N°d'ordre 12/2003-M/IN

République algérienne démocratique et populaire.
Ministère de l'enseignement supérieur.

Université des sciences et des technologies Houari Boumediènne
USTHB.

Faculté de génie électrique et d'informatique .

Thèse pour l'obtention
d'un grade de Magister en informatique.

Option :Intelligence artificielle et bases de données avancées.

Présenté par : Melle Fadila AOUSSAT.

Sujet :

Modélisation et exécution des procédés logiciels à base d'agents intelligents.

Soutenue le:13-07-2003, devant le jury composé de :

Mr N. Badache ;Maître de conférence ; USTHB.	**Président**
Mr M. Ahmed Nacer ; Maître de conférence ; USTHB.	**Directeur de thèse**
Mme Z. Alimazighi ; Maître de conférence ; USTHB.	**Examinateur**
Mr A. Belkheir ; Docteur d'état en informatique.	**Examinateur**

- 2003-

Dédicaces

Je dédie ce travail à ma petite famille, en particulier à ma maman, qui n’a jamais cessé de croire en moi et qui m’a toujours soutenu et encouragé dans ma vie,

Je t’aime maman.

A mon papa, qui nous m’a donner la liberté et les moyens de réaliser mes rêves.

Je t’aime papa.

A mes frères et sœurs : Salim, Sabrina, Farah et Mahdi, les rayons de soleil de ma vie.

Je vous aime.

A mon fiancé Fayçal et sa famille.

Tous ceux et celles qui m’ont aider dans ce travail, en particulier Nabila, et Nawel.

REMERCIEMENTS

Nous tenons à remercier toutes celles et tous ceux qui ont apporté conseils, aide et encouragements pour ce travail.

Nos remerciements vont en particulier à madame et messieurs les membres du jury, Mme Z.ALIMAZIGHI, Mr N.BADACHE et Mr A.BELKEIR, pour les soins qu'ils mettent à examiner notre travail ; nous sollicitons par avance leur aimable indulgence.

Que Mr M.AHMED NACER, notre directeur de thèse, trouve ici l'expression de toute notre reconnaissance pour sa patience et ses précieux conseils.

Nous n'oublions pas tous les enseignants qui nous ont permis à atteindre ce niveau.

Last but not Least, je remercie ma famille, en particulier mon papa, pour ses encouragements et ses coups de pouce en rédaction.

Sommaire :

Chapitre 6: Définition des connaissances de l'agent modèle de procédé logiciels.

Table des figures :

Résumé :

La modélisation des procédés logiciels doit permettre la représentation de toutes les caractéristiques et spécificités du développement logiciels. Le modèle de procédé logiciel idéal doit ainsi être suffisamment flexible pour permettre la «prise en charge des modifications» pendant l'exécution, sans pour autant perdre en vue les objectifs initiaux ; il doit permettre une part d'« autonomie» locale tout en gardant la cohérence globale du développement; il doit en outre supporter l'hétérogénéité des outils et langages de communications tout en fournissant le «support d'interactions» qui couvre tous les participants. En substances, notre modèle de procédé doit être «intelligent ».

Dans cette thèse nous présentons une nouvelle approche pour la modélisation et l'exécution des procédés logiciels à base d'agents intelligents. L'idée développée ici est de donner le même niveau d'importance à la modélisation et à l'exécution des modèles de procédé en exploitants toutes les caractéristiques de l'agent intelligent. Le système multi-agents définis est un système hiérarchique où les agents de chaque niveau s'occupent et représentent un certain aspect du modèle de procédé logiciels tel que la modélisation et l'exécution.

La première partie est un état de l'art relatif à l'utilisation des agents intelligents dans l'ingénierie logiciels ; elle sera suivie d'une synthèse des différents systèmes étudiés.

La deuxième partie est consacrée à la présentation des différents concepts liés à notre approche ; nous y définissons les agents de notre système et les interactions entre eux, ainsi que les connaissances de ces agents modèles de procédé logiciel.

Introduction générale

La modélisation des procédés logiciels est un domaine de recherche très actif. L'avancement rapide des technologies utilisées dans ce domaine et les pressions industrielles exigent continuellement de nouvelles méthodes de conception et de réalisation de logiciels afin d'avoir des résultats rapides, de qualité, et au moindre coût possible. Une des technologies les plus présentes dans le génie logiciel est l'Internet qui est considéré comme un support pour le travail coopératif et distribué ; ce dernier commence à remplacer progressivement le travail centralisé traditionnel.

Les changements dans les méthodes de travail des développeurs impliquent également des changements dans la modélisation des procédés logiciels. L'exploitation optimum des possibilités ainsi offertes par le travail distribué sera obtenue par une distribution tout aussi optimale des procédés logiciels afin de respecter les conditions de spécificité et d'autonomie des équipes de travail. Parallèlement, et afin de préserver la consistance et la cohérence du développement, un contrôle global et un suivi des activités sont indispensables dans tous les espaces de travail.

La distribution du travail exige également des modèles d'interactions performants qui s'adaptent aux types d'interactions inter et intra espaces de travail - inter et intra fragments de procédés logiciels. Ces modèles d'interactions doivent alors être fournis par le modèle de procédé logiciel.

Le modèle de procédé logiciel doit être dynamique ; il doit supporter et prendre en charge tout changement, y compris pendant son exécution, et particulièrement les événements qui n'étaient pas prévus ; il doit gérer également toute exception qui se présente, en prenant des décisions rapides et efficaces, sans influence sur son exécution ou retard dans les délais préalablement déterminées.

Si nous résumons ces caractéristiques par un seul mot, nous dirons que notre modèle de procédé doit être «intelligent » ; cette intelligence peut être caractérisée par le concept d' « agent » à la fois capable d'autonomie d'action et d'interaction tout en restant à l'intérieur des spécificités qu'on veut donner à notre modèle de procédé.

Les travaux effectués dans le génie logiciel nous montrent que le concept d'agent a été utilisé particulièrement pour le développement d'environnements de programmations destinés

à l'exécution des modèles de procédé logiciel. Les travaux se sont focalisés sur l'exploitation des capacités d'interactions (communications, négociations, coopérations…) et la possibilité de distribuer les tâches, par spécialisation ou par redondances, des agents. Par contre, peu de travaux ont été effectués concernant la modélisation des procédés logiciels basés agents **[13]**. Les capacités représentationelles des agents sont ainsi faiblement mises à profit.

Nous nous proposons, dans ce qui suit, de concevoir un modèle de procédé sous forme d'un système multi-agents . Ce modèle sera doté d'outils, de modèles d'interactions et des connaissances nécessaires pour fournir un procédé logiciel dynamique et distribué, et qui respecte les caractéristiques sus – citées.

- La première section est un état de l'art des différents travaux effectués dans le domaine du développement de logiciels et des agents intelligents. Elle sera suivie d'une synthèse des environnements centrés procédés utilisant le concept d'agent pour l'exécution des procédés logiciels.

  En substance, la majorité des travaux effectués dans le domaine du développement de logiciels s'est soldé par la conception et la réalisation d'environnements centrés procédés logiciels à base d'agents intelligents ; cette situation est due aux priorités données par les chercheurs à l'exploitation de la capacité d'action et d'interaction de l'agent.

- La deuxième section porte sur notre contribution, à savoir la présentation d'une nouvelle approche pour la modélisation et l'exécution des procédés logiciels à base d'agents intelligents. Nous y présentons les différents agents logiciels du système et les différents modèles d'interactions, ainsi que la plupart des connaissances stockées dans les bases de connaissances de nos agents modèle de procédé logiciels.

  Notre modèle de procédé est formalisé en UML ; toutefois, nous précisons bien que notre contribution se focalise sur l'action de la modélisation et de l'exécution et non pas sur le formalisme utilisé.

Introduction :

Pour une meilleure compréhension des approches, technologies présentées ; nous considérons qu'il est nécessaire de définir toutes les notions et concepts de bases qui vont être utilisées dans les chapitres suivants. De ce fait, ce chapitre introduit particulièrement des définitions sur : les agents, les environnements centrés procédés logiciels, les modèles de procédés ainsi que les systèmes gestionnaires de workflow.

I. Agent et agent logiciel :

1- Définition de l'agent :

Un agent peut être défini comme une entité physique ou abstraite autonome capable de percevoir et d'agir sur elle-même et sur son environnement. Il dispose d'une représentation partielle ou complète de cet environnement ; il a la possibilité de communiquer avec d'autres agents ; son comportement est la conséquence de ses objectifs, ses observations, ses connaissances, compétences et des interactions avec les autres agents.

Un agent est défini par un ensemble de caractéristiques tel que son rôle, ses objectifs, ses croyances (informations, connaissances), sa capacité décisionnelle, ses capacités de communication et d'apprentissages **[9]**.

Un agent logiciel est simplement un autre genre d'abstraction de logiciel, une abstraction de la même manière que les méthodes, les fonctions, et les objets sont des abstractions de logiciel. Un objet est une abstraction qui décrit des méthodes et des attributs d'un composant de logiciel. Un agent, toutefois, est une abstraction à niveau élevé de logiciel qui fournit une manière commode et puissante de décrire une entité complexe du logiciel. Plutôt que de définir en termes de méthodes et attributs, un agent est défini en fonction de son comportement.

L'intelligence est définie comme un degré de raisonnement et une disposition à l'apprentissage.**[23]**

L'agent est dit intelligent car il a un certain degré d'autonomie, des capacités décisionnelles et des moyens pour s'adapter à son environnement et mettre à jour ses connaissances ; en d'autres termes il a des capacités d'apprentissage.

2- Propriétés d'un agent :

L'agent possède les propriétés suivantes :

- **L'autonomie :** l'agent peut agir sans l'intervention directe de l'être humain (ou autres agents) ; ainsi il peut planifier des étapes d'exécution de tâches ou assigner des ressources de manière automatique sans une intervention externe.
- **Des capacités sociales :** cela veut dire que l'agent a les moyens pour interagir, communiquer et échanger des informations avec des personnes ou d'autres agents.
- **Des capacités d'apprentissage et d'adaptation :** l'agent peut s'adapter avec les changements dynamiques de son environnement ; par exemple : il peut mettre à jour sa base de données ou bien ajouter d'autres règles pour pouvoir agir selon l'état actuel du système.
- **Des capacités décisionnelles :** décide des actions à entreprendre selon le but à réaliser et selon l'état de l'environnement. **[13]**

Pour mener à bien son rôle un agent possède :
- Des connaissances sur les tâches qu'il est censé accomplir.
- Des structures de données pour représenter ses moyens, les manipuler et les gérer.
- Des moyens de communication.
- Des moyens d'agir sur l'environnement.
- Des moyens de planifier ses actions.
- Des moyens de se modéliser et de modéliser les autres **[18].**

3- Agents cognitifs et réactifs :

Ces deux concepts ont donné lieu à deux écoles de pensée :

L'école cognitive qui défends l'idée que le SMA est composé d'un nombre d'agents «intelligents». Chaque agent possède une base de connaissances comprenant l'ensemble d'informations nécessaire pour réaliser ses tâches ; ces agents sont dit intentionnels car ils possèdent des buts, plans d'actions ainsi que des mécanismes de planification qui leurs permettent de « raisonner » et exécuter la meilleure action.

L'école réactive prétend qu'il n'est pas nécessaire que l'agent ait une intelligence individuelle pour que le système soit intelligent globalement. Des mécanismes de réaction directe aux événements (sans interprétation des messages ou planification des tâches) suffisent largement pour résoudre des problèmes qualifiés de complexes. L'exemple le plus cité est celui d'une fourmilière **[9]**.

4- Les notions «faible » et «forte » du concept «agent » :

Selon la plus part des chercheurs un agent peut être défini de deux manières : soit avec la notion *«faible »* de l'agent, en d'autres termes il a les caractéristiques suivantes : l'autonomie, sociabilité, la réactivité et la continuité temporelle ; cette notion est la plus répandue et la plus utilisée pour la définition de l'agent.

Toutefois, pour certains chercheurs, particulièrement ceux du domaine de l'intelligence artificielle ; le terme agent a un sens plus spécifique et plus profond, ce qui donne naissance à la notion «forte de l'agent », et qui est caractérisée par :

- *La mobilité* : l'agent a la possibilité de se déplacer dans son environnement.
- *La rationalité :* cela suppose que l'agent agit dans le but d'achever ses objectifs.
- *L'adaptabilité :* l'agent peut ajuster son comportement en fonction de son utilisateur ou de son environnement.
- *La collaboration :* l'agent n'accepte pas automatiquement des instructions, mais prend en considération que l'être humain peut se tromper ou oublier des informations importantes.
- *Pas de contradiction :* cela veut dire que les agents n'ont pas de buts contradictoires**[23].**

5- Comparaison entre l'objet et l'agent :

Les programmeurs familiarisés avec l'approche orientée objet échouent souvent de saisir ce qui est nouveau dans l'approche orientée agent **[24].**

Il 'y a beaucoup de points en commun entre l'objet et l'agent, le concept de base des deux concepts est très ressemblant :

- L'objet est défini comme une entité conceptuelle qui encapsule des états ; il a la possibilité d'effectuer des actions (invocation de méthodes) et communique avec les autres objets par passations de messages, ces caractéristiques peuvent être facilement attribuées à l'agent sans risque d'erreur.

D'autre part, il ne faut pas oublier qu'il y a des différences assez importantes qui fait que l'agent apporte des mécanismes d'abstractions plus puissants que ceux de l'objet ; ces derniers

peuvent être très efficaces dans certains domaine ou l'objet apporte juste l'essentiel (ex : la construction des systèmes logiciels complexes **[10]**).

- La première différence est le degré d'autonomie de l'objet et l'agent : la programmation orientée objet peut permettre de déclarer les méthodes comme «privée » donc utilisable que par l'objet lui-même ou «publique » en d'autres termes utilisables par d'autres objets. Dans le premier cas cela peut nous faire penser à une sorte d'autonomie car l'objet a le contrôle total sur son état interne mais dans le deuxième cas l'objet n'est pas maître de ses méthodes publiques ni de son état interne car les méthodes publiques par définition peuvent être invoquées et exécutées par n'importe quel autre objet sans accord préalable ou contrôle de l'objet encapsulant la méthode.

Nous pourrons penser à déclarer toutes les méthodes comme méthodes privées mais ce genre de programmation n'est pas très efficace car il n'y aurait plus d'interactions et ceci est contraire aux caractéristiques des systèmes ouverts.

Dans le cas d'agent, aucune «méthode » ou action ne peut être effectuée sans l'accord de l'agent concerné, cela ressemble plus à une demande de service qu'à une invocation de méthode, dans ce cas l'agent a le contrôle total sur son état interne, il peut décider selon ses conditions d'exécuter ou de ne pas exécuter une action.

Le sens d'autonomie dans l'objet et l'agent n'est pas le même ; car l'exécution d'une action dans le cas de l'agent dépend toujours de l'agent qui encapsule l'action, ce qui n'est pas le cas pour l'objet.

- Le deuxième point est que contrairement à l'objet, l'agent est capable de comportement flexible(action, réaction et comportement social, autonomie), il peut s'adapter aux changements de son environnement et exécuter les actions qui lui conviennent selon son état interne et ses objectifs, et en prenant ses propres décisions.

- Le troisième point est que l'agent de sa nature est continuellement actif en d'autre terme il est en observation continuelle le de son environnement mettant à jour continuellement son état interne, sélectionnant et exécutant les actions qui conviennent. Par contre l'objet est de nature passive, il devient actif que par invocation de méthode.

6- Les avantages d'un système multi-agents :

Un SMA est constitué de résolveurs (agents) qui travaillent ensemble pour résoudre un problème donné ; les avantages les plus importants sont :

- **La décentralisation** : Il est capable de découper un système complexe en un ensemble de sous systèmes décentralisés coopérants, de plus, plusieurs groupes de l'organisation peuvent être distribués géographiquement.

- **Réutilisation des composants ou sous systèmes précédants** : les SMAs nous donnent cet avantage : le recyclage des composants, ce qui nous permet un gain de temps et la création de nouveaux systèmes en inter-opérant ceux déjà existants même si le degré d'hétérogénéité est assez élevé.

- **Support de travail coopératif :** Il est capable de modéliser et supporter une large gamme d'interaction dans le travail coopératif, pour cela les agents logiciels peuvent interagir en étant autonomes de toute assistance humaine.

- **Flexibilité :** Il est capable de prendre en charge les caractéristiques d'un environnement distribué, de systèmes hétérogènes, dans une évolution permanente**[25].**

7- Les langages de représentation des SMA : [9]

Les langages et formalismes utilisés pour la modélisation des SMA diffèrent selon que l'on veut implémenter, représenter les connaissances ou l'interaction de l'agent. Pour cela ces langages peuvent être classés en 5 types :

- **Type 1 : Langages d'implémentation**

Ces langages sont utilisés pour programmer le SMA ; ce qui inclus les structures informatiques, les systèmes de parallélisme, l'implémentation effective des comportements.
Les langages de programmation utilisés à ce niveau sont les langages classiques tel que : C++, Prolog, SmalltalK…

- **Type 2 : langage de communication**

Assure l'interaction entre les agents par la transmission d'information et l'échange mutuel des demandes de services et de renseignement.
Ce type de langage permet aux agents (éventuellement hétérogènes) de coordonner leurs actions et de coopérer afin de réaliser le but en commun.

- **Type 3 : langage de description de comportement et des lois de l'environnement.**

Il décrit les comportements des agents de manière abstraite, en d'autre terme, ce qui nous intéresse c'est le comportement et pas son implémentation.

- **Type 4 : langage de représentation des connaissances :**

Ces langages sont utilisés particulièrement pour les agents cognitifs pour représenter les modèles internes de leur monde et, dans le but de leurs permettre de raisonner, de faire des prévisions et de planifier leurs tâches en fonction de leurs objectifs, ressources et connaissances.
Ces systèmes sont souvent associés à des systèmes logiques qui dispose d'une syntaxe et sémantique rigoureuse pour expliciter leurs inférences.

