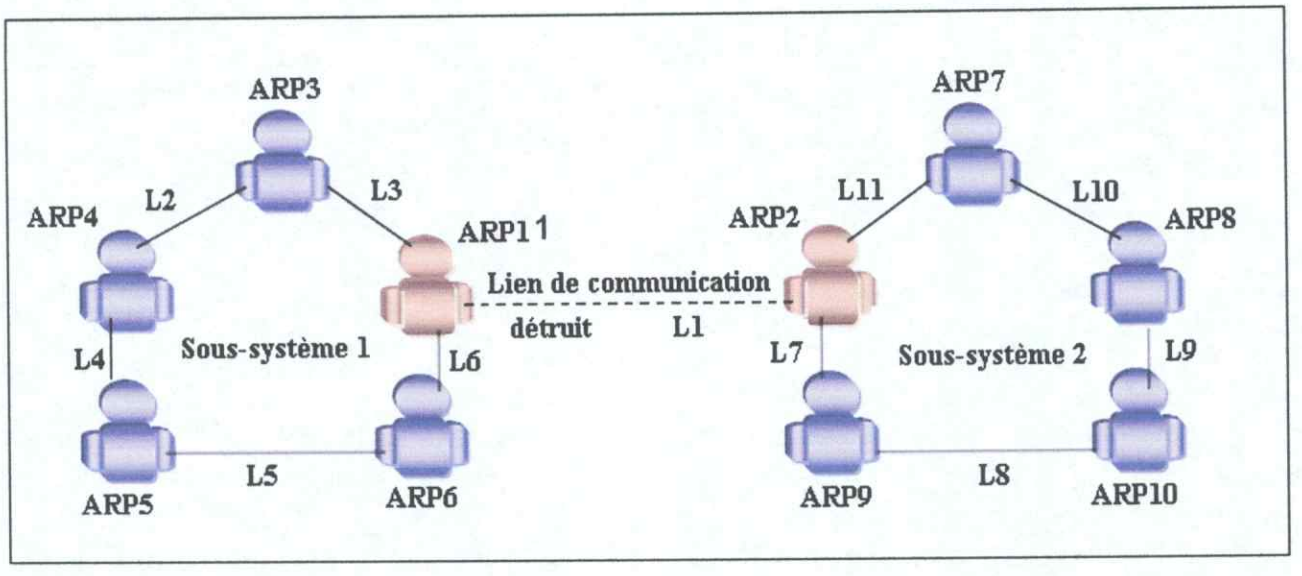


Figure 3.11. Système décomposé en deux sous-systèmes.

2. ***Agent ARP isolé voisin d'un agent ARP appartenant à un sous système :***

Lors de destruction d'un lien de communication entre deux agents (figure 3.12) et l'un d'eux deviendra isolé, l'agent isolé change son adresse du port et informe son voisin en lui envoyant un signal de vie pour marquer sa présence.

1. Si l'autre agent répond par un signal de vie, pour dire qu'il est d'accord pour cette nouvelle adresse, l'agent isolé réintégrera le système multi-agents.
2. Sinon, l'autre agent ne répond pas par un signal de vie à toutes les adresses de port envoyées, l'opérateur interviendra, car la panne est physique.

Remarque :

Contrairement au cas précèdent, un seul agent peut changer son adresse de port, c'est l'agent isolé.

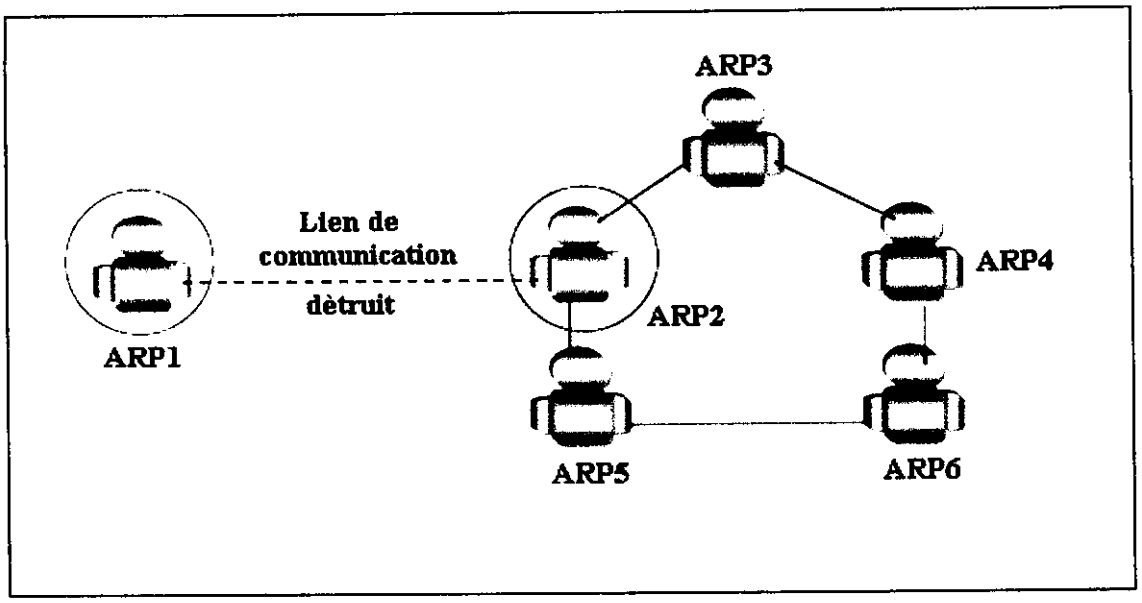


Figure 3.12. Un agent est isolé suite à la destruction de son lien de communication.

3. Les deux agents ARP appartiennent au même sous système :

Dans ce cas (figure 3.13), lors de la destruction d'un lien de communication entre deux agents, ces derniers peuvent trouver un autre chemin les reliant.

Chacun d'eux lance le processus calcul de chemin, puis lance le processus de calcul de position.

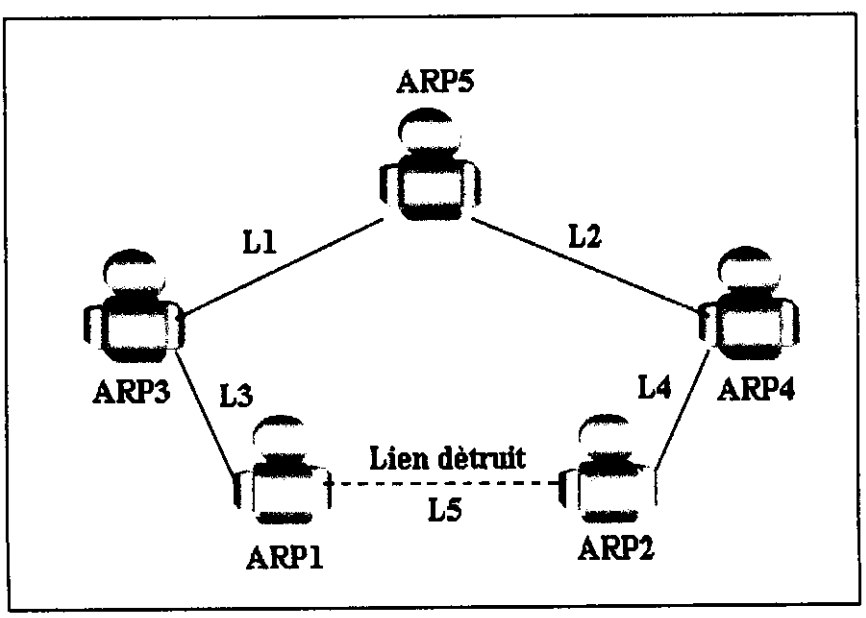


Figure 3.13. Les agents ARP appartiennent au même sous-système.

Algorithme proposé

Variables

Ap1, Ap2 : deux agents qui ont le lien de communication détruit.