- **Type 5 : langage de formalisation et de spécification :**

C'est le niveau le plus abstrait, l'objectif de ces langages est de formaliser les objectifs du SMA par la notion d'interaction et les conditions de modélisation et d'implémentations.

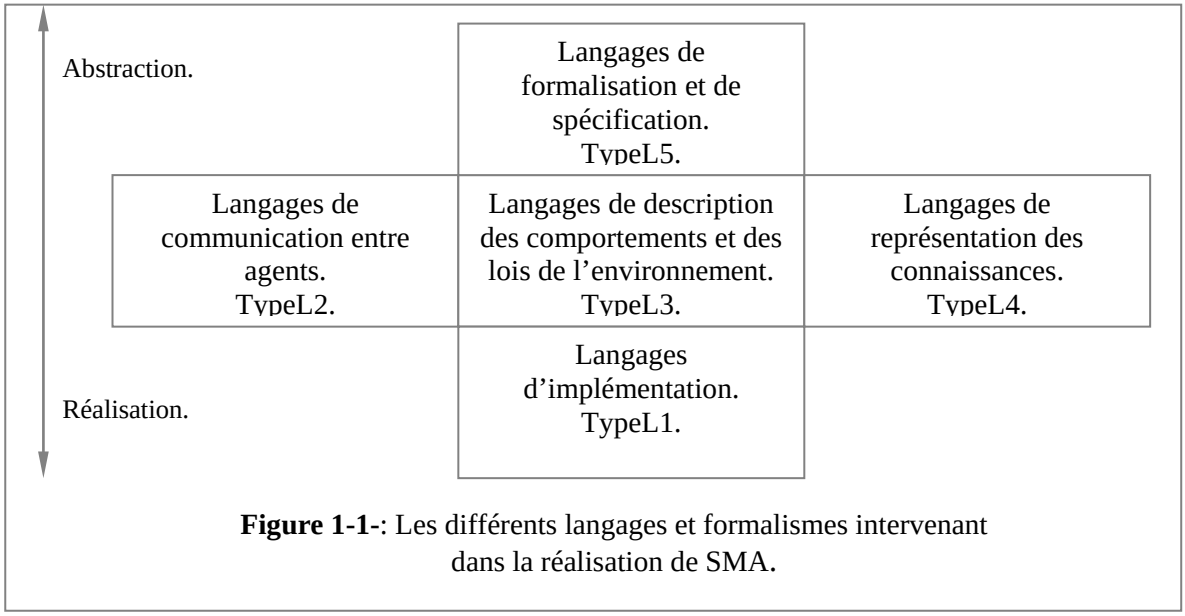


Figure 1-1-: Les différents langages et formalismes intervenant dans la réalisation de SMA.

8- Ou peut-on utiliser l'agent et les systèmes multi-agents :

Les agents avec leurs caractéristiques et spécificités tels que l'autonomie, sociabilité et la réactivité conviennent très bien au développement des systèmes qui ont une ou plusieurs des caractéristiques suivantes :

- **Les systèmes décomposables :** Des systèmes qui peuvent être décomposable en plusieurs sous systèmes. Ainsi, pour chaque composant, un ou plusieurs agents sont affectés et prennent charge son fonctionnement et son interaction avec les autres composants du système**[10].**

- **Les systèmes à interaction intense :** Les systèmes ou l'interaction et l'échange d'information a un grand impact sur la qualité du système ; l'efficacité des échanges d'informations reflète l'efficacité du système. Un exemple de ces systèmes : les environnements de développement de logiciel ou 32% du temps de développement et basé sur l'interaction et la négociation entre les développeurs **[2].**

- **Les systèmes hétérogènes :** C'est des systèmes qui contiennent des composants, outils, hétérogènes ou bien qui utilisent ces outils de manières différentes ; les systèmes qui utilisent plusieurs langages de programmations différents font aussi partie des systèmes hétérogènes. L'agent peut respecter cette hétérogénéité en utilisant un certain niveau d'abstraction pour un meilleur fonctionnement et une meilleur gestion de ces différences **[3].**

- **Les systèmes distribués :** L'agent s'adapte très bien à ce type de système et gère cette distribution de manière assez efficace car par sa nature autonomie_ sociabilité _réactivité ainsi que ses capacités d'interaction lui permettent de s'adapter a cette distribution.

- **Des systèmes évolutifs ou incrémental :** L'agent permet plus facilement l'évolution des systèmes, car il a des capacités décisionnelles pour agir au bon moment et de s'adapter au nouvel environnement, de plus l'ajout de nouveaux types d'agent en cas de nécessité est plus simple a faire que de concevoir un nouveau système avec de nouvelles données **[1].**

II. Les environnements d'ingénierie logiciels centres procédés :

1- Définition du modèle de procédés logiciels [23] [14] :

La plupart des articles étudiés définissent le procédé logiciel comme suit :

Un procédé logiciel est un ensemble d'activités, de règles, procédures, techniques, outils et ressources faisons intervenir un ensemble de personnes (développeurs) afin d'assurer le développement et la maintenance d'un produit logiciel et cela bien entendu dans les meilleurs délais avec le meilleur coût pour un produit de meilleure qualité possible.

Un modèle de procédé est la définition d'un procédé spécifique à un projet particulier et transcrit dans un langage de modélisation de procédé PML fournis par l'environnement centré procédé **[14].**

« Un modèle de procédé logiciel représente le chemin formel pour organiser et décrire de cycle de vie du logiciel, la méthodologie de développement, les outils et ressources et les développeurs qui travail dessus »**[14]**.

2- Définition de l'Environnement Centré Procédé logiciels (ECP) :

Un environnement centré procédés logiciels est un système ou les procédés logiciels sont modélisés et exécutés.

Un environnement centré procédés est un système dans lequel le logiciel est fabriqué selon son modèle de procédé. **[14]**

L'ECP peut fournir des modèles de procédés de base qui peuvent être taillés selon les besoins du projet à développer.

Il existe différents ECPs avec différentes architectures, cependant il est indispensable que chacun d'eux offrent un minimum de services.

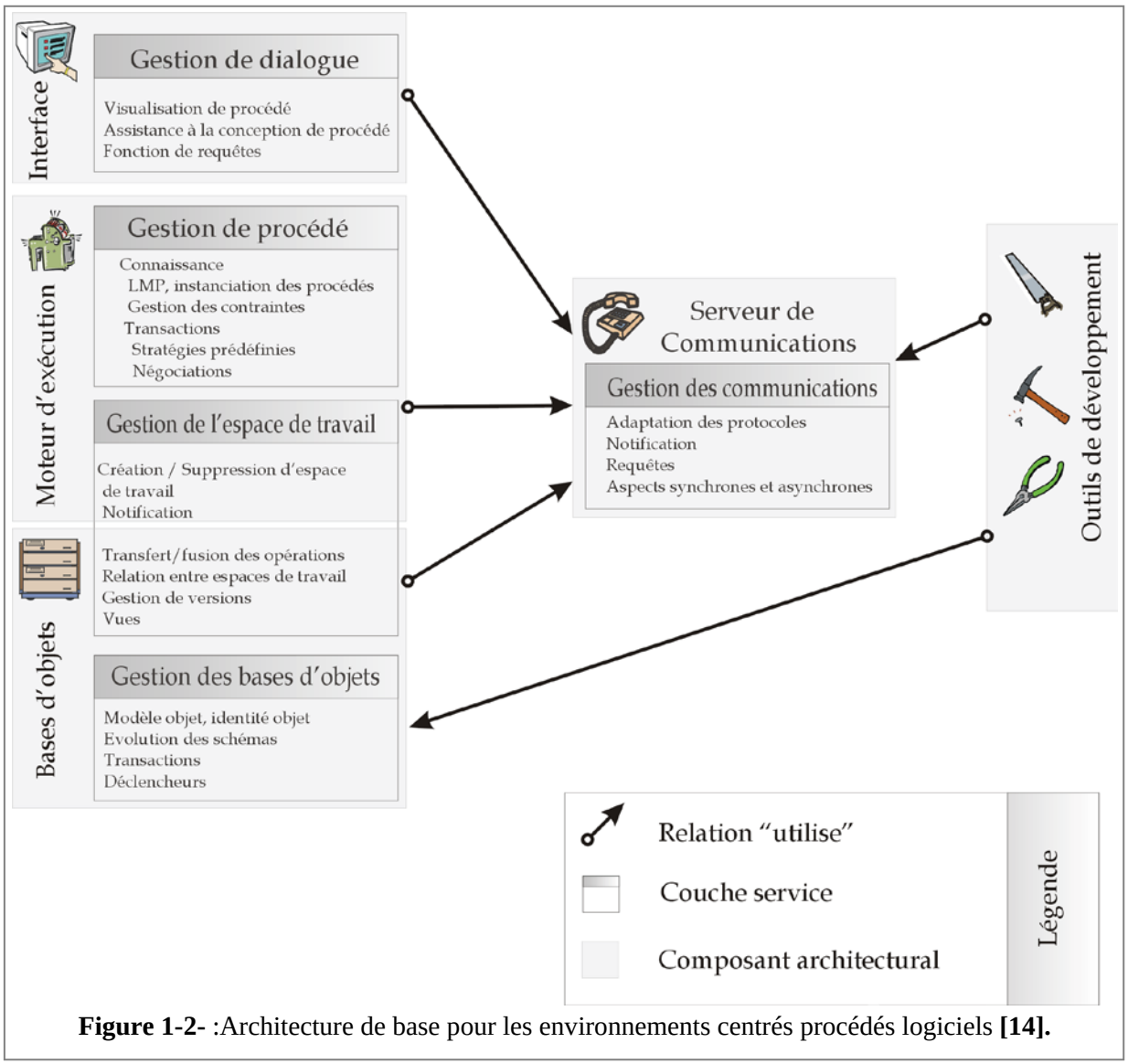


Figure 1-2- :Architecture de base pour les environnements centrés procédés logiciels **[14].**

- **Pilote d'exécution de procédé : (Process enactment driver) :**

Le PEP permet l'exécution ou l'interprétation d'un MPL (modèle de procédé logiciel) selon ses activités hiérarchiques ; il gèrent l'ordre des sous tâches qui doivent être effectuées et les conditions à satisfaire avant leurs applications ; il joue le rôle d'un pilote automatique qui

initialise le MPL, exécute ses sous tâches, accepte les données d'entrées d'un développeur, met a jour et diffuse l'état de ses sous tâches et leurs déclenchements.

L'état d'un MPL ou d'une tâche est l'indicateur de l'état actuel de développement ; l'état est mis à jour par le pilote d'exécution de procédé et il est basé sur l'interaction avec les développeurs. La mise à jour de l'état avec l'état actuel de développement représente la progression dans l'exécution du projet **[16].**

- **Interface développeurs :**

L'interface développeur est un environnement qui montre le MPL et ses activités ; cette interface permet aux différents développeurs d'exécuter les MPLs de manière concurrente et effectuer seulement les sous tâches dont ils ont besoin à travers un processus d'orientation (ou de guidage) et des espaces de travail. Le processus de guidage est un moyen d'informer les développeurs quels sont les sous tâches a effectuer et quand ces sous tâches sont prête à commencer.

L'interface développeur est constituée de plusieurs «fenêtre de tâche » qui représentent l'état de progression des tâches et leurs sous tâches sous forme explicite «graphes, formes géométriques…»**[16]**

- **Base d'objet :**

Cette base contient toutes les informations qui concernent le développement du logiciel, les versions développées, les documents le concernant, les modèles de procédés utilisés…etc.

Le gestionnaire des bases d'objets prend en charge tous ce qui concerne la base d'objets : le stockage des informations, les droits d'accès, l'accès concurrent …etc.

- **Espaces de travail :**

C'est un lieu (peut être réel ou virtuel) ou travail un groupe de développeurs, tous les moyens, outils de travail et de communications sont fournis pour avoir un bon environnement de développement.

- **Serveur de communication :**

C'est un élément important dans l'environnement centré procédé car c'est lui qui gère toute transaction, échange d'informations entre les espaces de travail, les développeurs que ce soit communication, négociation ou autre.

3- Langage de modélisation de procédés (PML) :

Un MPL spécifie un modèle de procédé sous forme d'une hiérarchie d'activités ; en d'autre terme il décrit la décomposition des activités de développement en petites activités hiérarchiques ; il décrit aussi les ressources requises, les outils, les développeurs et autres ressources critiques ; ces petites activités hiérarchiques sont appelées «tâches » ou «actions ». L'ordre d'exécution de ces tâches peut être séquentiel, parallèle, alternatif ou conditionnel.

Le niveau de décomposition peut être arbitraire selon le degré de complexité du procédé ; les tâches peuvent elles mêmes être décomposées en sous tâches, une action peut être une invocation d'outil ou une simple transformation de ressources.

Un modèle de procédé est la définition d'un procédé spécifique à un projet et transcrit dans un langage de modélisation de procédé PML fournis par l'environnement centré procédé. **[14]**

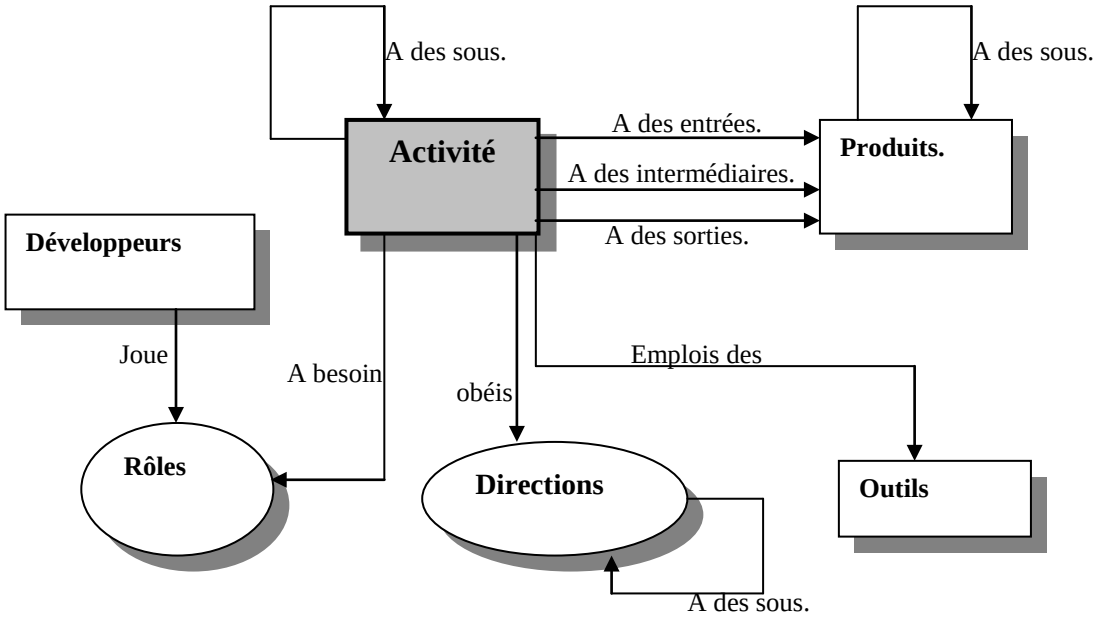


Figure 1-2- : Le concept de base pour la modélisation du procédé logiciel. **[23]**

- Activité : Est une opération atomique ou une étape de procédé.
- Produits : L'ensemble des objets, programmes à développer, délivrer, maintenir dans un projet.
- Ressources : Ce sont les moyens exigés par l'activité pour être exécutée, il y a deux types de ressources :
- Le développeur : (agent humain) généralement directement lié à l'activité par son rôle.
Le rôle : Peut être utilisé pour définir les responsabilités et obligations du développeur.
La direction : C'est le pilote « guide » qui peut être des règles ou des procédures qui gouverne l'activité.
- Les Outils : C'est l'équipement matériel ou logiciel dont l'activité a besoin pour son exécution.

Différents PML sont utilisés, nous pouvons distinguer 4 paradigmes importants :

- **PML basé langage de programmation :** Il décrit le procédé sous forme de programme. Pour cela, il utilise des langages de programmation classiques étendus par des concepts relatifs aux procédés.

- **PML basé règles :** Dans cette approche, le procédé est décrit par des règles Pré-condition/Action/Post-condition. La pré-condition est une contrainte nécessaire à l'exécution de l'action tandis que la post-condition représente l'effet de la condition.

- **PML basé réseau :** Le procédé est décrit par un réseau de Pétri où les transitions représentent les activités et les places les pré-conditions et post-conditions. La vérification d'une condition est représentée par la présence d'un jeton dans cette place**.[14] [23]**

- **PML basé multi-paradigmes :** Un seul formalisme peut difficilement décrire tous les éléments du procédé logiciel ainsi que toutes les étapes du procédé. En effet, un formalisme

qui décrit « tout » ne peut être que très complexe et surchargé par les concepts utilisés. L'approche multi-paradigme combine deux ou plusieurs paradigmes pour décrire les différents éléments et activités du procédé logiciel **[14].**

Quelques environnements d'ingénierie logiciel centrés procédés qui utilisent diffèrents paradigmes pour la modélisation de leurs procédés logiciels **[23].**

Basé programmation.	**Basé règle.**	**Basé réseau.**	**Basé Multi-paradigmes.**
Arcadia.	Marvel.	ProcessWeaver.	Adele.
IPSE2.5.	Merlin.	SPADE.	ALF.
Syner Vision.	OIKOS.		EPOS.

4- Les aspects d'un environnement d'ingénierie logiciels efficace :

Avec l'arrivée de l'Internet et l'évolution des traditions de travail des développeurs de logiciels ; les environnements d'ingénierie logiciels doivent répondre à des exigences grandissantes des développeurs et évoluer pour mieux exploiter les outils et nouvelles technologies qui s'offrent à eux.

Les environnements d'ingénierie logiciels centrés procédés logiciels actuels sont caractérisés par :

- L'ECP peut être distribué géographiquement tel que chaque groupe de travail peut être dans un lieu distant (une autre ville ou un autre pays) ; il peut développer un composant a n'importe quel moment de la journée sans se soucier des autres groupes de travail.

- Les modèles de procédés peuvent être hétérogènes un environnement d'ingénierie logiciel doit être capable de supporter et d'exécuter tous types de procédés logiciels.

- L'interaction, la communication, la coordination, prennent une place très importante et doivent être pris en charge de manière efficace car la plupart du temps de développement est consommé dans l'échange d'informations entre les développeurs.

- L'ECP doit gérer l'évolution dans des modèles de procédés, et doit supporter leur changement dynamique pendant l'exécution.

- ECP doit pouvoir utiliser tous types d'outils ou de langages, les intégrer facilement sans remettre en cause tout l'environnement et son fonctionnement.

- Dans le cas ou l'ECP est distribué, les espaces de travail distribués doivent avoir une liberté locale et pouvoir agir sur les tâches locales sans intervention hiérarchique ; d'autre part un certain contrôle global pour sauvegarder la consistance et la cohérence des modèles de procédés en exécution est exigé.

- L'ECP doit avoir le moyen de prendre en charge toutes déviations des modèles de procédés exécutés par rapport aux modèles de procédés instanciés de cela de manière dynamique.**[2]**

- La plus part des informations et connaissances qui circulent dans l'ECP doivent être partagées ; elles sont utilisées par plusieurs développeurs, il est possible de les modifier, les

supprimer ou les déplacer selon le niveau d'accès de l'utilisateur ; dans certains cas, elles sont distribuées et stockées dans plusieurs bases de données. Un ECP efficace doit pouvoir gérer tout accès ou modifications des informations et préserver la cohérence et la consistance des données manipulées.

Définition du workflow :

Selon les articles lus, le gestionnaire workflow est défini comme suit :

« Le gestionnaire workflow «*WFM* » est une technologie qui permet l'automatisation et l'intégration des procédés de tâches. Dans les systèmes gestionnaires de workflow ; le modèle workflow (workflow type) est prédéfinis pour chaque procédé business.

Le modèle workflow consiste en un nombre d'activité et un ensemble de transitions conditionnelles que relie ses activités ensemble. Aucune restriction n'est faite sur la localisation des activités qui peuvent être potentiellement différées a travers le réseau dans un système workflow entier organisationnel »**[15].**

Aussi, suivant la coalition du workflow gestionnaire, le workflow est définis comme suit :

« Le workflow est l'automatisation du procédé business (travail ou affaire), en ensemble ou en partie, durant lequel les documents, informations ou tâches sont passé d'un participant à un autre pour effectuer une action en respectant un ensemble de règles procédurales ».

Si on compare entre la technologie des procédés logiciel et le workflow nous remarquons que le premier se focalise sur les procédés d'ingénierie logiciels, cette technologie est plus générale et inclue l'évolution des procédés comme une partie d'elle, aussi contrairement aux procédés business, les procédés logiciels ne sont pas stables et changent fréquemment, ils peuvent même changer pendant l'exécution, aussi la technologie des procédés logiciels doit être capable de matérialiser le travail créatif centré humains ; par contre le procédé business est généralement stable et prédéfinis car il est constitué de tâches monotones et qui ne changent pas souvent tel que le routage des documents**[23].**

III. Conclusion :

D'après les propriétés de l'agent définis dans la plupart des articles et qui sont : l'autonomie sociabilité réactivité **[23], [24]**, **[12]** et d'après les exigences demandées par les environnements d'ingénierie logiciels centrés procédés ; il est évident que l'agent peut répondre et régler la plus part des conditions exigées par ces environnements. Ainsi ; l'agent par sa nature peut s'adapter très naturellement aux exigences des ECP tel qu'il est montré dans le tableau suivant :

Caractéristiques d'un ECP performant.	**Propriété de l'agent qui peut gérer cet aspect de l'ECP.**
- Interactions intenses : Communications, négociations, coordinations à différents niveaux.	- Sociable et possède des capacités de communication et d'échange d'informations.
- Autonomie décisionnelle locale.	- Autonomie et des capacités décisionnelles.
- Evolution continue des procédés logiciels, de l'environnement et des outils utilisés. - Dynamique.	- Active de manière continue, - Capacités d'adaptation au nouvel environnement. - Capacités d'apprentissage.
- Peut être distribué géographiquement.	- un SMA est constitué de plusieurs agents qui peuvent être distribués géographiquement - Les agents peuvent être mobiles donc gérer un certain aspect de la distribution géographique.
- Hétérogène (composants, outils, langages …)	- plusieurs type d'agents peuvent être définis, chacun avec ses propres capacités et spécificités pour gérer cette hétérogénéité.

Les autres aspects tel que la cohérence des données ou la consistance des informations manipulées peuvent être prise en charge par l'agent (des types d'agents définis spécialement pour cela) ; cela, en association avec d'autres approches ou techniques **[2].**

Développer un environnement «puissant » d'ingénierie logiciels centrés procédés logiciels basés sur le concept d'agent est une solution qu'il faut explorer avec beaucoup d'intérêt car l'agent avec ces caractéristiques manifeste beaucoup d'avantages exploitables pour le génie logiciels et pas seulement pour les environnements centrés procédés.

Introduction :

Dans ce chapitre, nous présentons les plus importants travaux concernant l'utilisation des agents intelligents dans le domaine du génie logiciels, nous Définissons les plus importants environnements centrés procédés logiciels existants qui utilisent les agents logiciels pour gérer une partie ou tous les services que doit fournir un environnement de développement de logiciels efficace ; ces environnements sont : l'environnement CAGIS, la structure ALLIANCE, l'environnement PEACE+ et la structure ADEPT.

I. L'environnement centré procédé logiciels CAGIS :

En 1997 débute, à l'Université des Sciences et de la Technologie de Norvège, un projet appelé «**C**oopérative ***A**gents **I**n **G**lobale **I**nformation **E**space* » (CAGIS) destiné à déterminer comment le travail distribué hétérogène peut être supporté dans un environnement centré procédés logiciels.

L'équipe du projet CAGIS est constituée des groupes « base de données », « système d'information » et « ingénierie logiciels » de la dite Université .

Le projet a abouti à la réalisation d'un ECP prototype qui fournit un support pour les procédés d'ingénierie logiciels coopératifs.

Les objectifs initiaux du projet étaient de :

- Fournir un meilleur support pour le travail distribué et concurrent, pour des équipes de développeurs qui travaillent en coopération (concepteur, ingénieurs…).
- Fournir un espace de travail doté d'agents logiciels distribués et d'espaces de stockages d'informations utilisant des outils et formalismes spécifique pour chaque domaine.
- Fournir des agents logiciels spécialisés, tels que des agents de recherche Internet, pour opérer dans un espace d'information global.

Le CAGIS PCE à trois composants de base :

- **Des agents logiciels**, qui sont utilisés pour supporter la coopération dynamique des procédés logiciels; ces agents logiciels prennent généralement en charge les activités inter espaces de travail tel que la négociation ou la coordination entre les outils et les éléments workflow.

- **Un système workflow,** supportant les procédés logiciels distribués mobiles. Ce système workflow est utilisé pour modéliser des procédés logiciels simples et répétitifs**[21]**.

  Le système workflow permet l'instanciation des modèles workflow et la distribution sous forme de fragments de procédés à travers les différents espaces de travail.

  Un des avantages de cette méthode est de permettre l'adaptation du workflow aux conditions environnementales locales. Il permet, aussi, de faire déplacer les instances workflow durant l'exécution, ce qui est un atout la réallocation des activités en exécution, la gestion des exceptions…etc.

- **Le glue serveur agent- workflow ,** fournit le moyen pour que le système multi-agents interagisse avec le système workflow.

  Le modèle glue définit la relation entre les éléments workflow et les agents logiciels ; le glue serveur va alors fournir un support où les agents logiciels peuvent déclencher des

activités workflow ou, inversement, le système workflow assigne des activités aux agents logiciels **[23].**

Dans les paragraphes qui suivent nous allons essayer de définir de manière plus détaillée le fonctionnement de chaque composant, et plus spécialement les agents et le système multi-agents pour le travail coopératif.

1 - Architecture multi-agents pour la coopération dans l'ingénierie logiciel :

1- Les composants d'un système multi-agents pour la coopération dans l'ingénierie logiciels :

Le SMA développé pour l'environnement CAGIS se compose de 4 éléments essentiels :

- **Agents:**

un agent est une entité logiciel autonome crée et actionnée par un utilisateur (ou un autre agent), il a des caractéristiques d'autonomie, d'interaction, la réaction à l'environnement... etc.

Chaque agent est activé pour effectuer une tâche particulière, l'objectif de cette distribution est d'arriver à un fonctionnement cohérent et une utilisation optimale des ressources.

Selon les services fournis par les agents nous pouvons distinguer :

Agents systèmes : Ces agents sont crées par défaut pour l'administration de l'architecture multi-agents comme la création et la suppression de «places de rencontres ». Dans certains systèmes plusieurs agents systèmes sont indispensables ; par exemple : les agents moniteurs tracent les événements dans «les espaces de travail » et «les places de rencontres » pour avoir des informations et mesures pertinentes en se basant sur des métriques prédéfinies.

Agents locaux : Pour assister les développeurs dans «les espaces de travail », ces agents agissent comme des secrétaires personnels, ils aident dans les tâches locales, définissent les plans et aide dans l'exécution des modèles de procédés.

Agents d'interactions : ils aident les participants dans leur travail coopératif ; ces agents peuvent être vus comme des intermédiaires entre les différents éléments «espaces de travail », ils sont classés en 4 sous classes :

- *Agents communicants :* Ils sont utilisés pour prendre en charge les différences entre les méthodes de communication entre les participants.
- *Agents négociateurs :* Formulent leurs demandes dans le but d'avoir un arrangement avec leur interlocuteur et pouvoir prendre une décision pour avancer.
- *Agents de coordination :* Représentent des gestionnaires émettant des ordres de travail qui concernent un groupe de développeurs ; ils peuvent être des gestionnaires de haut niveau pouvant être l'intermédiaire entre des agents de négociations pour trouver un compromis.
- *Agents médiateurs :* sont utilisés pour aider les agents de négociations à trouver un compromis (arrangement) dans des négociations qui n'aboutissent pas, pour cela, l'agent médiateur peut consulter les expériences précédantes comme il peut agir selon

les règles de l'organisation ou demander au gestionnaire de projet (humain) à l'aider à prendre une décision.**[25]**

- **Espace de travail (work space WS) :**

  Un espace de travail est un endroit ou les développeurs et les agents logiciels peuvent accéder à des données partagées (généralement des fichiers) et des outils qui peuvent être privés ou partagés par un groupe de personnes. Aussi, l'interaction entre les personnes et les agents logiciels s'effectue dans cet espace. La forme la plus simple d'un espace de travail peut être un fichier système muni de services pour lire et écrire des fichiers, un espace de travail plus complexe peut fournir le versionnement des fichiers, l'accès à quelques bases de données (espaces de stockage), différents services, support pour le web…etc.

- **Place de rencontre d'agents (PRA) Agent meeting place :**

  C'est un lieu ou les agents se rencontrent pour interagir, aussi elle peut être considérée comme un marché ou les agents s'échangent des informations et des services. PRA doit être muni de moyens intelligents pour faciliter l'interaction. Le but le plus important de la PRA est de faciliter la coopération entre les agents et les applications.

- **Les espaces de stockage (reposiroty) :**

  Un espace de stockage peut être vu comme une mémoire persistante qui peut être locale, globale ou distribuée, comme il peut être un serveur d'informations. Dans la forme la plus simple il offre des services de sauvegarde et retrait de données, et dans une forme plus complexe il peut fournir des services pour modéliser les données, rechercher, comparer, traiter…etc. Les espaces de stockages peuvent être, également, accéder par des outils et des agents.

  L'espace de stockage «dépôt » le plus important est le «dépôt de production » car c'est ici que les versions du produit sont sauvegardées, les autres dépôts peuvent contenir les modèles de procédés, les résultats des expériences précédantes, rapports sur les erreurs remarquées par des utilisateurs…etc.**[19]**

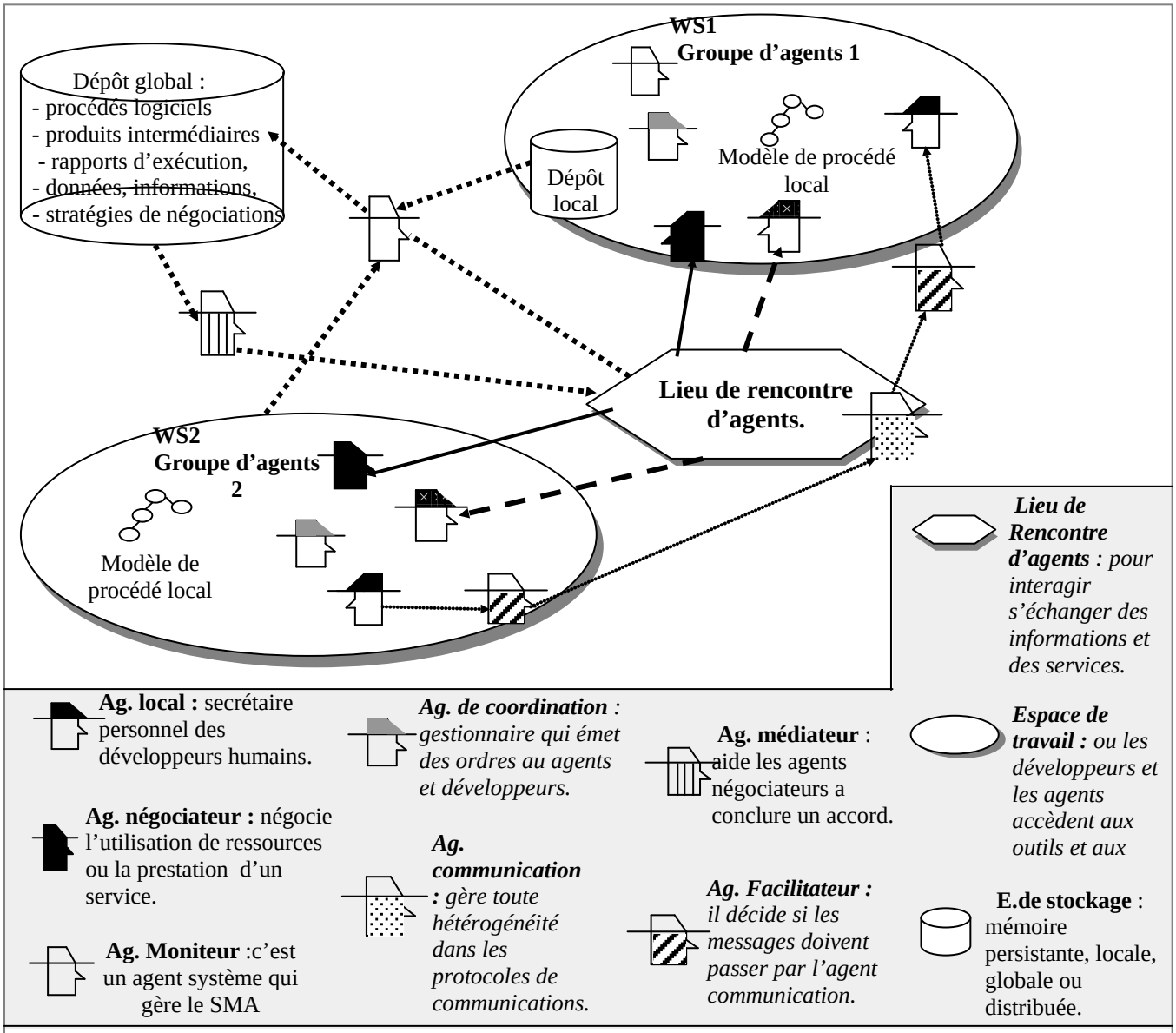


Figure 2-1- : Les 4 composants de l'architecture de la coopération d'ingénierie logiciel et leur interconnexion et inter coopération **[12].**

2- Système d'agents intelligents distribués « DIAS » [23] :

« Le ***S****ystème D'****A****gents* ***I****ntelligents* ***D****istribués »* est l'implémentation de l'architecture multi-agents pour la coopération dans l'ingénierie logiciels. Les composants de base du DIAS sont les agents logiciels et les places de rencontre d'agents.

Plusieurs problèmes ont été rencontrés lors de l'implémentation du DIAS ; les problèmes les plus importants sont le résultat de l'utilisation de plusieurs langages de programmation à la fois. Pour chaque composant implémenté, un langage de programmation «le plus adéquat » a été choisi, et ces langages n'étaient pas toujours compatibles ; de ce fait plusieurs versions ont été implémentées, la dernière est le DIAS III.

Un des problèmes majeur était de savoir comment supporter la mobilité des agents, car les langages choisis ne permettaient pas cela.

- **Les composants de DIAS :**

L'architecture de DIAS est assez simple elle consiste en seulement deux importants composants :

- L'agent :

il est programmé pour réaliser un ensemble de tâches, il garde les caractéristiques définis auparavant tel que l'autonomie et l'interaction, trois types d'agents sont définis :

Les agents systèmes : Ils administrent l'architecture DIAS, plusieurs types d'agents systèmes existent, nous définissons :

- *Les agents gestionnaires(manager):* Sont responsable de la gestion des espaces de rencontres d'agents (AMPs).
- *Les agents facilitateurs (facilitator):* Ils facilitent la communication entre les agents.
- *Les agents moniteusr :* Ils enregistrent les événements et gèrent la sécurité des places de rencontres.
- *Les agents dépôts :* Ils gèrent l'accès aux espaces de stockages MAJ, consultations…
- *Les agents interfaces* : Ils jouent le rôle d'intermédiaire entre les agents systèmes et les autres applications.