Algorithme

```
DEBUT
    Si Ap1 et Ap2 appartiennent à deux sous-systèmes alors
        Début
            Ap1 et/ou  Ap2 change son adresse de port ;
            Ap1 (éventuellement Ap2)  envoi un signal de vie à Ap2 (éventuellement
                                                                        Ap1) ;
            Ap2 (éventuellement Ap1) répond au signal de vie ;
        Fin
    Si Ap1 n'a qu'un seul voisin Ap2 alors
        Début
            Ap1 change adresse de port ;
            Ap1 envoi un signal de vie à Ap2 ;
            Ap2 répond au signal de vie ;
        Fin
    Si Ap1 et Ap2 appartiennent au même système alors
        Début
            Ap1 (éventuellement Ap1) lance le processus calcul de chemin ;
            Ap1 (éventuellement Ap1) lance le processus calcul de position ;
        Fin
    Fsi
FIN.
```

Le diagramme d'activité correspondant (figure 3.14) est :

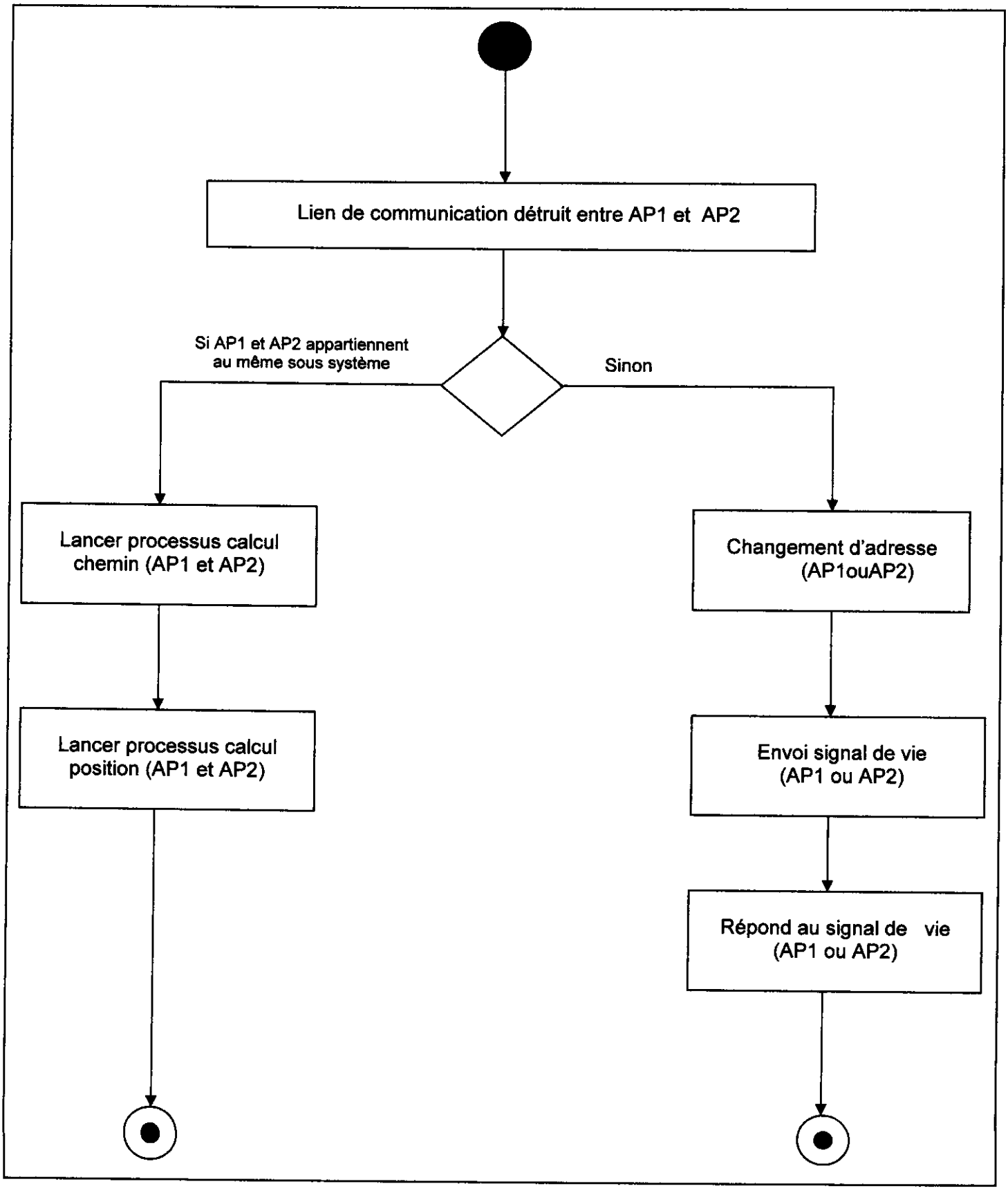


Figure 3.14. Diagramme d'activité de la panne du lien de communication.

3. Conclusion

Dans ce chapitre nous avons expliqué le fonctionnement d'un système de production, pour ensuite proposer une approche multi-agents appropriée. Nous avons donné les caractéristiques de chaque agent du système, ainsi que les fonctions qu'il doit remplir.

Nous avons parlé de la communication inter-agents, puis nous avons proposé des algorithmes, expliqués et suivi de schémas et diagrammes, se reposant sur les processus constituant le protocole auto-organisable.

CHAPITRE IV

Implémentation

1. Introduction

Après avoir achevé l'étude conceptuelle de l'approche multi-agents, nous entamons la dernière phase de notre projet, qui consiste à simuler son comportement et les différents cas qui le constituent.
Ce protocole est réalisé avec le langage LOGO à l'aide De la plate forme multi-agents NetLogo.

2. Environnement de développement

2.1. La plate forme NetLogo

NetLogo est un environnement de modelage programmable pour simuler des phénomènes naturels et sociaux. Il est adapté en particulier pour modeler des systèmes complexes qui se développent avec le temps. Les concepteurs peuvent donner des directives à des centaines ou milliers d'agents indépendants fonctionnant concurremment. Cela rend possible d'explorer le rapport entre le comportement micro-niveau d'individus et les patterns du macro-niveau qui émergent de l'interaction de plusieurs individus.
NetLogo laisse des étudiants ouvrir des simulations et "travailler" avec, en explorant leur comportement sous plusieurs conditions. C'est aussi un environnement autoritaire qui permet aux étudiants, les professeurs et les promoteurs des programmes scolaires de créer leurs propres modèles. NetLogo est assez simple que les étudiants et les professeurs puissent exécuter des simulations facilement ou même construire leurs propres modèles. Il est assez avancé pour servir comme un outil puissant pour les chercheurs dans beaucoup de domaines.
NetLogo a une documentation étendue et des travaux pratiques. Il vient aussi avec une bibliothèque des Modèles qui est une grande collection de simulations pré-écrites qui peuvent être utilisées et peuvent être modifiées. Ces simulations s'adressent à beaucoup de domaines: les sciences naturelles et sociales, y compris biologie et médecine, physique et chimie, mathématiques et informatique, industrie,

économie et psychologie sociale. Plusieurs autres modèles et programmes sont actuellement sous développement.

NetLogo peut propulser aussi une classe appelée "the participatory-simulation outil" appelé HubNet. À travers l'usage des réseaux d'ordinateurs ou appareils portatifs tels que les instruments calculateurs de Texas (TI-83+), chaque étudiant peut contrôler un agent dans une simulation.

NetLogo est la prochaine génération de la série de langues du modelage multi-agent qui ont commencé avec StarLogo. Il construit des fonctionnalités du produit StarLogoT et ajoute des nouveaux traits considérables et une langue redessinée et interface de l'utilisateur. NetLogo est écrit en Java, donc il peut courir sur toutes les plates-formes du majeur (Mac, Windows, Linux). Il est connu comme une application autonome.

Les modèles individuels peuvent être exécutés comme applets de Java à l'intérieur d'un navigateur web. [Net 06]

2.2. Le langage LOGO

Logo n'est pas un langage de programmation comme les autres. Capable d'enthousiasmer les plus jeunes sur les bancs de l'école, il fascine tout autant les adultes. Car, au-delà de la tortue graphique qui exécute sur l'écran d'étonnantes figures, se cache un vrai langage de programmation, riche, structuré et fonctionnel.

Les créateurs de Logo, Papert et Minski [Hal 06], ont souhaité créer un langage de programmation afin d'utiliser la puissance de l'outil informatique dans les tâches d'enseignement. Plus que tout autre langage, Logo a été conçu dans le but de démystifier les ordinateurs et la programmation. Tout adepte de Logo s'oppose naturellement à l'utilisation injustifiée des jargons et à toute tendance visant à faire de l'informatique un domaine à part. De ce fait, la simplicité de mise en œuvre de ce langage est étonnante : Quelques minutes suffisent pour en comprendre la logique et se lancer dans la création de projets personnels.

Pourtant, malgré cette simplicité apparente, Logo est un véritable langage de programmation qui permet d'aborder des concepts fondamentaux de l'informatique tels que la récursivité, le traitement de listes ou la pleine fonctionnalité. En cela, il se

rapproche de langages utilisés par les étudiants en informatique tels que Lisp ou Caml.

Le nom « Logo » vient du grec « *Logos* » qui signifie « *parole* », « *discours* ». Les premières versions de Logo permettaient essentiellement de manipuler des mots et des phrases, mais on se rendit rapidement compte que la manipulation de symboles ne suscitait pas un grand engouement.

C'est Seymour Papert qui imagina de se servir de ce langage pour tracer des graphiques [Pap 81].

Logo est un langage issu le Lisp. Comme lui, c'est un langage fonctionnel. Comme Lisp, c'est un langage interprété, ce qui permet une utilisation directe sans passer par une phase de compilation.

Le premier système Logo a fonctionné en 1970 au Massachusetts Institute of Technology (MIT), dans le laboratoire d'intelligence artificielle. L'ordinateur était un PDP 10 de la société *Digital Equipment Corporation*. Les périphériques se composaient de deux écrans, l'un réservé au texte, l'autre au graphisme. Par ailleurs, cet ordinateur commandait les déplacements d'un petit robot, la tortue de sol (en fait un petit chariot dont les mouvements étaient commandés par des instructions introduites sur le clavier d'un ordinateur), capable de laisser une trace de ses trajets grâce à un crayon dont le maniement était prévu dans le langage.

Le Logo s'adapte à une vaste gamme d'applications qui vont de la recherche sur l'intelligence artificielle à la conception d'applications graphiques et à l'enseignement au niveau préscolaire. On a même utilisé Logo dans l'enseignement donné aux handicapés mentaux, leur permettant de contrôler cet appareil puissant qu'est l'ordinateur.

Dans NetLogo, il y a trois types d'agents: tortues « turtles », pièces « patches », et l'observateur « observer ». Les tortues sont des agents qui se déplacent dans leur monde. Le monde est à deux dimensions et est divisé en une grille de pièces. Chaque pièce est un morceau carré de "terre" sur laquelle les tortues peuvent déplacer les « pièces ». L'observateur n'a pas d'emplacement, vous pouvez l'imaginer comme une entité externe observant le monde de tortues et pièces.

3. Description de l'application

Le but de l'application est de simuler le comportement d'un ensemble d'agents dans un système de production. Le système d'agents représenté dans l'application est formé de :

- Un ensemble d'agents de même type, c'est-à-dire agent ressource pilote, chaque agent dans le système peut être : initiateur d'un calcul, leader, remplaçant d'un autre agent, de plus, chaque agent est caractérisé, dans l'interface, par le nombre de tâches qui lui y sont affecté et sa position par rapport à l'agent initiateur. Les agents sont répartis dans le système selon une architecture initiale.
- Un ensemble de liens de communication qui relient les agents.

Dans ce qui suit, nous présentons les différents cas de pannes que peut un système de production rencontrer, ainsi que les traitement et solutions proposés.

4. Les boutons et leurs fonctions

Comme la figure 4.1 le montre, les boutons sont répartis en trois catégories :

- Boutons d'installations.
- Boutons de communication.
- Boutons des simulations des pannes.

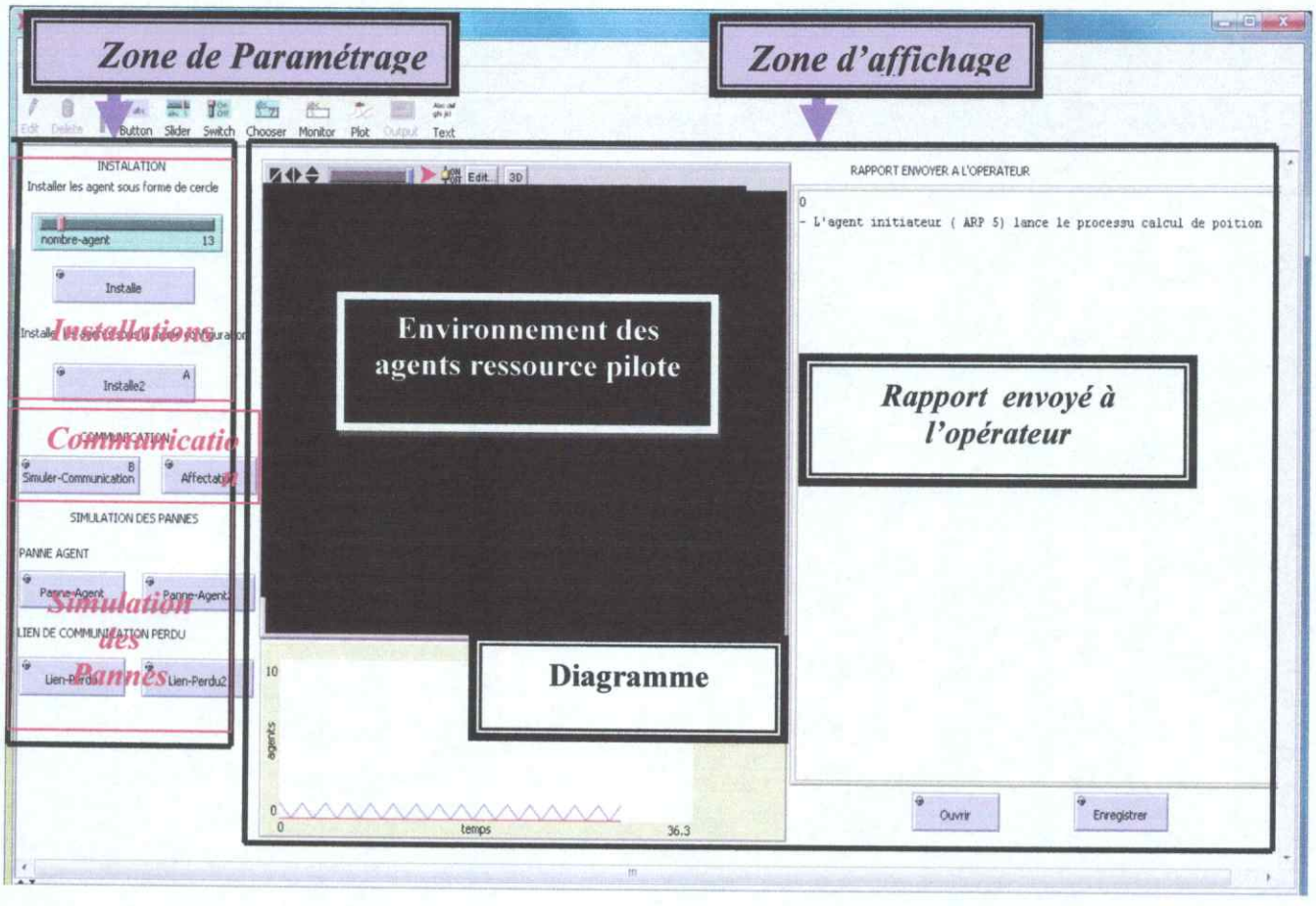
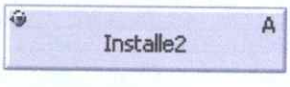


Figure 4.1. Interface initiale de l'application.

4.1. Boutons d'installations

Les boutons d'installations permettent l'installation du système.

- ***Le bouton " Installe2 " :***



Bouton

Le bouton " installe2 " donne toujours la même configuration (figure 4.2).

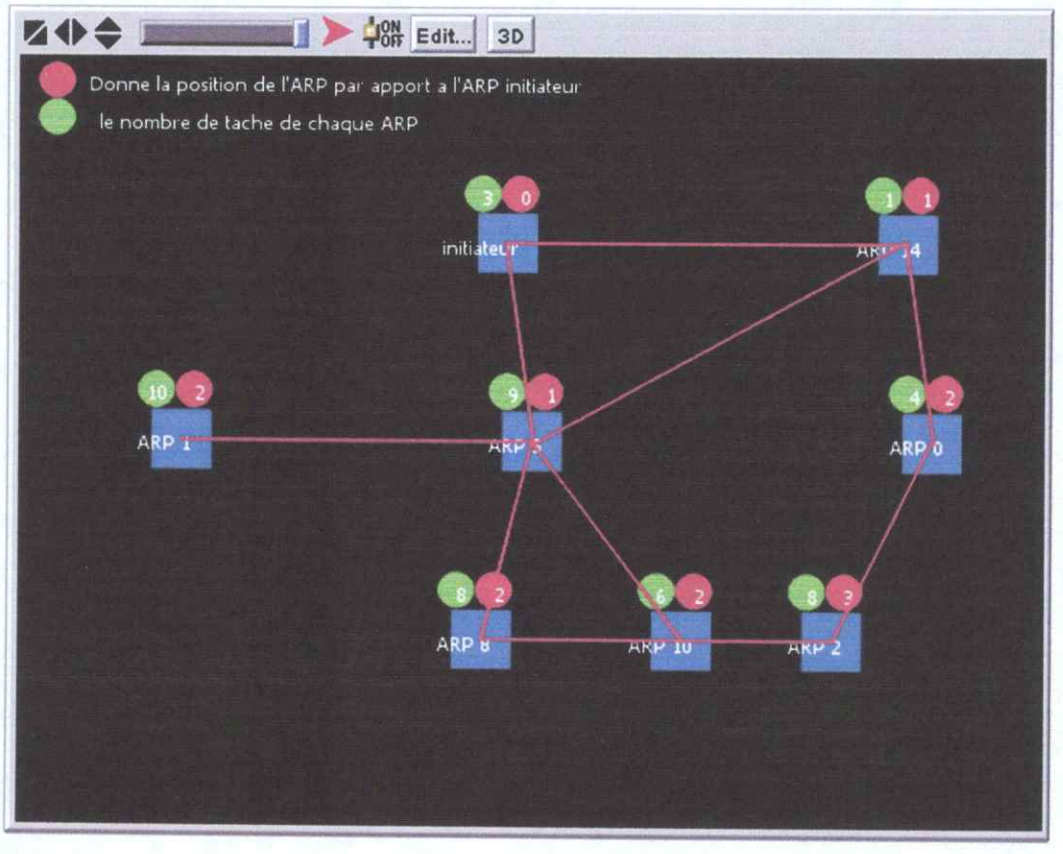
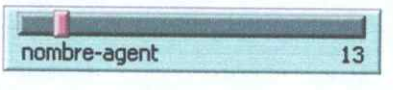


Figure 4.2. Interface résultante du bouton Installe2.