Les agents participants : Ces agents peuvent être stationnaires ou mobiles, ils fournissent un support pour les processus de coopération dans les places de rencontres d'agents. Les agents participants typiques sont :

Les agents communicants, les agents médiateurs, ces agents sont déjà définis dans les parties précédantes.

Les agents KQML : Ils permettent la communication entre les agents et les utilisateurs à travers des messages KQML.

Les agents utilisateurs : Ils sont créés par l'utilisateur ou les développeurs des applications d'agents, ils peuvent être stationnaires ou mobiles.

- Les places d'agents :

Comme définis précédemment, c'est le lieu ou les agents logiciels peuvent interagir, il existe deux types de places d'agents :

- *Places de rencontres d'agents :* c'est le lieu ou les agents s'échangent leurs services, communiquent avec d'autres agents.
- *Agent docking Place (place de rencontre agents- utilisateurs) :* c'est le lieu ou l'utilisateur interagit avec son agent et le lieu ou les agents peuvent être créés ou détruits.

Les agents peuvent aussi communiquer localement dans les ADPs.

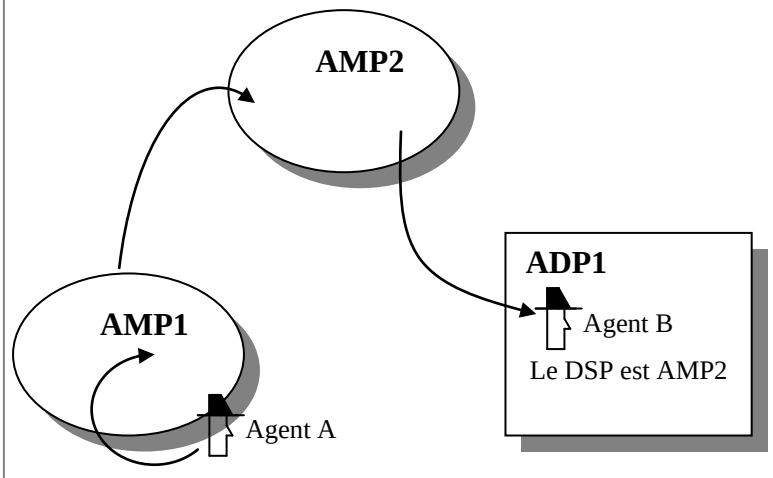
3- Les principaux concepts pour les systèmes d'agents mobiles :

- **La localisation d'agents :**

Ce qu'il y a de plus important dans un système multi-agents mobile est la localisation des agents mobiles. Un agent utilisateur peut commencer son déplacement par un ADP et visiter plusieurs AMPs, les agents participants aussi sont mobiles et peuvent se déplacer entre les AMPs et les ADPs . Le problème qui se pose est comment communiquer avec ses agents mobiles sachant que la communication est primordiale dans un SMA. Le première chose à faire pour assurer l'interaction avec des agents mobile est de repérer leur position. Pour cela trois solutions ont été proposées :

1- Chaque agent sait ou l'autre agent est situé : cette solution n'est pas très efficace car elle exige des mises à jour permanentes des informations reçues sur la position des autres agents et cela posera un problème de performance et d'extension.
2- La deuxième proposition est que l'agent laisse une trace la ou il passe : cette solution est aussi inefficace que la première, car quand un agent veut localiser un autre, il faudra qu'il suive tous ses déplacements à travers tous les AMPs parcourues ; cela n'est pas très pratique car il y aurait une perte de temps et de ressources ; de plus si un agent est détruit, toutes les traces doivent être suivies et effacées.
3- L'agent se rapporte à une AMP la ou il va : La troisième solution est la solution adoptée pour localiser les agents mobiles. Dans cette approche l'agent doit informer sa nouvelle position à une «AMP » avant tout déplacement.
   Le fournisseur de services DIAS (provider services DIAS - DSP-) est le composant qui a la position de l'agent, chaque agent est géré par un seul DSP.
   Deux règles sont utilisées pour décider du DSP de l'agent :
   - L'agent crée dans une AMP aura cette AMP comme DSP.
   - l'agent crée dans une ADP utilisera le DSP de cette ADP. (quand une ADP est crée une AMP doit être spécifiée comme DSP)**. [27]**
   Chaque agent a son DSP comme une partie de son identificateur de telle sorte a ce qu'il soit le DSP reconnue rapidement.

- **La communication agent :**



1- Le message de l'agent A vers l'agent B est premièrement envoyé à l'agent système local de AMP1.

2- L'agent système de AMP1, trouve l'adresse du DSP de l'agent B, et envoie le message à ce DSP, (ici c'est AMP2)

3- L'agent système de AMP2 cherche la position de l'agent B et envoie directement le message à l'agent B dans ADP1.

4- L'agent B reçoit le message et envoie un autre message à l'agent A directement.

Figure 2-2- : Exemple de communication.

La figure 2-2- est un exemple simple de l'échange d'information entre deux agents ; les AMPs s'échangent dynamiquement des identificateurs d'agents du système, d'un autre côté les AMPs peuvent être des DSPs, ils sauvegardent des informations sur les agents et leurs positions pour un usage ultérieur.
Deux types d'agents sont utilisés pour la communication dans le DIAS :

Les agents facilitateurs : chaque AMP et ADP a un agent facilitateur, tous les messages du système DIAS doivent être envoyés à travers des agents facilitateurs, qui déciderons si le

message doit être reçue directement par l'agent local ou passer par un agent de communication pour trouver la bonne destination.

Les agents de communication : Ils sont responsables de la communication entre les AMPs ; ils reçoivent des messages par des agents facilitateurs et ils les remettent à l'agent facilitateur le l'AMP destination.

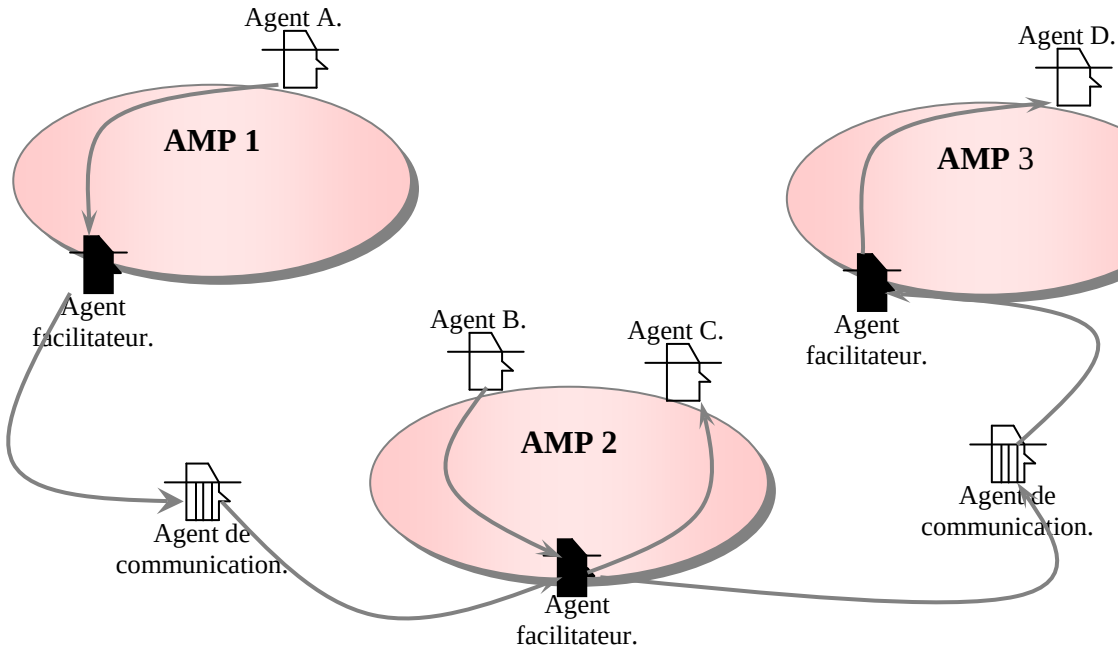


Figure 2-3- : Exemple : comment les agents facilitateurs travaillent.**[27]**

- **Mouvement de l'agent :**

DIAS supporte et utilise des agents mobiles; par conséquence, le système doit être capable de localiser les agents à n'importe quel moment. Pour cela l'agent en déplacement vers une autre AMP doit fournir des informations à trois places d'agents :

- Pour le DSP sur la place de rencontre ou il va se déplacer.
- Pour la place d'agent de la quel il va se déplacer (quitter).
- Pour la place d'agent qu'il va rejoindre.

- **La destruction d'agent :**

Dans DIAS les agents peuvent être créés et détruits pendant le temps d'exécution, cependant les agents systèmes ne sont pas détruits individuellement mais avec leurs places d'agents : si une place d'agents devient inutile donc destinée à la destruction, tous les agents associés sont détruits avec elle.

Les agents participants ont aussi une nature dynamique, cependant ils peuvent être éliminés immédiatement.

La durée de vie de l'agent utilisateur dépends de l'utilisateur ; il a la possibilité de l'éliminer directement sauf qu'il faut effacer leurs informations des places de rencontres avant leur destruction.

2- le système workflow :

1- Le procédé logiciels mobile CAGIS : [20]

Si en se base sur l'idée de développer un environnement de travail «distribué et coopératif » ; il est difficile voir impossible d'utiliser un modèle de procédé logiciel «centralisé » pour modéliser toutes les activités, tout en respectant les caractéristiques et différences environnementales des espaces de travail.

Un procédé logiciel est composé d'un ensemble d'activités, pour palier à ce problème : « la centralisation du modèle de procédés », l'idée est de décomposer le procédé logiciel en un ensemble de sous procédés reliés entre eux, ou chaque sous procédé appelé «fragment » peut être exécuté dans un espace de travail particulier. A l'intérieure de l'espace de travail il y a une certaine autonomie locale tel qu'il est possible de modifier, et adapter les fragments de procédés aux spécificités locales des espaces de travail, de ce faite, il serrait possible de respecter les différences et libertés de développement des espaces de travail.

Un fragment de procédé est composé d'un ensemble d'activités reliées entre eux par des pré_liens « prélink » : ces liens donnent les activités qui doivent être exécutées avant l'activité ; et des post-liens « post-link » : donnent l'ensemble d'activités qui doivent être exécutées après l'activité.
Le fragment de procédé peut être n'importe quel combinaison d'activités tel que montré dans la figure2-4- .

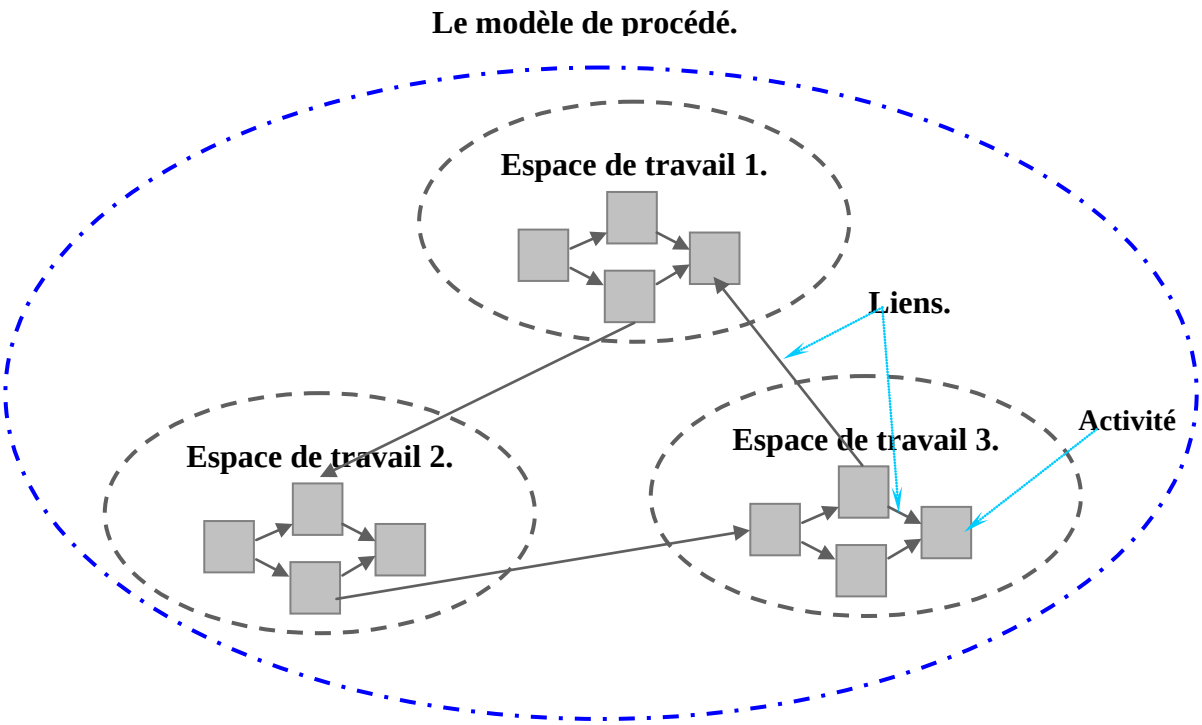


Figure 2-4- : Composants d'un modèle de procédé**[20].**

2- Déplacement d'un fragment de procédé d'un espaces de travail à un autre :

Le passage d'un espace de travail à un autre ne change rien à l'état fragment de procédé ni à n'importe quel chose qui peut influer sur l'exécution du fragment, la seul chose qui change est le nom de l'espace de travail ou il est localisé ;
pour passer d'un espace de travail à un autre il faut suivre les étapes suivantes :

Etape 1 : Le serveur de procédé 1 envoi un message au serveur de procédé 2 avec des paramètres:- le serveur de procédé destination : serveur de procédé 2.

- type de la requêtes : mouvement d'un fragment de procédé .
- l'espace de travail destination : WS4.
- le URL du fichier XML qui va contenir le fragment de procédé à envoyer :
  http:// www.serveur de procédé1/ WS1/ fragment de procédé1.xml.
  et le URL d'autres fichiers spécifiques.

Etape 2 : Le serveur de procédé 2 envoi une requête de téléchargement de fichier XML et des autres fichiers en utilisant le protocole http.

Etape 3 : Le serveur de procédé 2 reçoit le fichier xml et les autres fichiers, initialise le fragment dans le serveur de procédé 2 et le WS4 s'il n'y a pas de conflit.

Etape 4: Le serveur de procédé 1 détruit toutes les informations concernant le fragment de procédé1.

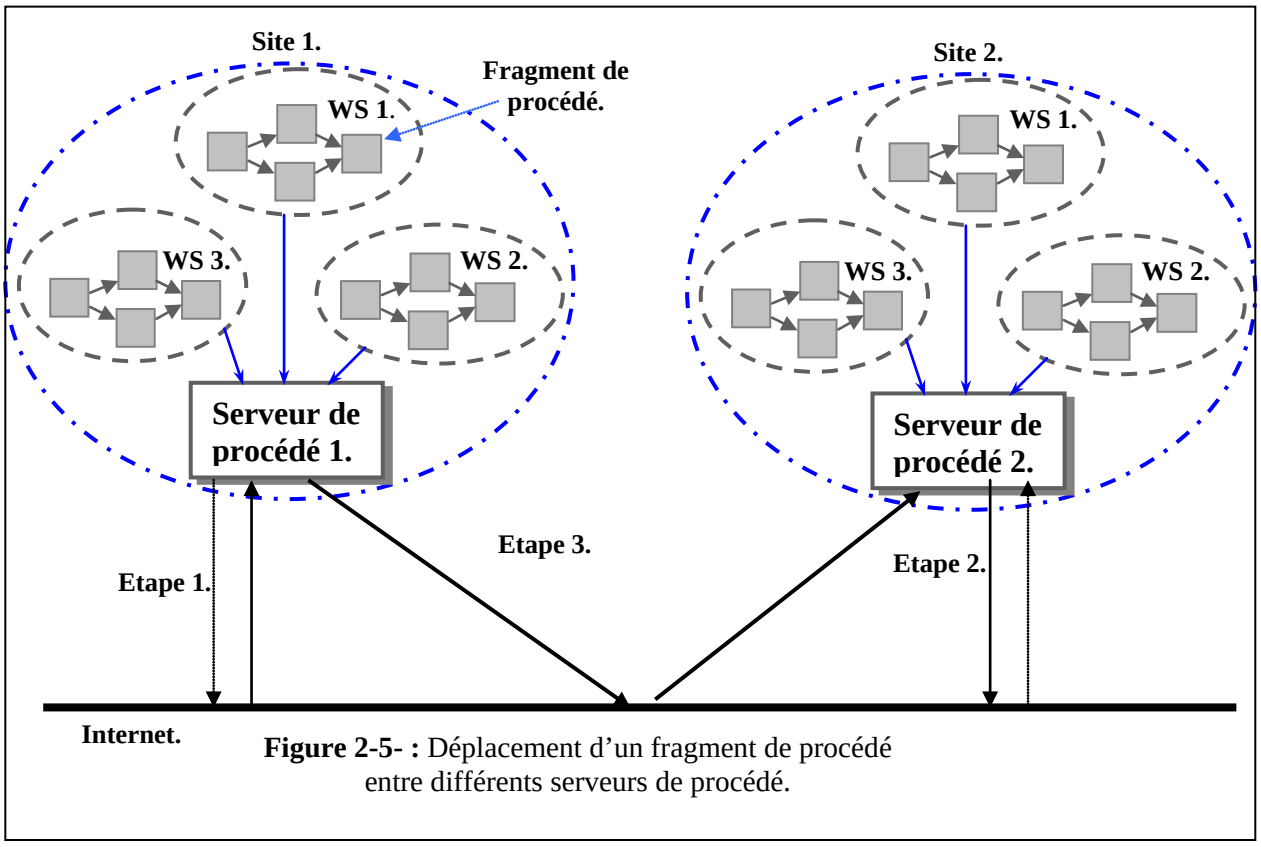


Figure 2-5- : Déplacement d'un fragment de procédé entre différents serveurs de procédé.

3- Les outils utilisés pour supporter les procédés logiciels et leurs mobilités :

Pour gérer la mobilité des fragments et le suivie de l'exécution du procédé logiciel même s'il est distribué le système DIAS est muni de 4 composants de base :

- **Le serveur de procédé :** Son rôle est de gérer l'état des informations concernant les activités tel que la synchronisation et les liens prélink et postlink ; aussi il s'occupe de l'enregistrement des nouvelles activités et la suppression de celles déjà existantes. Finalement comme nous l'avons déjà vu il gère le déplacement des fragments de procédé d'un serveur de procédé à un autre.
- **Le modélisateur de procédé (Process modeller) :** c'est une interface web qui permet au client de faire rentrer le modèle de procédé de manière incrémentale (activité par activité) et nous permet aussi de voir les modifications effectuées.
- **Agenda gestionnaire :** Le client web est muni d'un agenda gestionnaire qui permet à l'utilisateur ou à un groupe d'utilisateurs de choisir une activité à exécuter, de naviguer à travers le modèle de procédé ou de visualiser les activités.
- **Moniteur :** c'est l'administrateur de ces outils, ce qui nous permet la gestion de la progression du modèle de procédé.

La mobilité des fragments de procédés fournis beaucoup d'avantages, aussi, d'un autre côté, elle crée d'autres types de problèmes : si un procédé de logiciel change continuellement d'espace de travail il serait difficile à l'utilisateur de le localiser ou de suivre son chemin ; la solution adoptée pour résoudre ce problème est l'utilisation d'agents logiciels.
Les agents moniteur sont utilisés pour gérer les événements qui arrivent dans les espaces de travail et les places de rencontres d'agents, et les agents d'espaces de stockages sont utilisés pour stocker toutes les informations concernant les événements, de cette manière il est possible à n'importe quel utilisateur de savoir ou est localiser le fragment de procédé, son état en demandant des informations aux agents concernés.

Après définitions de ces deux composants (le système workflow et le système multi-agents) le problème qui se pose est comment combiner entre les fragments de procédés et les agents logiciels. Comment faire pour que chaque fragment ait ses propres agents logiciels aux moment ou il en a besoin, comme faire pour négocier l'allocation d'une certaine ressource entre deux fragments de procédés ; comment des agents logiciels peuvent communiquer avec des fragments pour effectuer leurs tâches tel que coordonner entre ces fragments.

3 - le glue serveur : [26]

La fonctionnalité la plus importante du glue serveur est la communication entre les système workflow et le système multi-agents, l'instanciation et l'exécution du «glue modèle » ;
il est constitué de 4 éléments :

- **Fragment serveur :** Il reçoit les requêtes du système workflow. Les requêtes glue serveur contiennent l'indentificateur du fragment de procédé demandant au glue serveur et des autres fragments de procédés concernés par l'interaction ; le nom et la classe de l'agent qui doit être initialisé.

- **Le moteur glue :** Exécute le modèle glue et définis les pairs (fragment , agent) basé sur la classe de l'agent et l'identificateur du fragment .

- **Agent client :** Quand les paires (fragment, agent) sont définies dans le modèle glue, l'agent client se connecte au système multi-agents et initialise l'agent comme spécifié dans la requête glue serveur et le modèle glue.

- **Le fragment client :** Selon les résultats retournés de l'initialisation des agents de ce qu'a été demandé dans le modèle glue, le fragment client envoi une réponse au workflow ou les actions vont être effectuées.

- **Le modèle glue :** L'utilisation du modèle glue spécifie comment les fragments et les agents interagissent, les fragments de procédé et les agents sont regroupés en pairs (fragment, agent) cette dernière modélise toutes les inter-opérations du fragment et de l'agent dans l'environnement centré logiciel.

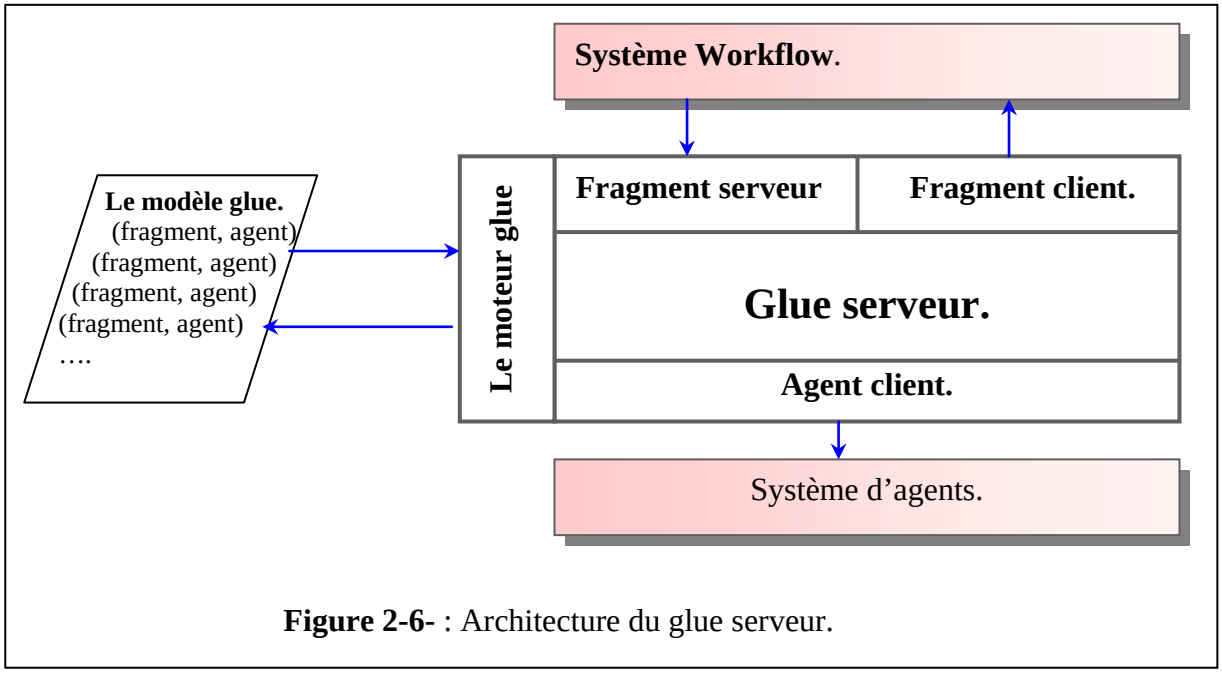


Figure 2-6- : Architecture du glue serveur.

Les interactions typiques entre les trois composants sont :

1- Le système workflow rapporte son état via l'interface workflow
2- Le glue serveur trouve une paire (fragment de procédé, agent) dans le modèle glue en utilisant le moteur glue.
3- L'interface agent initialise un agent comme spécifié dans le modèle glue.
4- Le système agent rapporte le résultat de l'interaction de l'agent initialisé par le glue serveur.
5- Le glue serveur active la réaction qui convient selon le glue modèle.

La figure suivante illustre les différents composants de l'environnement CAGIS en interaction.

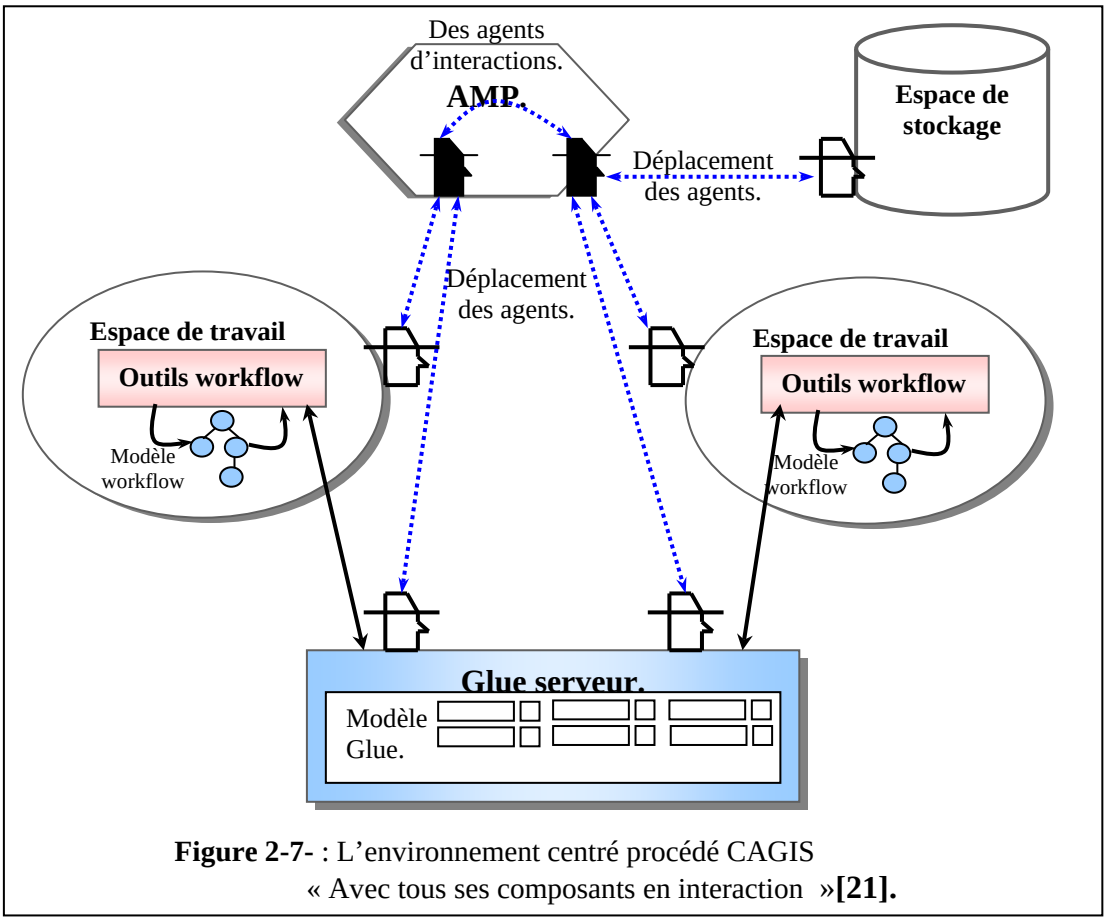


Figure 2-7- : L'environnement centré procédé CAGIS « Avec tous ses composants en interaction »**[21].**

4 – Le problème de consistance du modèle workflow utilisés dans CAGIS : [21]

Les modèles workflow utilisés dans CAGIS sont des modèles distribués ; les fragments de procédés sont adaptés, modifiés localement selon les moyens, compétences locales, de plus, les fragments sont mobiles et peuvent changer d'espace de travail.

La version courante de ECP CAGIS n'émet aucune restriction pour que l'utilisateur effectue des changements dans le modèle de procédé local, chaque utilisateur peut changer la définition de son fragment de procédé dans son espace de travail, la définition du workflow étant en XML, il peut effectuer des modifications en utilisant un simple éditeur de texte.

Toute cette liberté de modification et cette mobilité peut créer des problèmes de consistances du modèle workflow et influer sur la qualité d'exécution du modèle de procédé, les changements typiques sont : ajouts ou suppressions d'activités, fusions ou modifications d'activités.

Ces incohérences peuvent apparaître pour plusieurs raisons les plus importantes sont :

- Lorsqu'on introduit des changements pendant l'exécution du modèle de procédé logiciel.

Ces changements sont dus au faite que le modèle workflow ne peut être défini définitivement avec tous ses détails du premier coup avant le début de son exécution.

Ces changements peuvent être effectués pour plusieurs raisons, les plus essentielles sont liées au client tel que : les ajournements du client ; révision des besoins et ajouts d'autres tâches à effectuer ; un malentendu avec le client…etc.

- Lorsqu'un fragment de procédé est dupliqué et les copies envoyées à divers espaces de travail pour être exécutées ; chaque copie peut être modifiée et adaptée à l'espace de travail récepteur, ce qui peut créer des incohérences.

La solution retenue pour gérer l'inconsistance est d'avoir un espace de travail gestionnaire qui à toutes les copies, versions des fragments du modèle de procédé, ainsi, à l'aide de certains agents spécialisés il est possible de détecter les incohérences et de les traiter.

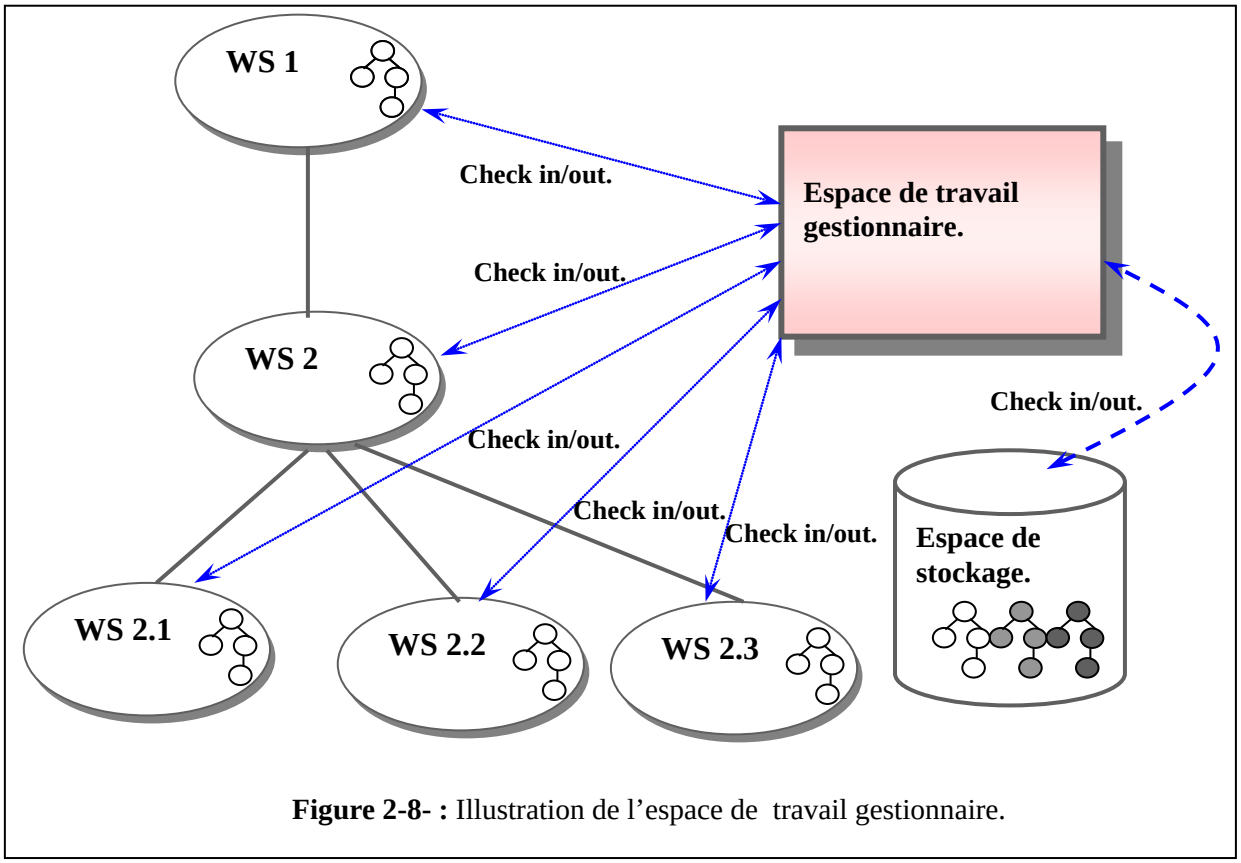


Figure 2-8- : Illustration de l'espace de travail gestionnaire.

1- Les agents prise en charge de la consistance :

Les agents responsables de la consistance du modèle workflow ont été identifiés :

- Les agents négociateurs.
- Les agents coordinations : pour synchroniser les changements dans les cas du scénario anarchie
- Les agents notifications.
- Les agents de mise à jours.
- Les agents de vote : ces agents sont utilisés quand l'organisation concernée par le changement est basée sur des principes démocratiques, exemple s'il y a un changement de règle d'accès, l'espace de travail gestionnaire initialise des agents de vote pour collecter les décisions des espaces de travail concernés par le changement, la réalisation de changements dépendras des résultats du vote

II. L'environnement ALLIANCE [2] :

1- La structure ALLIANCE :

La structure ALLIANCE *« **Al**gebra and **L**ogic for **I**nteropérable **A**ge**N**ts in **C**oopérative **E**nvironment »* est une structure logiciel qui permet la modélisation, l'exécution et le contrôle des procédés logiciels intensif.

Les procédés logiciels intensif peuvent être des procédés workflow, des procédé logiciels, des systèmes d'information automatisé…etc.**[2]**

Cette structure permet plus particulièrement la décentralisation, la coopération et le changement dynamique des procédés logiciels.

Afin de modéliser les procédés logiciels, la structure ALLIANCE utilise un langage de description de procédé basé logique flou et se base sur le concept de tâche.

Pour exécuter et contrôler les procédés logiciels ALLIANCE se basent sur l'approche basée buts ou elle utilise des agents logiciels pour réaliser ces objectifs.

Le type des agents est défini selon les tâches à accomplir, ils utilisent leurs connaissances et capacités pour interagir et réaliser les objectifs de la tâche qu'il implémente.

Dans ce qui suit nous présentons de manière résumé les agents de la structure ALLIANCE et leurs fonctionnement.

2-Cycle de vie du procédé dans ALLIANCE :

Dans ALLIANCE, le procédé suit un cycle de vie spécifié en terme de phase interactives pour : La définition, instanciation, activation, le contrôle et le changement des procédés.