➢ ***Le bouton " Installe " et le glisseur***

Bouton



Glisseur

Le bouton "Installe " donne une configuration initiale aléatoire du système sous forme d'un cercle (Figure 4.3). Un glisseur permet de déterminer le nombre d'agents dans le système.

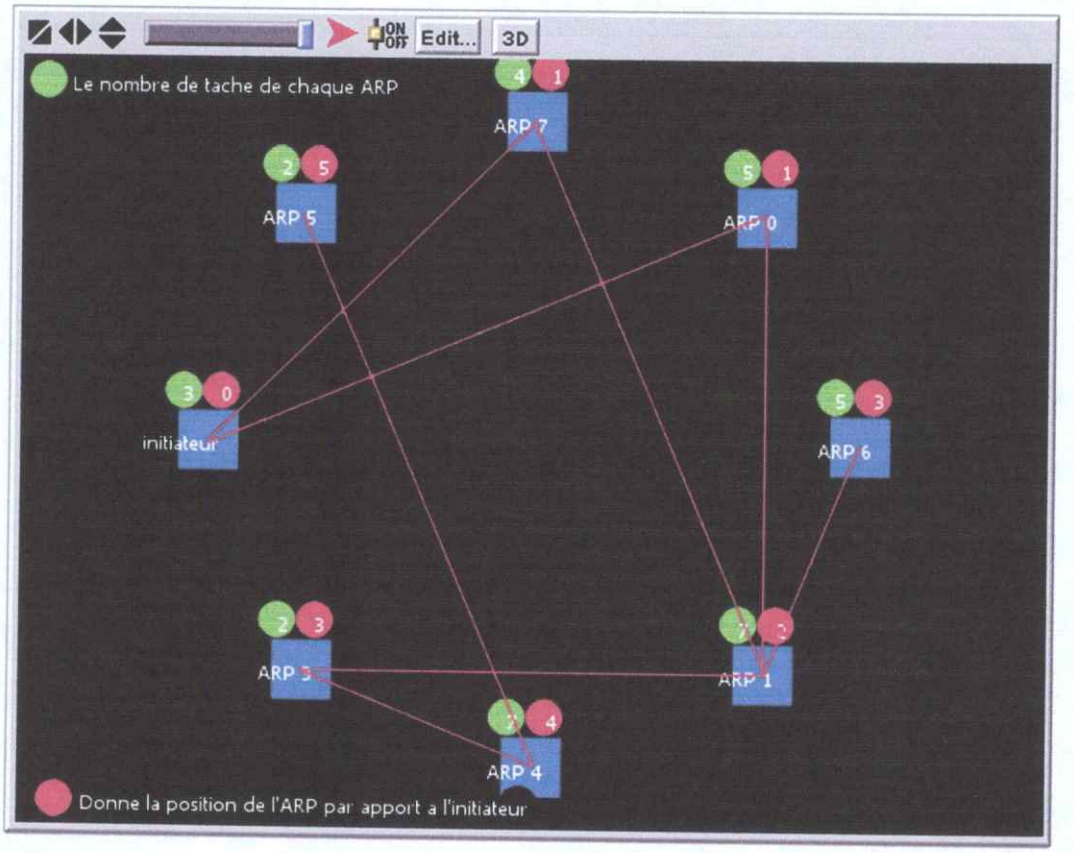


Figure 4.3. Interface résultante du bouton installe.

Après avoir installé le système à l'aide d'un des deux boutons "Installe" ou "Installe2", un message apparait dans la zone d'affichage. Ce dernier nous informe que l'agent " Initiateur " lance le processus "calcul de position" (Figure 4.4) pour permettre aux différents agents ressource pilote de connaitre leur position par rapport à l'agent " Initiateur ", ainsi que les tâches de leurs voisins.

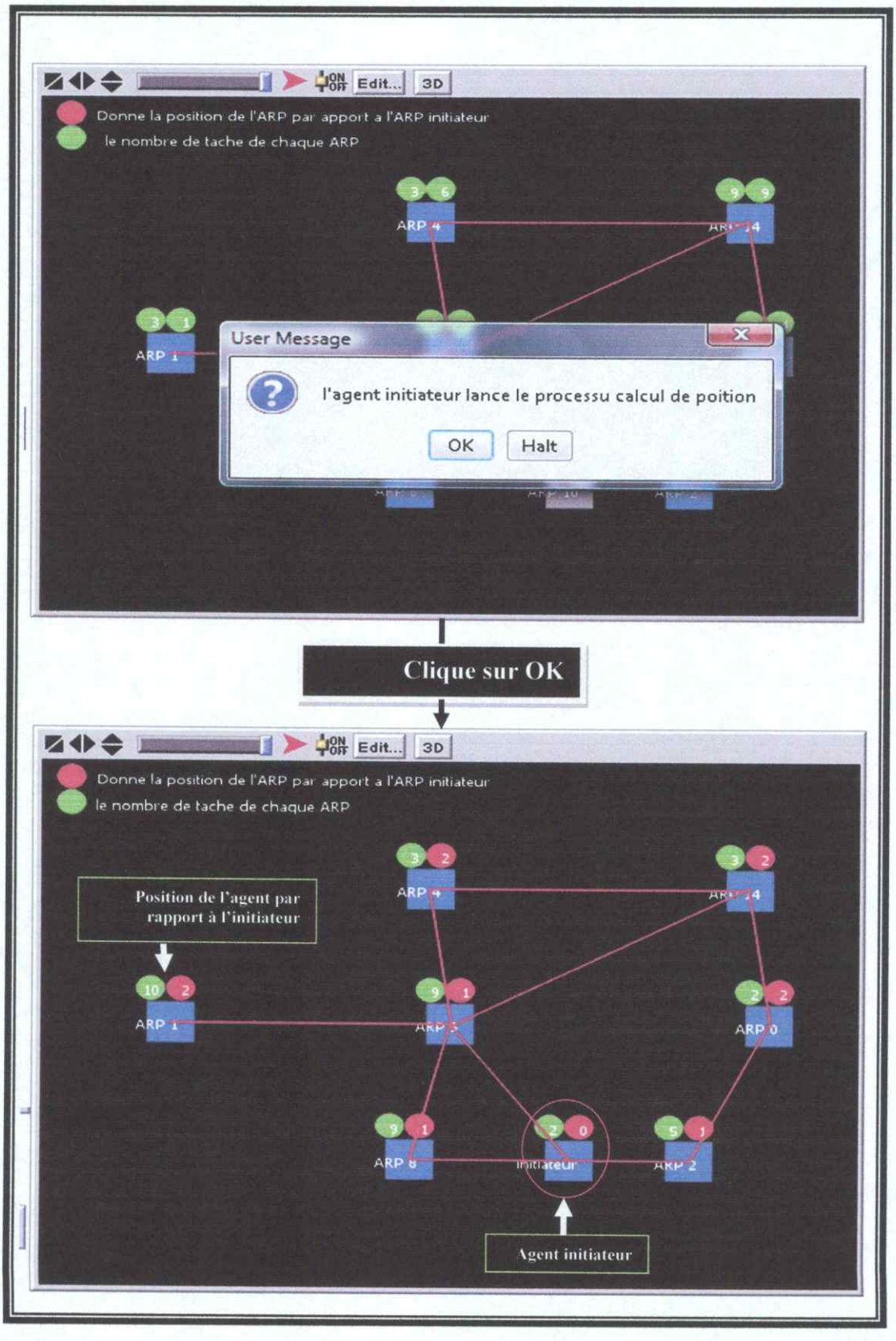


Figure 4.4. Interface calcul de position.

4.2. Boutons de communication

Les boutons de communication permettent de simuler une communication entre les différents agents du système.

➢ ***Le bouton " Simuler-communication "***

Le bouton " Simuler-Communication " comme son nom l'indique, permet de faire une petite simulation de la communication entre les agents, ainsi que l'exécution de leurs tâches.

Lorsque la couleur d'un agent est rouge, cela veut dire qu'il a terminé l'exécution de ses taches (Figure 4.5).

Lorsque la couleur d'un lien de communication est orange, cela veut dire que ce lien transmet une communication d'un agent à un autre. (Figure 4.5).