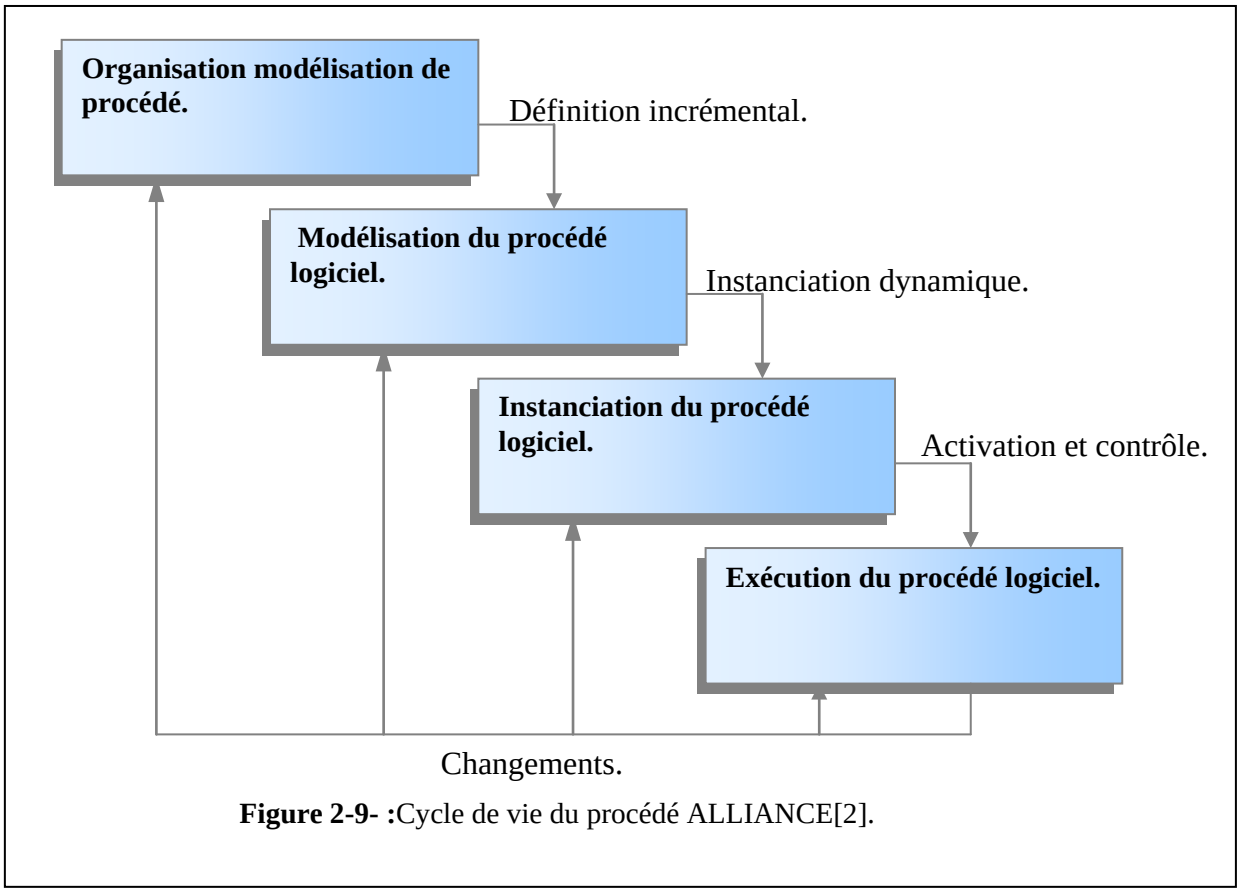


Figure 2-9- : Cycle de vie du procédé ALLIANCE[2].

- Dans la phase organisation modélisation de procédé logiciel, les procédés business sont modélisés en terme des objectifs ou buts à atteindre.

- Dans la phase modélisation de procédé logiciel, des sous ensembles du procédé business destinés à être automatisés à travers un système support de procédé sont définis.

- Dans la phase d'instanciation de procédé logiciel, des activités de travail sont allouées aux participants ou sont relier à des applications logiciel.

- Dans la phase exécution, les procédés logiciels instanciés sont activés par le système d'exécution du procédé logiciel ; ce dernier est capable d'interpréter la définition des procédés qui sont instanciés, aussi, il interagit avec les participants et invoque des outils logiciels et autres applications.

  C'est aussi dans cette phase que le contrôle quantitatif est effectué sur les procédés. Cela peut engendrer des changements sur les procédés au niveau de différentes phases du cycle de vie.

3-Les caractéristiques de l'agent dans ALLIANCE :

- Les agents sont des entités autonomes et sont définis par les objectifs qu'ils doivent atteindre. Des connaissances sur le procédé sont incluses dans l'agent, aussi il a des capacités individuelles de réalisation d'actions et des capacités d'interaction pour des inter-opérations avec d'autres agents.
- L'agent a un comportement autonome guidé par les objectifs qu'il doit atteindre, pour cela l'agent utilise toutes les capacités dont il dispose pour mener à terme sa mission.
- Les agents sont organisés de manière hiérarchique ce qui permet d'avoir des agents plus au moins spécialisés.
- L'agent peut être composé d'autres agents, l'agent qui implémente une tâche atomique consiste généralement en une invocation d'outils.
- Les agents qui implémentent les mêmes tâches peuvent travailler aussi bien de manière compétitive que de manière coopérative.
- Quand un agent implémente une tâche ses objectifs sont les objectifs de la tâche.

4-L'architecture conceptuelle de ALLIANCE :

L'architecture de ALLIANCE est basée agent, elle est constituée principalement de 3 types d'agents, ces agents sont définis selon le type de tâche qu'ils doivent effectuer :

- **Les agents procédés :** leur rôle est d'implémenter la définition du procédé supporté.
  Il inclut :

  - **Agent interprétation de procédé *:*** il réalise les tâches du procédé en allouant les ressources demandées et en invoquant les outils nécessaires afin d'atteindre les buts fixés.
  - **Agent utilisateurs :** il constitue une interface entre les agents interprétation du procédé et les humains qui sont actuellement responsables des tâches qui doivent être effectuées.

  - **Agent outils :** il constitue une interface entre les agents interprétation de procédé et les outils logiciels utilisés pour effectuer leurs tâches.

- **Les agents infrastructures :**

Ils forment les agents de base pour faire tourner le système entier, il inclut :

- **L'agent déclencheur** *:* il initialise les agents et recherche les ressources dont ils ont besoin pour leur exécution.

- **L'agent gestionnaire d'objets** *:* informe les agents sur les événements qui arrivent dans la base d'objets, cette base peut contenir les modèle de procédés, l'exécution du procédé et leur état d'exécution.

Il contrôle aussi l'accès à la base d'objets et effectue des changements sur ces objets si cela est nécessaire.

- **L'agent intermédiaire (courtier)** *:* il est responsable du routage des informations et des événements aux agents.

- **L'agent chronomètre** *:* il se charge de la génération des événements temporels.

- **Les agents de contrôle quantitatifs :**

Ces agents coopère pour assurer un certain contrôle sur l'exécution du modèle de procédé, car si le modèle de procédé instancié ne permet pas de représenter fidèlement les caractéristiques du projet a réaliser, l'exécution du modèle de procédé risque de dévier et de ne plus respecter le modèle de procédé instancié, ainsi les agents de contrôle quantitatif permettent de surveiller l'exécution du modèle de procédé et d'agir au bon moment pour éviter une déviation trop flagrante qui pourrait faire échouer le projet à réaliser. Ces agents inclus :

- **Agents moniteur :** l'agent moniteur implémente la tâche de contrôle, son but est de détecter d'éventuelles déviations entre les modèles de procédés en exécution et ceux instanciés. La comparaison n'est pas faite globalement mais sur des aspects individuels du procédé, sur lesquels des facteurs de conformité sont traités.

- **Agent décision :** l'agent décision est l'implémentation de la tâche décision sa capacité la plus importante est l'inférence. En commençant par les informations et résultats obtenus de l'agent intermédiaire ; l'agent décision «infère » décide de la classification d'actions possible qui peut être effectuée pour réduire les incohérences entre le modèle de procédé instancié et celui qui est exécuté.

- **Agent changements** *:* il implémente la tâche de changement, son rôle est d'analyser et voir comment implémenter les changements demandés sur le modèle instancié. Si les changements sont effectués l'agent changement doit s'assurer que la consistance du modèle de procédé instancié est maintenue, en d'autres termes, l'implémentation des changements doit être effectuée de manière contrôlée.

Les changements à implémenter peuvent être demandés par l'agent décision ou par l'utilisateur lui-même ce qui permet une certaine flexibilité dans l'exécution du procédé.

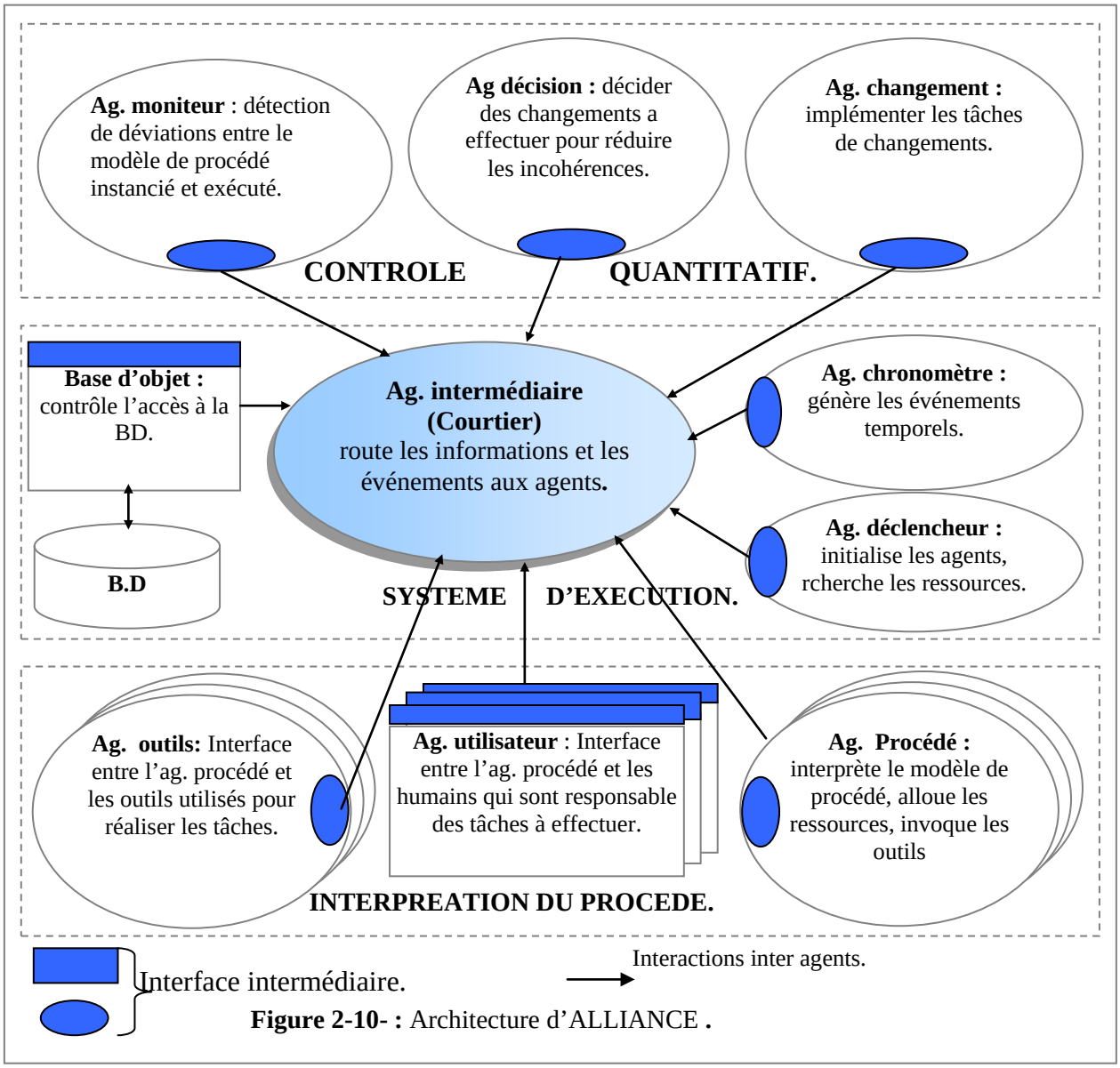


Figure 2-10- : Architecture d'ALLIANCE **.**

III. L'ENVIRONNEMENT CENTRE PROCEDES LOGICIELS PEACE+ : [32]

1- Présentation de l'environnement PEACE+ :

PEACE + **[1]** ,ou *« **P**rocess-centred **E**nactable and **A**daptable **C**omputer-aided **E**nvironment »* est un support pour la coopération dans les environnements centrés procédés logiciels ; il est constitué d'un formalisme de modélisation de procédés logiciels basé connaissances et d'un système qui exécute les modèles de procédés écrits.

L'objectif de PEACE+ est d'améliorer les interactions humaines en :

- Assistant l'ingénieur logiciel dans son travail coopératif,
- Automatisant les interactions (partiellement) durant le procédé de développement de logiciels

PEACE+ utilise le paradigme de l'agent pour assister l'utilisateur dans l'exécution de ses tâches. Ces agents sont décrit comme intentionnels et rationnels. PEACE+ utilise les spécificités de l'agent (telles que les capacités d'interactions ou la possession de connaissances personnelles sur l'environnement) pour supporter et automatiser l'interaction dans l'ECP.

Les modèles de procédés logiciels écrits dans le formalisme de modélisation PEACE+ sont définis en termes de groupe d'utilisateurs ayant un rôle, un ensemble de tâches, et un ensemble d'agents qui assiste l'utilisateur dans la réalisation des tâches du procédé.

L'aspect organisationnel du modèle de procédé est défini par la distribution des rôles des utilisateurs, tandis que l'aspect fonctionnel du procédé logiciel est représenté par les tâches et l'aspect dynamique est représenté par les agents artificiels.

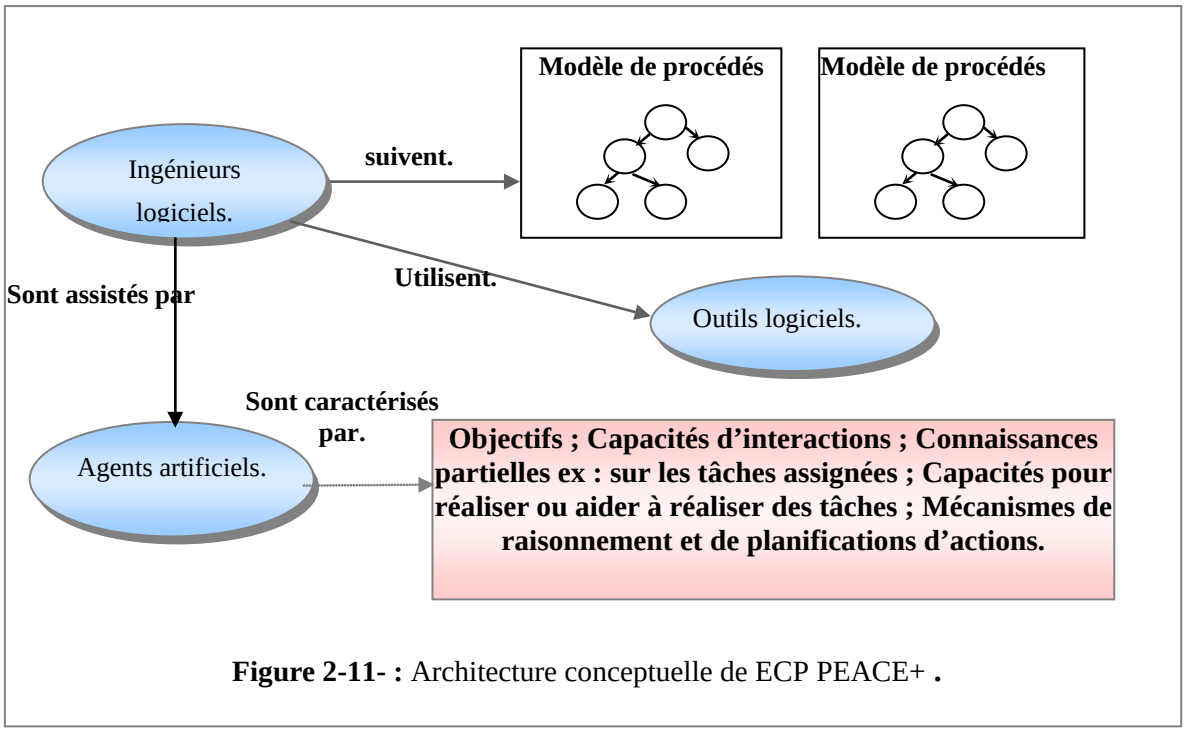


Figure 2-11- : Architecture conceptuelle de ECP PEACE+ **.**

Différents types d'exécution des modèles de procédé sont fournis par PEACE + ; allant du support passif (l'initiative est donnée à l'utilisateur) jusqu'à l'automatisation totale de l'exécution (l'initiative est donnée à l'ECP). Cela pour donner plus de liberté à l'utilisateur en ce qui concerne la décision des parties à automatiser coté interaction, et de respecter la manière de travail de l'utilisateur ; ainsi les solutions suivantes sont fournies :

1. Initiative à l'utilisateur : l'utilisateur sélectionne la prochaine étape de l'interaction et le ECP prend en charge la consistance avec les modèles d'interaction définis.
2. Assistance à l'utilisateur : l'ECP sélectionne les prochaines étapes possibles est l'utilisateur choisi une.
3. Automatisation partiel : l'ECP sélectionne les prochaines étapes possibles, selon une automatisation spécifique il sélectionne une étape et la présente à l'utilisateur pour confirmation.
4. Automatisation totale : c'est l'ECP qui sélectionne toujours la prochaine étape selon le modèle d'interaction.

La plus grande contribution de PEACE+ est qu'il présente une approche intentionnelle pour l'interaction d'agents. En mettant ensemble les concepts de l'interaction et de l'intention permet non seulement de fournir une sémantique aux interactions mains aussi le moyen de les contrôler à travers des pré-conditions, cette solution fournis une automatisation puissante des modèles de procédés permettant l'exécution des tâches les plus complexes qui exige le raisonnement et l'interaction.

2- Définition de l'agent procédé PEACE+ :

Les agents sont des composants artificiels dans l'ECP, ils ont une connaissance partielle sur un domaine, par exemple les tâches et leurs inter dépendances, sur les outils qui doivent être utilisés. Ils sont capables d'exécuter ces tâches automatiquement par le lancement des outils exigés, aussi ils peuvent assister l'utilisateur dans la réalisation de ces tâches « collectives » et leurs interactions. Cela est possible grâce à leur capacité de raisonnement sur les connaissances qu'ils ont sur le procédé et leur capacité de communication avec les autres agents dans l'ECP.

Chaque agent a un ou plusieurs modèles d'interactions constitués d'états et de transitions qu'ils doivent respecter pour la progression du procédé logiciel.

L'agent procédé est défini par :

- Ses connaissances sur les objets et les tâches du procédé.
- Ses capacités en terme d'opérateurs sur les objets.
- Ses capacités d'interactions en terme d'accointances et modèles d'interactions.
- Ses objectifs.

L'agent procédé a des mécanismes de raisonnement, lui permettant la prise des décisions et la planification d'actions.

3- Les concepts clé de PEACE+ : l'intention et l'interaction

La description et la modélisation des interactions d'agents se basent sur quelques idées tirées des deux théories suivantes :

- La théorie des actes de langage, ou la communication est vue comme un acte social impliquant deux agents et exigeant des conventions sociales.
- La théorie de l'action rationnelle ou la communication est vue comme une action psychologique ; tel que son accomplissement est effectué de manière rationnelle par l'agent et cela en respectant ses connaissances, ses capacités et ses intentions.

L'intention est définie comme la sincérité de l'agent à vouloir effectuer un objectif donné. L'objectif est une expression logique qui appartient à la description statique ; alors que l'intention de l'agent peut changer ou être annuler sous certaines conditions, ainsi l'intention appartient à la description dynamique de l'agent.

Pour modéliser les interactions des agents et des utilisateurs ; PEACE+ se base sur le concept d'interaction qui représente le support formel de l'interaction et l'intention pour avoir la sémantique de l'interaction.

Les capacités d'interaction d'un agent sont matérialisées par ses accointances et sa connaissance sur eux, ainsi que les modèles d'interactions.

Les modèles d'interactions contiennent les protocoles d'interaction « possible » de l'agent avec ses accointances ; le modèle est constitué d'un ensemble d'états et d'un ensemble de transitions.

Pour effectuer (envoyer ou recevoir) un acte de communication il faut effectuer une transition ; en d'autre terme passer d'un état initial à un état final, et pour effectuer cette transition il faut que la pré-condition de l'acte de communication soit vérifiée.

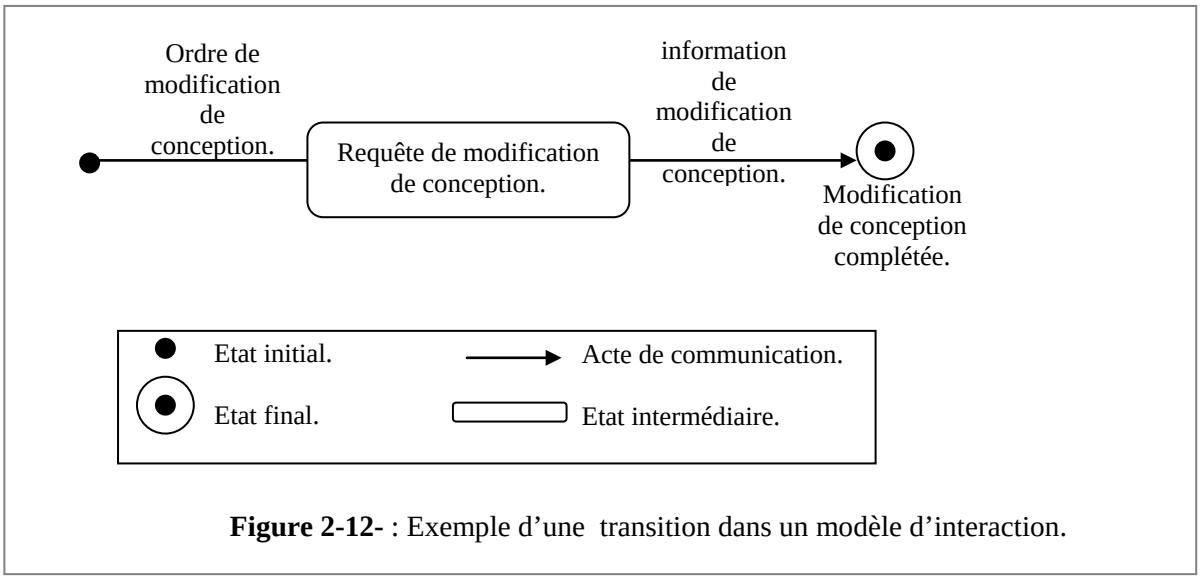


Figure 2-12- : Exemple d'une transition dans un modèle d'interaction.

4- Support d'exécution de l'interaction PEACE+ :

Dans l'exécution de l'interaction des agents, les agents procédés effectuent toujours deux étapes :

Réception de l'acte de communication et son interprétation ; la deuxième étape est la sélection de l'intention qui convient, établir et exécuter un plan d'action pour l'honorer.

L'agent établi un plan d'action en utilisant ses capacités de raisonnement ainsi que ses connaissances sur le domaine concerné.

Entre temps d'autres intentions peuvent être générer, le choix le l'intention à réaliser dépend des connaissances de l'agent ainsi que les stratégies prédéfinis pour la sélection de l'intention adéquate. Par exemple l'agent peut choisir l'intention de haute priorité si toutes les ressources demandées à sa réalisation sont disponibles.

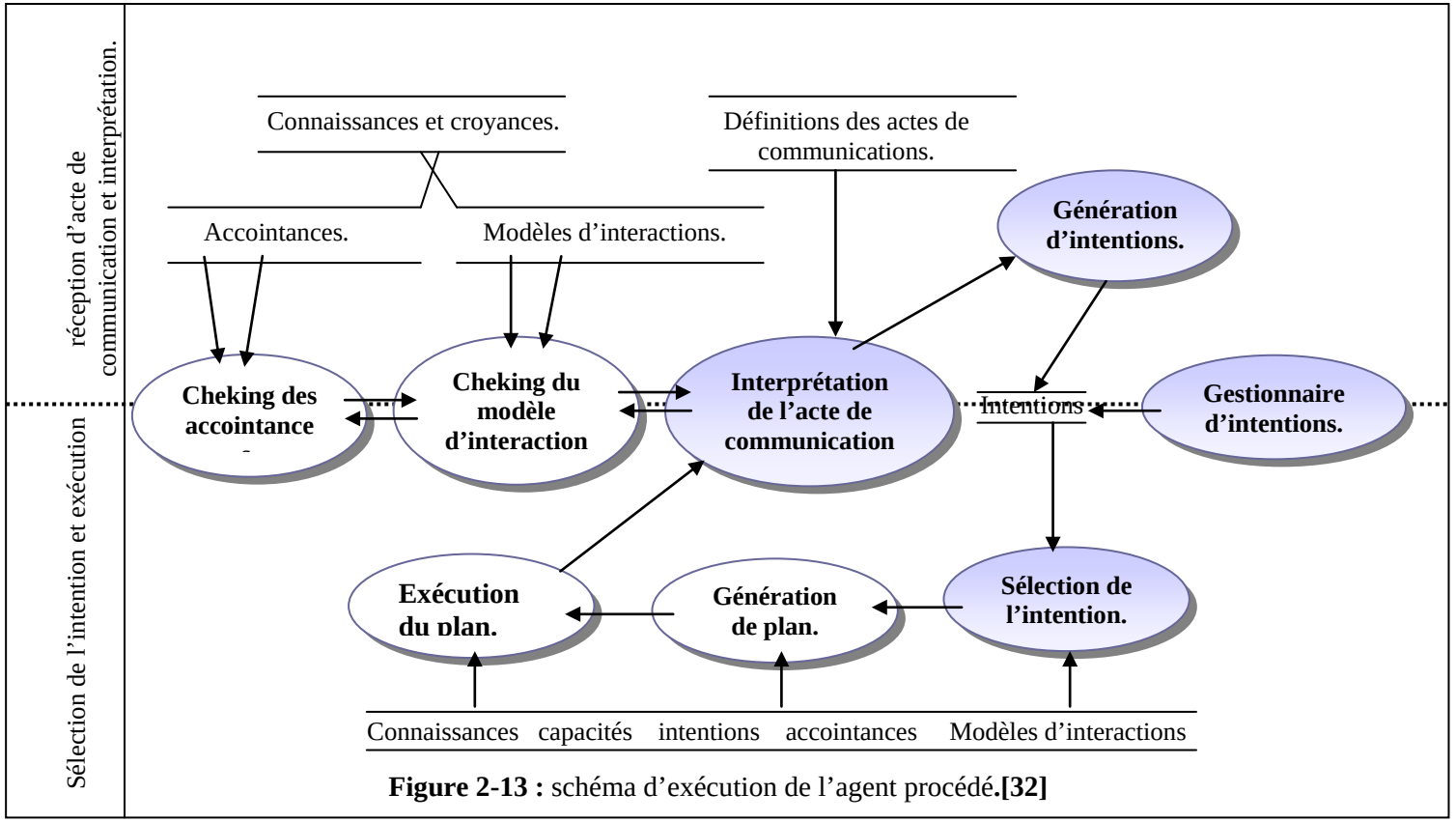


Figure 2-13 : schéma d'exécution de l'agent procédé**.[32]**

V- Gestionnaire de procédés business basé technologie d'agents (ADEPT):

Il existe de grandes similitudes entre les environnements d'ingénierie logiciels et les systèmes gestionnaire de procédés business : Les deux constituent des supports pour la gestion et l'exécution des procédés (logiciels ou business).
Chaque technologie peut s'inspirer de l'autre et appliquer ses concepts ; il y a même des chercheurs qui pensent que les procédés logiciels peuvent être exécutés dans des systèmes workflow et avoir de bons résultats**[30]**.

Contrairement aux environnements centrés procédés logiciels, beaucoup de gestionnaires workflow basés technologie d'agent existent ; pour avoir un état de l'art complet il serait intéressant d'avoir une idée sur le fonctionnement d'au moins un système gestionnaire de procédés business.

Un seul système gestionnaire de procédés business basé agents sera présenté, qui est le système ADEPT. Le choix de ce système ne sait pas fait par hasard ; il s'est basé sur le faite que ce dernier doit remplir les même conditions qu'un ECP tel que la distribution, la flexibilité, la dynamicité dans l'exécution des procédés ; il doit supporter des procédés business qui ont les caractéristiques d'un procédé logiciel tel que : la distribution, imprévisibilité, les changements dynamiques du procédés, possibilités de déviations...etc.

1- Définition du système ADEPT[11][12][13] :

ADEPT(***A**dvenced **D**ecision **E**nvironment for **P**rocess **T**ask*) est un système gestionnaire de procédés business utilisant l'approche basée agent pour supporter l'exécution et la gestion des procédés business.

Dans le projet ADEPT le procédé business est vu comme une collection d'entités résolveurs de problèmes autonomes qui communiquent, négocient avec d'autres entités dans le but d'offrir ou de recevoir un service et pour pouvoir coordonner l'exécution des tâches dans le système avec tous ce que cela comporte comme contrôle, allocation, communication…etc.

Le but de ce système est de décentraliser la responsabilité et la prise de décision pour avoir plus de dynamicité et de parallélismes dans l'exécution des tâches. Aussi cela rendra plus facile la gestion des procédés business si le système est géographiquement distribué.
Les concepts de base D'ADEPT sont :
Le service : correspond à une unité conceptuelle d'activité résolveur de problème dans le procédé business ; dans ADEPT on parle de service et pas d'activité ou de tâche.
L'agency : Consiste en un agent responsable unique, et un ensemble de tâches que l'agent est responsable d'exécuter et un ensemble de sous agency qui sont sous la direction de l'agent responsable.

2- Les fonctions de l'agent ADEPT :

Tous les agents ADEPT ont la même architecture : ils sont constitués d'un nombre de modules fonctionnels distincts –**figure2-14-**, chaque module est responsable d'un certain aspect du système, tel que la communication, l'exécution des services, l'estimation des la situation…etc. Ces modules travaillent ensemble, s'échangent des informations pour une meilleure exécution des services (temps, coût et qualité). Ces modules sont :

- **Modèle personnel et le modèle d'accointances : Self Model (SM), Accointances Model (AM)**

Le modèle personnel ou d'accointances sont des bases de stockage de connaissances des agents , ils contiennent des connaissances sur eux même, leurs voisins et leurs environnement. -

Le modèle personnel contient des informations sur les services que l'agent peut fournir, les ressources disponibles et celles qui sont occupées…etc.
- Le modèle d'accointances contient les informations sur les autres agents, états, les services qu'ils peuvent fournir…etc.

- **Module gestionnaire d'interaction : Interaction Manager Module (IMM)**

Il est responsable de la fourniture des services que demandent l'agent à travers « la négociation » avec les autres agents. Il décide des services que l'agent peut fournir et des conditions à respecter.
Le processus d'obtention de service d'un autre agent est initialisé par le **SAM** .
C'est le IMM qui est responsable de déterminer quel agent contacter et quel stratégie de négociation utiliser.
Pour remplir son rôle le IMM prend ses décisions, fait ses choix en se basant sur 4 informations :
- Les contraintes de scheduling émanant du **SAM.**
- Les connaissances sur l'agent : ses priorités, préférences, le prix des services représentés dans le SM (Self Modèle).
- Les résultats des négociations précédantes ou courantes représentés dans **SM**.
- Les capacités, états des autres agents représentés dans le **AM.**

Le processus de décision pour «les services à fournir pour un autre agent » est initialisé par la réception d'une proposition de l'agent qui serra traitée, et négociée jusqu'à l'arrivé à un accord (acceptation, refus…).

- **Module d'évaluation de situation : Situation Assensement Module (SAM)**

Sa responsabilité est d'estimer, contrôler les capacités de l'agent à respecter les contrats établies. Cela exige plusieurs activités tel que : la gestion des allocations des services et tâches de l'agent, ainsi que la prise en charge de toutes exception ou déviation qui peut se produire pendant l'exécution des tâches.

En cas de déviation le SAM essaye programmer des tâches ou services pour minimiser la déviation et régler le problème. Si cela ne suffis pas une deuxième négociation est demandée.

- **Module exécution de services : (SEM)**

Sa fonctionnalité principale est la gestion des activités pendant leur exécution. cela exige :
- Les résultats du SDL pour pouvoir respecter les décisions des autres modules.
- La gestion, routages des informations entre les tâches et services et les informations des autres agents pendant l'exécution.
- La prise en charge des exceptions de bas niveau, en d'autres termes réagir en bon moment et correctement à toutes les exceptions qui peuvent se présenter pendant l'exécution des tâches.

- **Module communication : Communication Module (CM)**

Il est responsable de :
- Stockage des messages échangés entre agents dans un langage et format d'information partagés, il s'occupe de la réception et l'interprétation des messages des autres agents, et de leurs tâches durant la gestion des tâches(activation, suspension, annulation …etc.).

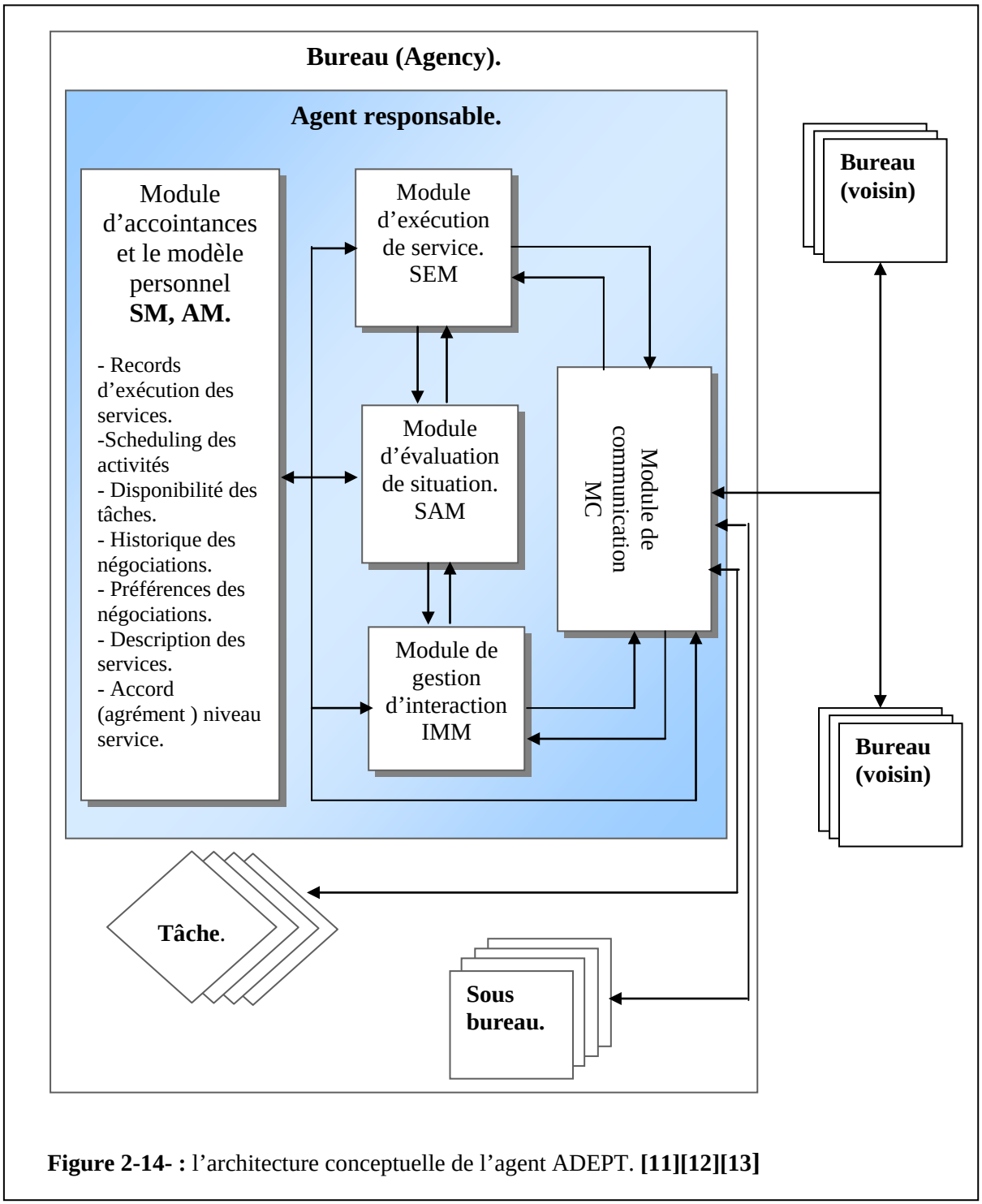


Figure 2-14- : l'architecture conceptuelle de l'agent ADEPT. **[11][12][13]**

Conclusion :

Dans ce chapitre nous avons présenté les plus importants environnements d'ingénierie logiciels à base d'agents intelligents; la plupart de ces systèmes donnent la priorité à l'exécution du modèle de procédé ou à l'interaction de ces composants. Les systèmes pour la modélisation des procédés logiciels à base d'agents intelligents sont plus rares, le peut de publication trouvées sur la modélisation à base d'agents ne sont pas convaincantes **[31]** car le concept d'agent est exploité de la même manière que le concept objet. (Voir Annexe).