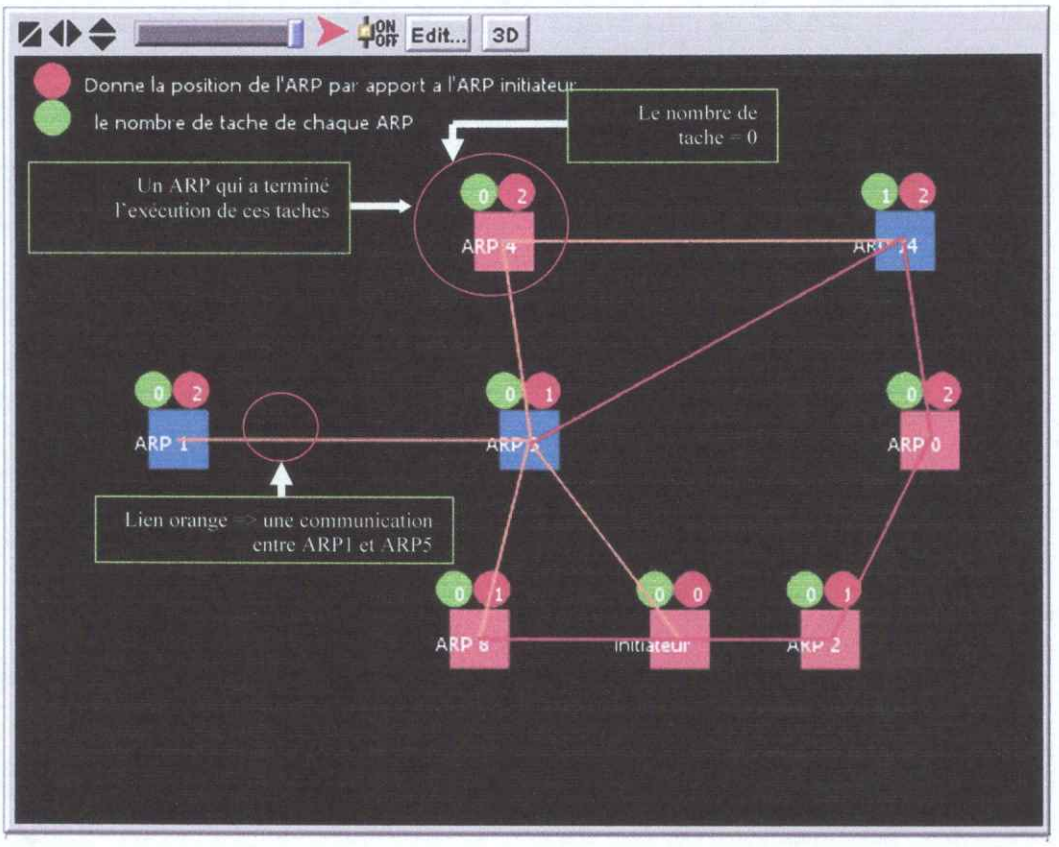
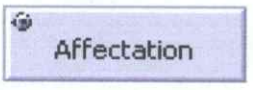


Figure 4.5. Interface résultante du bouton Simuler-communication.

➢ ***Le bouton " Affectation "***



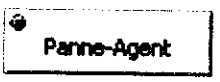
Le bouton " Affectation " permet de réaffecter, d'une façon aléatoire, de nouvelles tâches pour chaque agent.

4.3. Boutons de simulation de pannes

Les boutons de simulation des pannes permettent de simuler quelques pannes et de montrer comment les agents ressources du système changent leurs comportements afin de gérer la panne.

4.3.1. Simulation des pannes des agents :

- ***Le Bouton " Panne-agent "***



Le bouton " Panne-Agent " simule le cas où un agent est en panne, et que ses tâches sont prises en charge par l'un de ses voisin direct ou indirect qui lance le processus d'élection d'un agent leader qui supprime l'agent en panne.
Lorsque la couleur d'un agent est marron, cela signifie qu'il est en panne. L'agent jaune est le voisin qui prend en charge les tâches de l'agent en panne dans le cas où il est un voisin direct. Dans le cas d'un voisin indirect il prend la couleur orange. L'agent de couleur magenta est l'agent leader (Figure 4.6).

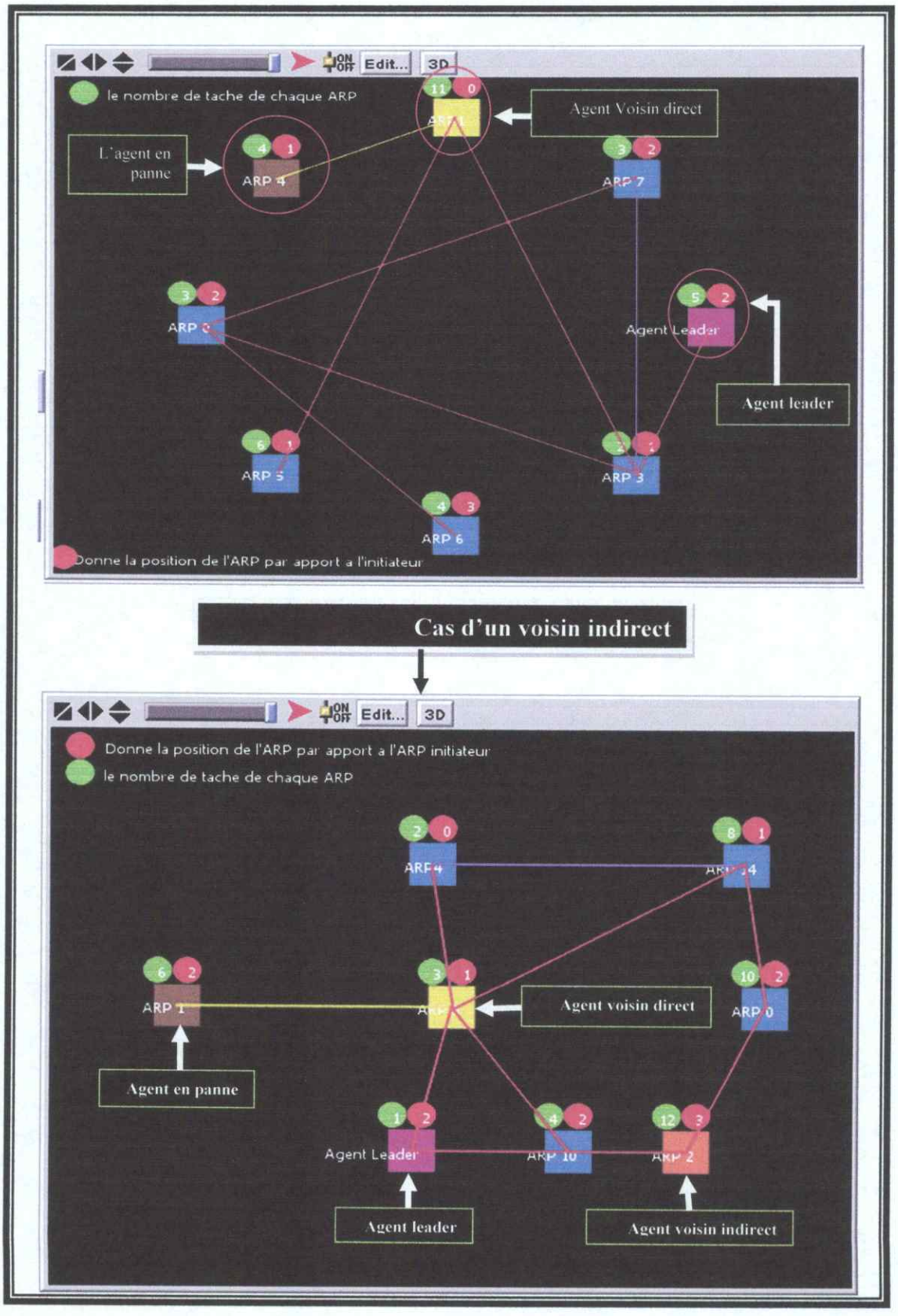
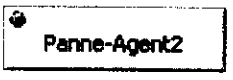


Figure 4.6 Panne-agent.

➢ ***Le bouton " Panne-Agent2 "***



Le bouton "Panne-Agent2 " simule le cas d'un agent en panne (couleur marron) et aucun agent du système ne peut prendre en charge ces tâches. Dans ce cas des agents voisins lancent le processus d'élection d'un agent leader (couleur magenta) pour la suppression de l'agent en panne et son remplacement par un nouvel agent (Figure 4.8).

Un rapport détaillé est envoyé à l'opérateur sur l'état du système (agent en panne, élection d'un agent leader, nouveaux agents,...) (Figure 4.7)

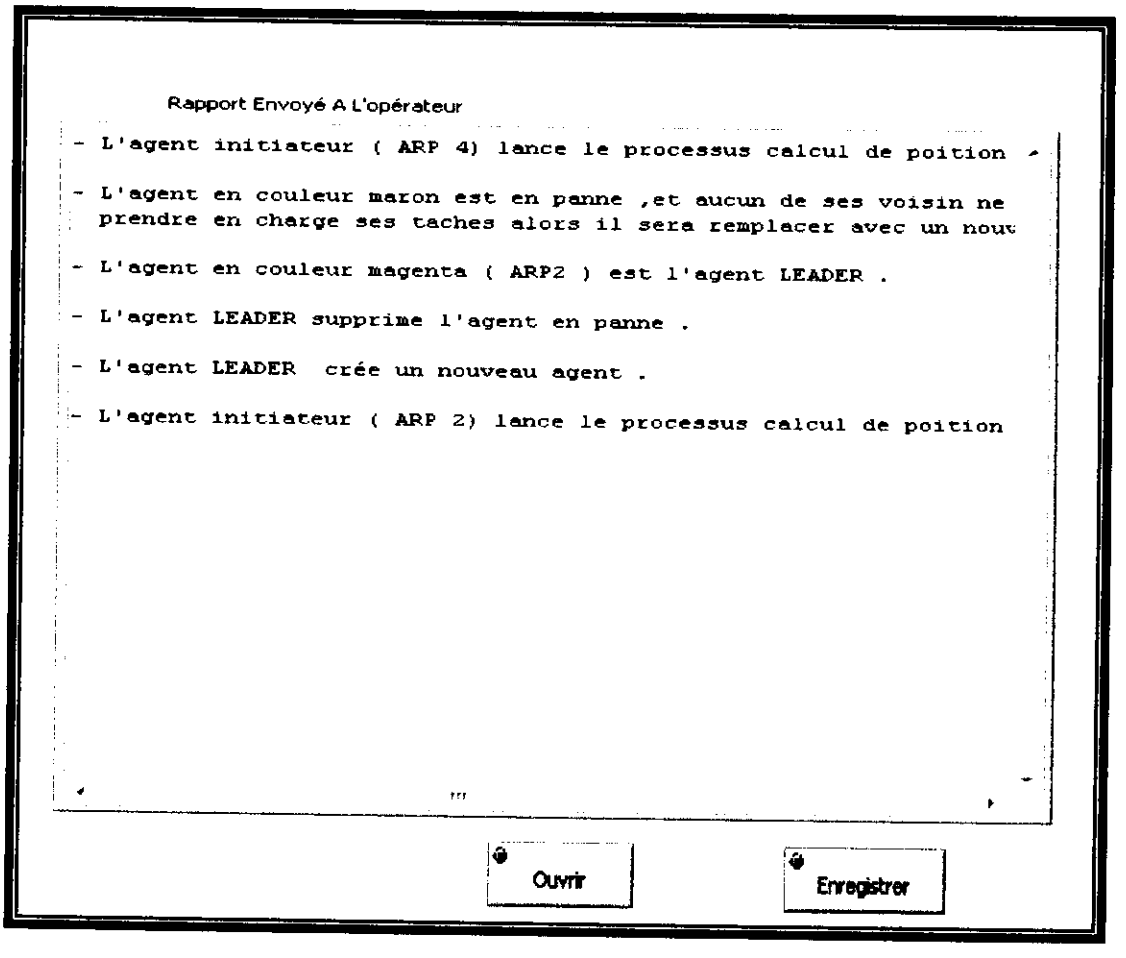


Figure 4.7. Rapport envoyé à l'opérateur.

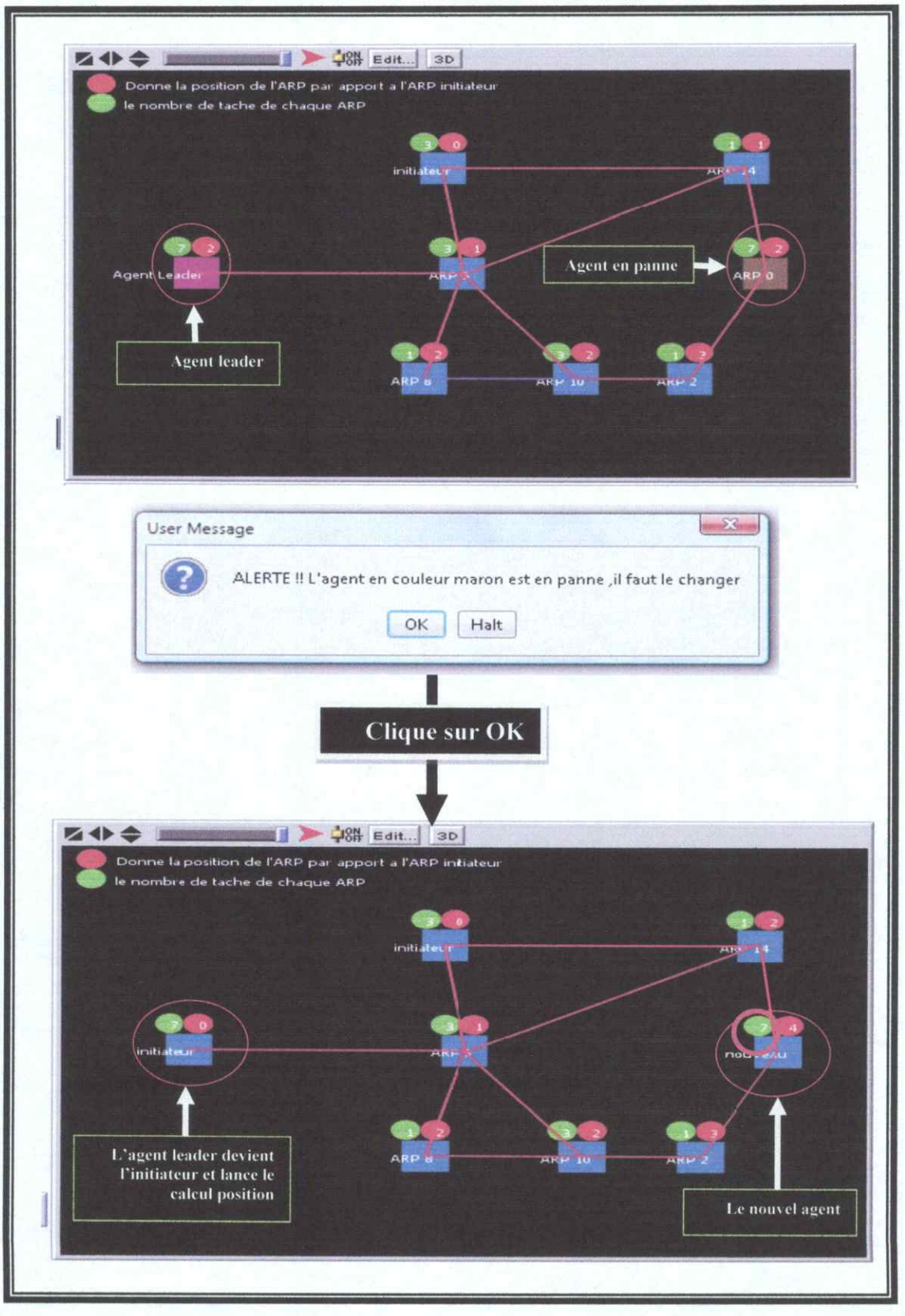


Figure 4.8. Panne-agent2.

4.3.2. Simulation des pannes des liens de communication :

- **Le bouton " Lien-Perdu "**

Le bouton " Lien-Perdu " simule le cas d'un agent isolé (couleur jaune) suite à une destruction d'un lien de communication (couleur jaune). Dans ce cas, l'agent isolé change son adresse et l'envoi à son voisin (Figure 4.9).

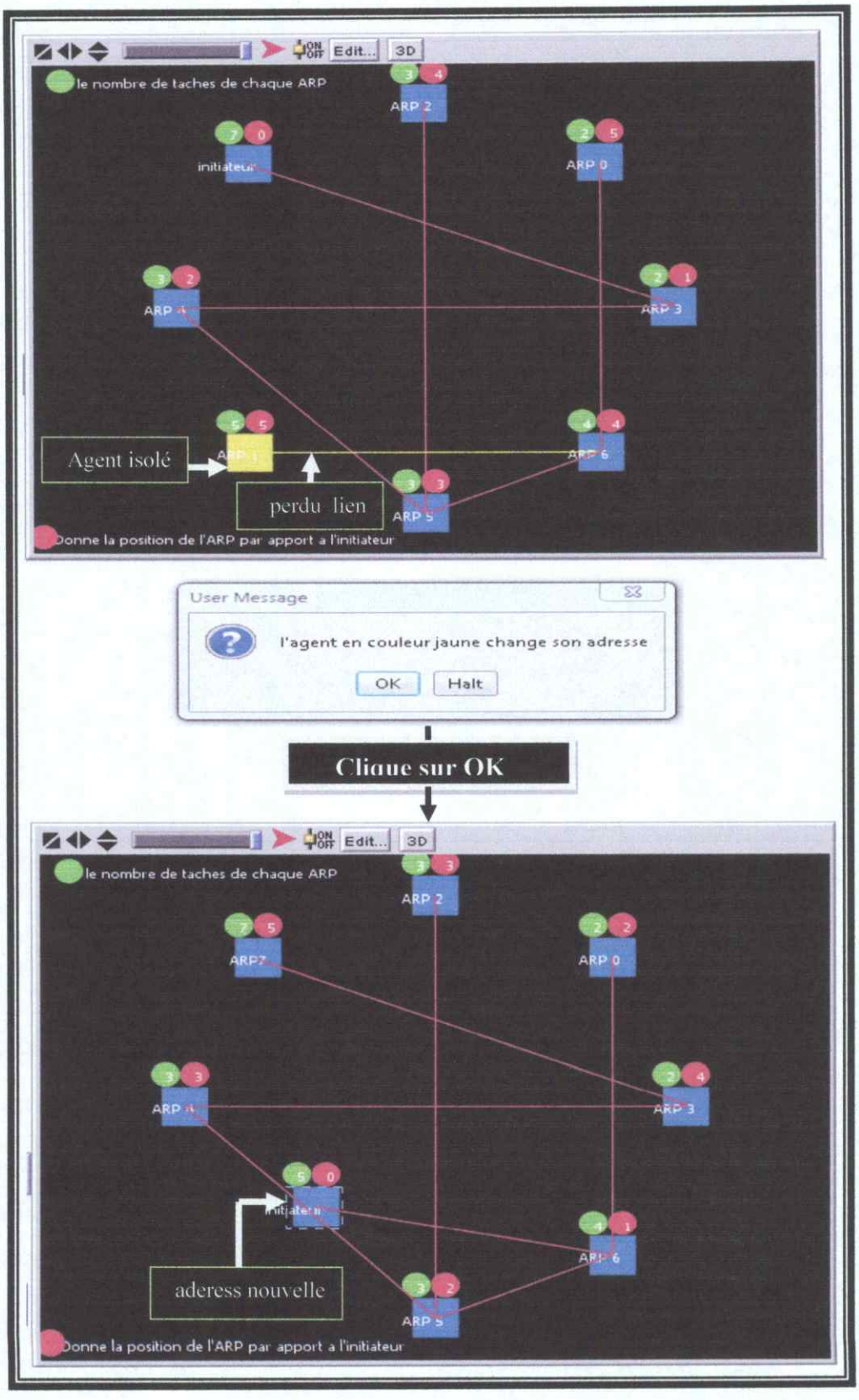


Figure 4.9. Lien perdu.

➢ **Le bouton " Lien-perdu2 "**

Dans ce cas (Figure 4.10), lors de la destruction d'un lien de communication entre deux agents, ces derniers peuvent trouver un autre chemin les reliant. Dans notre application la couleur du chemin est verte.

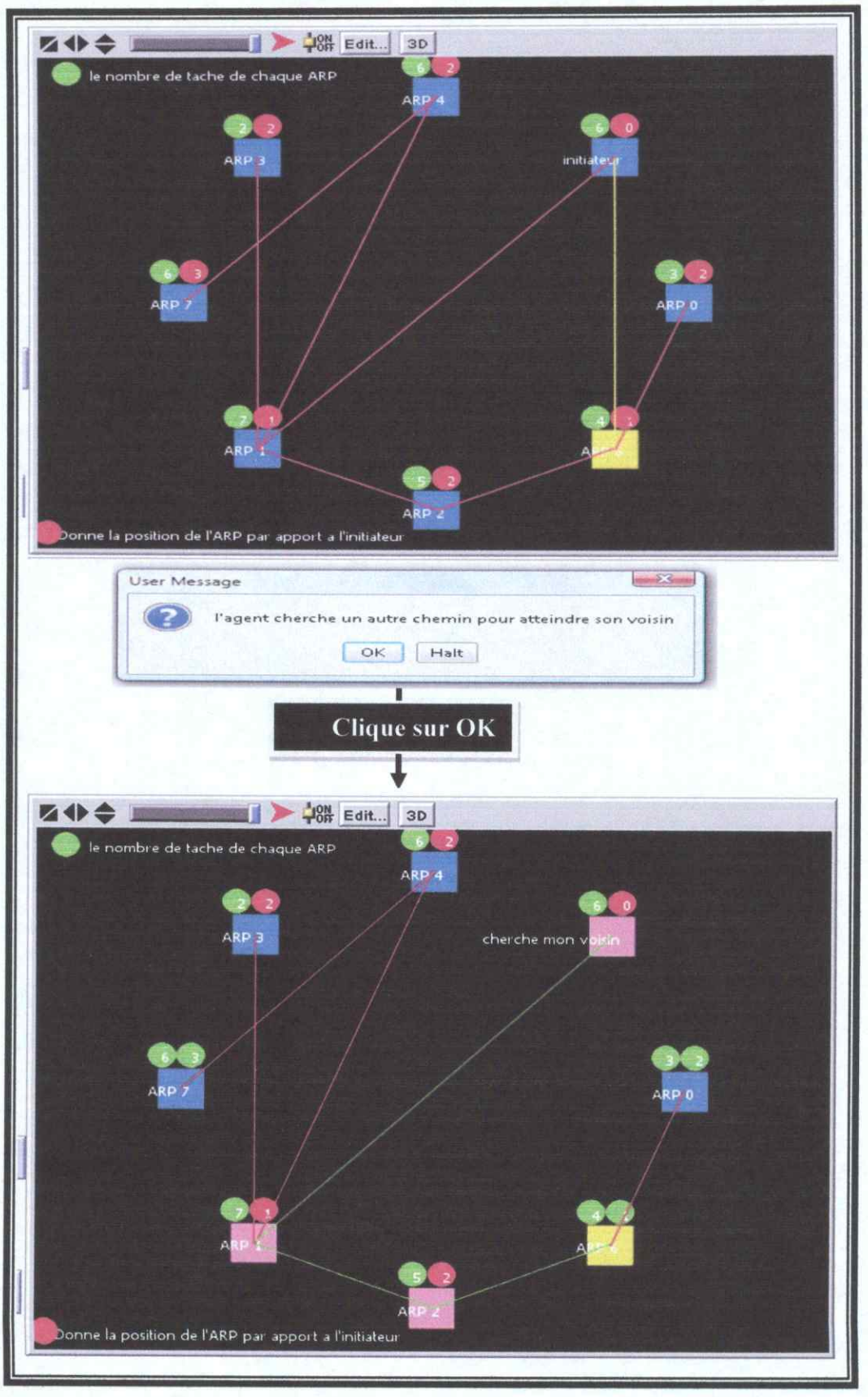


Figure 4.10. Lien perdu2.

5. Conclusion

Dans ce chapitre, nous avons présenté une simulation des différents traitements des pannes qui peuvent survenir.

Les deux cas que nous avons présenté dans ce chapitre sont :

- Agent en panne.
- Lien de communication détruit.

Ces cas regroupent les comportements les plus essentiels du protocole auto-organisable :

- Calcul de position.
- Election (le leader supprime l'agent en panne).
- Calcul de chemin (trouver le voisin perdu).
- Signal de vie.

Conclusion générale et perspectives

Conclusion générale

Dans ce document, nous avons proposé une approche multi-agents pour la surveillance du fonctionnement d'un système de production. Le système multi-agents proposé est animé par un protocole de communication qui est un protocole auto-organisable, ce protocole présente les avantages suivants :

- Il assure la décentralisation du contrôle tel que chaque agent peut prendre des décisions pour interagir avec ses voisins.
- Une conséquence logique de l'avantage précédent est que les agents peuvent pratiquer plusieurs fonctions durant leur vie dont la principale est celle de Leader, sans pour autant que celle-ci soit spécifique à un seul agent.
- Il permet l'émergence de nouvelles structures d'organisation grâce aux interactions inter agents pour faire face aux changements non souhaitables, et c'est ce qui offre au système une tolérance accrue aux pannes, et c'est l'avantage le plus important.

Nous avons commencé par une définition des agents et systèmes multi-agents ainsi que leur environnement et la communication inter-agents. Ensuite, nous avons donné une explication de l'organisation et l'auto-organisation, pour à la fin faire une description du protocole auto-organisable et l'ensemble des processus caractérisant ce protocole.

Par la suite, et après avoir étudié les systèmes de production et leurs caractéristiques, nous avons considéré le pilotage de ces systèmes de production, précisément sur la surveillance : détection, diagnostic et traitement des pannes.

Dans la partie conception, nous avons proposé une approche multi-agents appropriée et donné les algorithmes expliqués et modélisés, chacun, par un diagramme d'activité, un diagramme des cas d'utilisation, un diagramme de séquence, un diagramme d'états/transitions et un diagramme de collaboration.

Enfin pour bien montrer l'apport d'appliquer un protocole AO lors des interactions des agents nous avons fait recours à une plateforme de simulation qu'est le Netlogo.la plate forme NetLogo est un outil facile de simulation. Cette plate

forme nous a permis de réaliser les différents cas de traitement de pannes dans un système de production, tout en manipulant un ensemble d'agents, un ensemble de tâches et un ensemble de liens de communication.

Ce que nous avons présenté à travers ce mémoire, n'est qu'une étape primaire qui nécessite d'être suivie d'une autre, qu'est la mise en œuvre réelle du système sur un système concret de production. Ainsi, beaucoup de points sont à approfondir. Pour cela, nous suggérons ces perspectives :

- Réaliser ce travail sur un cas de figure de système de production concret, tel qu'un atelier flexible, ou une cellule flexible.
- Mettre en œuvre les autres processus du protocole AO, tel que :
  - Quorum d'Agent.
  - Point de Contrôle et Consensus.

- Etendre l'étude sur l'ordonnancement dans le pilotage des systèmes de production.
- L'implémentation multi-postes sur un réseau physique en utilisant Java.

Ce mémoire préparé au sein du C.E.R.I.S.T. nous a permis d'enrichir nos connaissances dans le domaine des Systèmes Multi-Agents, de comprendre les Systèmes de Production, il nous a permis aussi de comprendre mieux comment chercher, extraire et manipuler l'information utile parmi une panoplie d'informations.

Bibliographie

Bibliographie

- **[Bar 95]** M. Barbuceanu M. and Fox M.S. Cool: a language for describing coordination in multi agent systems. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-1), pages 17-24, San Francisco, CA, 1995.
- **[Bus 04]** S Bussmann. N.R. Jennings, M. Wooldridge. Multiagent Systems for Manufacturing Control: A Design Methodology. Springer Series on Agent Technology, Springer. 2004.
- **[Cam 98]** V Camps. Vers une théorie de l'auto-organisation dans les systèmes multi agents basée sur la coopération : application à la recherche d'information dans un système d'information répartie. Thèse de Doctorat, Université de Paul Sabatier de Toulouse, Janvier 1998.
- **[Che 99]** Cheng-Gang Bian, Wen Cao, Gunnar Hartvigsen. ViSe2 _ An Agent-Based Expert Consulting System with Efficient Cooperation. Department of Computer Science Institute of Mathematical and Physical Sciences University of Tromsø, Norway 1999.
- **[Con 95]** R Conte, Castelfranchi C. Introduction of cognitive and social action. P.1-16-UCL Press-ISBN: 1-85728-186-1.1995.
- **[Dav 83]** R. Davis and R. Smith. Negotiation as a metaphor for distributed problem solving. Artificial Intelligence. 20(1): 63-109, January 1983.
- **[Dra 98]** G DRAGHICI, N BRINZEI, I FILIPAS. La modélisation et la simulation en vue de la conduite des systèmes de production. Les cahiers des enseignements francophones en Roumanie, 1998.
- **[Dur 79]** Durand D. la systémique, Que sais-je ? presse universitaire France-1979.
- **[Fer 95]** J FERBER. Les systèmes multi-agents, Vers une intelligence collective. Ed Masson, 1995.

- **[Gia 88]** GIARD V. Gestion de production. 2Eme édition. Paris : Ed Economica 1988.
- **[Hab 06]** G HABCHI, M HUGET, M PRALUS. D'une approche coposant vers une approche agent pour un pilotage optimisé des systèmes de production. 6e Conférence Francophone de MOdélisation et SIMulation – MOSIM'06 Rabat – Maroc du 3 au 5 avril 2006.
- **[Hal 06]** I HALILALI, Z MESLOUB. Vers un protocole d'auto-organisation au sein d'un système multi-agents. Mémoire de fin d'étude Université H. Boumediene. Algérie, 2006.
- **[Her 98]** A. HERBON. On line production control of a flexible multi-cell manufacturing system operating in a highly dynamic environment. International journal of production research, vol 36, n10, pp2771-2791. 1998.
- **[Hut 04]** G HUTZLER. Systèmes Distribués, Réactifs et Adaptatifs, Systèmes Multi- Agents réactifs. LaMI (Laboratoire de Méthodes Informatiques) SyDRA hutzler@lami.univ-evry.fr http://www.lami.univ-evry.fr/~hutzler/Cours/SMA.htm. 2004.
- **[Ish 92]** T Ishida, Les Gasser, and Makoto Yokoo. Organization Self-Design of Distributed Production Systems. IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 2, April 1992.
- **[Jai 97]** A. K. JAIN et H. A. ELMARAGHI. Production scheduling/rescheduling in flexible manufacturing. International journal of production research, vol35, n1, pp281-309.1997.
- **[Kou 97]** Kouiss K., Pierreval H., Mebarki N. Using muliagent architecture in FMS for dynamic scheduling. Journal of Intelligent Manufacturing, vol. 8, pp.41- 47. 1997.
- **[Lin 01]** J LIND. Iterative Software Engineering for Multiagent Systems. Volume 1994 of Lecture Notes in Artificial Intelligence, Springer Verlag, Heidelberg, 2001.
- **[Mal 99]** E malville. l'auto-organisation de groupe pour l'allocation de tâches dans les systèmes multi-agents : application à CORBA. thèse doctorat, univ de SAVOIE, le 25 mars 1999.
- **[Mat 06]** http://www.math-info.univ-paris5.fr/~pastre/IA. 2006.

- **[Min 79]** Mintzberg H, H. The Structuring of Organizations: a synthesis of the reaserch. Ed Hugo, 1979.
- **[Mor 77]** Morin E. La méthode (1) : La nature de la Nature. Ed Le seuil; 1977.
- **[Nag 00]** R Nagpal. A Catalog of Biologically-inspired Primitive for engineering self-Organization. Department of Systems Biology, Harvard Medical School240 Longwood Avenue, Boston MA 02115, USA.2000.
- **[Net 06]** NetLogo User Manual version 3.1.1June 16, 2006.
- **[Nou 94]** J NOUBISSI-TCHAKO. Contribution à la conception d'un système de pilotage distribué pour les systemes automatisés de production. Thèse de doctorat, université de Valenciennes, 1994.
- **[Oul 99]** D. OULHADJ. Système multi-agents pour le pilotage distribué de cellules flexibles de production autonomes. Thèse Magister, institut national de formation en informatique (INI). 1999.
- **[Pap 81]** Seymour PAPERT. Jaillissement de l'esprit - Ordinateurs et apprentissage. Ed Flammarion, 1981.
- **[Piq 96]** C Piquemal-Baluard, Pierre Glize. Des aptitudes non cognitivistes d'agents pour l'auto-organisation. Actes de la Journée PRC-GDR Intelligence Artificielle sur le thème des Systèmes Multi-Agents, Toulouse, 2 février 1996.
- **[Ras 91]** Rasmussen, J. Modelling Distributed Decision Making. In Distributed Decision Making: Cognitive Models for Cooperative Work. Ed J. Rasmussen, B. Brehmer and J. Leplat, John Wiley & Sons Ltd, 1991.
- **[Ros 75]** De rosney joel. le merascope – vers une vision globale. Ed du seuil. 1975-p-94-97.1975.
- **[Roy 98]** Roy D. Une architecture hiérarchisée multi-agents pour le pilotage réactif d'ateliers de production. Thèse de Doctorat : Université de Metz, 1998.
- **[Rus 95]** S. RUSSEL & P. NORVIG. Artificial Intelligence: a Modern Approach. Ed Prentice-Hall. 1995.
- **[Sab98]** I. SABUNCUOGLU. A study of scheduling rules of flexible manufacturing systems: a simulation approach. International journal of production research, vol 36, n 2, pp 527-546.1998.

- **[Sch 91]** Schmidt, K. Cooperative Work: A Conceptual Framework, In Distributed Decision Making: Cognitive Models for Cooperative Work. Ed Leplat, 1991.
- **[So 93]** Young-pa So and Edmund H. Durfee. An Organizational Self-Design Model for Organizational Change. In Working Notes of the AAAI-93 Workshop on AI and Theories of Groups and Organizations, July 1993.
- **[Sua 98]** O .A SUAREZ, J.L. A. FORONDA et M. Abren. Standard based framework for development of manufacturing control system. International journal of computer integrated manufacturing, vol 11.n 5. 1998.
- **[Uni 05]** http://www.univ-montp3.fr/miap/~jq/IntCollective6.pdf 2005.
- **[Üns 93]** Ünsal Cem. Self-organization in large populations of mobile robots. Master of Science in Electrical Engineering, Blacksburg, Virginia; May 1993.
- **[Vie 00]** http://www.vieartificielle.com/article/index.php?action=article&id=51 Université Paris 5 - Maîtrise de mathématiques - Maîtrise MASS - MST ISASH Dominique Pastre Module INTELLIGENCE ARTIFICIELLE. 1999/2000.
- **[Wei 99]** Weiss G. Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence. The MIT Press, Cambridge, Massachusetts.1999.
- **[Woo 00]** M WOOLRIDGE. On the Sources of Complexity in Agent Design, In Applied Artificial Intelligence. Ed Conte, 2000.
- **[Woo 95]** M. Wooldridge and N.R. Jennings. Intelligent agents: Theory and practice. The Knowledge Ingineering Review, 10(2):115-152, USA 1995.
- **[Yam 01]** YAMAM. Un modèle d'organisation pour les systèmes multi-agents. Implémentation dans la plate-forme Phoenix. 3e Conférence Francophone de Modélisation et SIMulation -Conception, Analyse et Gestion des Systèmes Industriels-, MOSIM'01, Troyes France. – du 25 au 27 Avril 2001.