Introduction :

Afin d'avoir une synthèse minutieuse des environnements et approches étudiées ; il serrait pertinent d'établir un tableau qui englobe tous les aspects qui caractérisent un ECP, et qui sont susceptible d'être pris en charge par des agents logiciels. Ce tableau doit comporter les agents qui composent les systèmes étudiés et l'aspect pris en charge par ces derniers.

Un autre tableau est établi pour la comparaison et l'évaluation de l'efficacité de ces agents logiciels.

Selon les systèmes étudiés les aspects qui peuvent être prie en charge par les agents logiciels sont :

- **L'interaction** : Cela inclus la communication et tous ces problèmes tel que l'envoi des messages, le format des messages utilisés …etc, la négociation pour l'utilisation des ressources existantes, la coordination entre les fonctionnements des agents…

- **Les ressources** : outils ; composants utilisés, ressources humaines ou matérielles, allocation de ces ressources, la durée de l'allocation …etc.

- **Les données** : leurs formats, stockages, accès, mise à jours…etc.

- **L'espace organisationnel** : La localisation et le déplacement de tous agents, outils, composants, fragments de procédés ou les utilisateurs si la mobilité est prise en considération. Ainsi que l'extension de l'environnement de développement.

- **Assistance de l'utilisateur** : Aider l'utilisateur a suivre le procédé instancié a effectuer les activités demandées.

- **La consistance** : Est la cohérence des données manipulées, des activités exécutées ou bien la consistance des fragments de procédés dans le cas ou le procédé est dupliqué, découpé ou manipulé de manière à ce qu'on perde la version originale du procédé instancié.

- **La modélisation du procédé logiciels** : Même s'il est impossible de se passer de la réflexion humaine pour réaliser cette tâche, il est possible de fournir des outils pour aider l'être humain à modéliser, introduire des changements ou gérer la cohérence du modèle de procédé logiciel.

- **Exécution du procédé** : l'exécution des tâches, activités programmées du procédé logiciel.

- **Le contrôle de l'exécution du procédé** : De manière à ce que le procédé instancié soit le plus possible ressemblant au procédé exécuté.

- **La mobilité** : la mobilité par exemple des agents ou code n'est pas primordiale dans un ECP ;cependant, elle peut régler beaucoup de problème et donner certains avantages lors de la recherches des ressources par exemple.

2- Les agents logiciels agissant sur les environnements étudiés :

Ce tableau est un résumé des agents intelligents des 3 systèmes étudiés ainsi que leurs fonctions :

<table>
<tr><th colspan="2">Environnements centré procédés logiciels basés agents.

Aspects pris en charge par les agents.</th><th>Environnement CAGIS.

Approche basée interactions</th><th>La structure ALLIANCE.

Approche basée objectifs</th><th>L’environnement PEACE+.

Approche basée connaissances</th></tr>
<tr><td rowspan="3">Interaction.</td><td>Communication.</td><td>- Communiquant,
- Facilitateur.</td><td>- Intermédiaire (courtier).</td><td rowspan="12">▪ Les agents de PEACE+ ne sont pas spécialisés.

▪ Ils ont tous les même caractéristiques.

▪ La différence entre eux est dans leurs objectifs.

▪ Leurs objectifs dépendent du rôle de l’utilisateur qu’ils assistent

▪ Leur type dépend de l’utilisateur qu’il assiste et de son rôle. Par exemple agent concepteur en référence à l’ingénieur concepteur.</td></tr>
<tr><td>Négociation.</td><td>- Négociateur,
- Médiateur.</td><td rowspan="2">Les agents s’occupent eux même de leurs négociations</td></tr>
<tr><td>Coordination.</td><td>- Coordination.</td></tr>
<tr><td colspan="2">Cohérences et consistances des fragments de procédés.</td><td>- Votant,
- Négociateur,
- Mise à jour,
- Coordination,
- Notification.</td><td>(les procédés ne sont pas fragmentés)</td></tr>
<tr><td colspan="2">Ressources. (matériels ou humaines.)</td><td>- Négociateur.
- Médiateur.</td><td>- Outils</td></tr>
<tr><td colspan="2">Espaces de stockages (données et base de données).</td><td>- Espace de stockage.
Médiateur.</td><td>- Gestionnaire d’objets.</td></tr>
<tr><td colspan="2">Modélisation du procédé logiciels.</td><td>-</td><td>-</td></tr>
<tr><td colspan="2">Exécution du procédé.</td><td>-</td><td>- Interprétation de Procédés.
- Déclencheur.
- Chronomètre.</td></tr>
<tr><td colspan="2">Le suivie de l’exécution du procédé.</td><td>-</td><td>- Moniteur.
- Décision.
- Changement.</td></tr>
<tr><td colspan="2">Places(organisation).</td><td>- Gestionnaire.</td><td>-</td></tr>
<tr><td colspan="2">Mobilité de composants. (agents, procédés)</td><td>- Mise à jour</td><td>(Pas de mobilité).</td></tr>
<tr><td colspan="2">Assistance utilisateur.</td><td>- Local,
- Utilisateur.</td><td>- Utilisateur.</td></tr>
</table>

3- Performances des agents logiciels des environnements étudiés :

Environnements basés agents. Aspects pris en charge par les agents.		Environnement CAGIS. Approche basée interactions	La structure ALLIANCE. Approche basée objectifs	L'environnement PEACE+. Approche basée connaissances
Interaction.	**Communication.** 1	+++	++	+++
	Négociation.	+++	++ (pas beaucoup de diversité côté historique et stratégies) 7	+++
	Coordination. 2	+++		+++
Cohérences et consistances des fragments de procédés. 3		++(cas limité)	–	Dépend de l'efficacité de distribution des rôles sur les développeurs.
Ressources. (Matériels ou humaines.)		++	++	
Espaces de stockages (Données et base de données).		+++	++	
Modélisation du procédé logiciels. 4		–	–	
Exécution du procédé.		–	+++	
Le suivie de l'exécution du procédé. 5		–	+++	
Espaces (organisation).		++	–	
Mobilité de composants. (agents, procédés) 6		++	–	
Assistance utilisateur.		+++	+++	+++

1 Dans l'environnement CAGIS l'interaction est prise en charge de manière très efficace ; plusieurs types d'agents spécialisés dans un certain type d'interaction tel que la communication ou la négociation sont définis ; cela rend facile et plus rapide la prise en main de tous type d'échange d'information « locale ou globale » ,même si les formats des messages ou des protocoles sont hétérogènes.

Ce n'est pas spécialement le cas pour la structure ALLIANCE car pour la communication par exemple il n y a qu'un seul agent «courtier » qui se charge de toutes les interactions : Toute communication, échange ou transaction passe par l'agent courtier, ainsi il peut être surchargé et dépassé s'il y a un grand trafic d'informations.

2 Dans la ALLIANCE les agents s'occupent eux même de leurs négociations, l'avantage et qu'il ne faut pas un intermédiaire pour régler ses conflits. L'inconvénient est que ce n'est pas des agents spécialisés dans la négociation et n'ont pas une connaissance approfondie des stratégies de négociations, contrairement aux agents négociateurs de l'ECP CAGIS.

3 La gestion de la consistance des fragments de procédés n'est pas très efficace car il faut qu'il y est une solution à n'importe quel cas d'incohérences qui se présente, or les

solutions proposées concerne quelques cas seulement, ce qui n'est pas acceptable pour un puissant ECP.

4 La modélisation n'est effectuée par aucun agents mais elle est prise en charge d'une autre manière.

5 Dans CAGIS l'exécution des procédés logiciels n'est pas effectuée par des agents mais par un moteur d'exécution locale. L'inconvénient est que le suivie de l'exécution du procédé n'est pas effectuée de manière efficace ; s'il y a une déviation, il faut envoyer un rapport au « **glue serveur »** qui va consulter le « **modèle glue »** pour prendre une décision et implémenter des changements si c'est indispensable. Ces opérations manquent de flexibilité et de dynamicité, cela peut se répercuter négativement sur le temps d'exécution du modèle de procédé.

Dans la structure ALLIANCE, un système très strict est mis en place pour surveiller l'exécution du procédé logiciel ; s'il y a une déviation quelconque, les agents de contrôles évaluent selon des facteurs d'évaluations prédéfinis, décident et implémentent des modifications. Cela est fait de manière très dynamique et automatique, ce qui donne un grand avantage à ce système par rapport à CAGIS.

6 La mobilité des agents, des outils ou des procédés logiciels n'est pas supportée dans la structure ALLIANCE, mais elle est prise en charge par CAGIS. Certes la gestion de la mobilité demande plus de techniques et d'agents pour localiser le placement des composants ou mettre à jour les adresses de résidences des composants ; cependant elle donne un avantage non négligeable qui est une meilleure exploitation des ressources existantes (humaine ou matérielles) ; qui ne sont pas forcément dans l'espace de travail ou s'exécute le fragment de procédé demandeur de ces ressources.

7 Dans l'environnement PEACE+ il n y a pas d'agents spécialisés pour la gestion de l'interaction. L'interaction est un concept de base mais elle est gérée d'une autre manière ; tel que chaque agent possède des modèles d'interactions constitués d'un ensemble de transitions ; ces derniers décrivent tout type d'interaction susceptible d'arriver entre l'agent et son environnement.

4- Evaluation de la qualité des systèmes étudiés :

Ce tableau résume les caractéristiques générales des systèmes étudiés et le dégrée d'efficacité des agents logiciels à les prendre en charge :

Environnements basés agents. Caractéristique générale de ECP		**Environnement CAGIS.** **Approche basée interactions**	**La structure ALLIANCE.** **Approche basée objectifs**	**L'environnement PEACE+.** **Approche basée connaissances**
-1- Interaction.	- Communication - Négociation. - Coopération. - Coordination.	+++	++	+++
-2- Distribution.	- Des espaces de travail, - Des outils, - Des ressources -Des développeurs.	+++	+	Selon l'efficacité des développeurs.
-3- Hétérogénéité.	- Des outils -Des langages de communication - format des messages	+++	+	
-4- Dynamicité. (coté prise en charge de changements)	- Prise en charge dynamique des changements. - Possibilité d'effectuer des changement pendant l'exécution de modèle de procédé.	+	+++	
-5- La mobilité des composants	Des fragments de procédés, Des agents logiciels, Des outils…	++	_	
-6- La consistance.	Des données Des activités	+	+++	
-7- Modélisation du procédé logiciels	- Cohérence de la représentation -Introduction des changements pendant l'exécution	**La modélisation des procédés logiciels n'est pas effectuée par les agents.**		

Après l'étude détaillée des environnements CAGIS, ALLIANCE, PEACE+ et ADEPT et la comparaison de ces systèmes, nous avons déduit que chaque système est adapté à un certain type de développement de logiciels.

Ainsi, l'ECP CAGIS est plus adapté au développement de logiciels sur un site étendue, distribué géographiquement. Il est adapté à des projets ou le facteur temps n'est pas une «ressource critique » ; les outils, techniques de développement, langage de communication peuvent être hétérogènes.

La structure ALLIANCE est plus adaptée au développement ou le facteur temps est important, ou la communication est réduite et utilise des formats homogènes.
L'espace de travail ne doit pas être très distribué car les moyens de communication ne sont pas très développés par rapport à ceux de ECP CAGIS.

L'environnement PEACE+ donne plus de liberté aux développeurs, car ils peuvent utiliser l'environnement de manière manuelle, semi-automatique ou automatique. Il est adapté au développement de logiciels ou le facteur humain est très important ; car il lui donne la possibilité de prendre des décisions et lui facilite l'interaction à travers les modèles d'interactions définis auparavant.

L'environnement ADEPT a une autre approche pour traiter les procédés «pas forcément logiciels ». Il n y a pas d'agents spécialisés, un seul type d'agent est défini, chacun est responsable d'un certain nombre de tâches «mais pas un certain type de tâche ». Chaque agent évolue selon les tâches ou activité qui lui ont été assignées, il gère tous les aspects tel que la communication, négociation, stockage des données et des connaissances lui-même.
L'avantage est que cette approche donne plus de liberté à l'utilisateur pour assigner ses activités ; de plus il est pus facile de faire apparaître la structure organisationnelle de l'entreprise et d'assigner les tâches selon cette organisation. Les même avantages peuvent être attribuer à l'environnement PEACE+ du fait que ses agents ne sont pas spécialisés.

Les recherches effectuées concernant l'utilisation des agents dans l'ingénierie des logiciels se sont surtout soldé par le développement d'environnements et de structures centrées procédés logiciels, l'objectif des agents est de prendre en charge tous ce qui entoure le procédé logiciels pour qu'il s'exécute dans les meilleures conditions. Ainsi la gestion des ressources et des outils, l'accès et le stockage des données, la gestion des espaces de travail sont des objectifs de l'agent logiciel.
D'un autre côté, il faut souligner que la modélisation du modèle de procédé n'a pas eu beaucoup d'intérêt ; la plus part du temps, le modèle de procédés pouvait être écrit avec n'importe quel langage ; cela n'a pas beaucoup d'influence puisque le but était de fournir l'environnement de l'exécution et pas de manipuler la modélisation. La plus part des capacités de l'agent tel que : l'autonomie, l'actions, la réaction et l'interaction ont été bien exploité ; cependant, les capacités représentationnelles ont été négligées pénalisant ainsi l'émergence de solutions concernant la modélisation de procédés logiciels .

L'exécution et la modélisation du procédé logiciels sont traitées de manière séparée ; La dynamicité et la flexibilité de la modélisation des procédés logiciels n'est pas une priorité alors que c'est un objectif concernant l'exécution ; l'idéal serrait de mettre la modélisation et l'exécution du modèle de procédé logiciels sur le même niveau d'intérêt ; d'exploiter en plus des autres caractéristiques de l'agent, ses capacités représentationnelles, en alimentant par exemple sa base de connaissances avec son fragment de procédé, son rôle serrait de « gérer » sa modélisation et son exécution.

5- Exploitation des capacités de l'agent par les systèmes étudiés :

Capacité de l'agent	L'environnement CAGIS.	La structure ALLIANCE.	L'environnement PEACE+.
Interaction.	+++	+++	+++
Autonomie.	+	++	+
Mobilité.	++	—	—
Action.	+++	+++	+++
Représentationnelles. (Pour la modélisation du procédé logiciels).	—	—	—
Perception de l'environnement.	+	++	+
Raisonnement et planification.	+	++	+
Activité continue	—	++	—

+++ :Bien exploité **++ :Exploité** **+ : Mal exploité** **— : Non exploité**

Les recherches effectuées concernant l'utilisation des agents dans l'ingénierie des logiciels se sont surtout soldé par le développement d'environnements et de structures centrées procédés logiciels, l'objectif des agents est de prendre en charge tous ce qui entoure le procédé logiciels pour qu'il s'exécute dans les meilleures conditions. Ainsi la gestion des ressources et des outils, l'accès et le stockage des données, la gestion des espaces de travail sont des objectifs de l'agent logiciel.

D'un autre côté, il faut souligner que la modélisation du modèle de procédé n'a pas eu beaucoup d'intérêt ; la plus part du temps, la modélisation du procédé logiciels n'était pas prise en charge par les agents, le modèle de procédé pouvait être écrit avec n'importe quel langage ; cela n'a pas beaucoup d'influence puisque le but de l'agent était de fournir l'environnement de l'exécution et pas de régler des difficultés de modélisation. La plus part des capacités de l'agent tel que : l'autonomie, l'action, la réaction et l'interaction ont été bien exploité ; cependant, les capacités représentationnelles ont été négligées pénalisant ainsi l'émergence de solutions concernant la modélisation de procédés logiciels.

L'exécution et la modélisation du procédé logiciels sont traité de manière séparée ; la dynamicité et la flexibilité de la modélisation des procédés logiciels n'est pas une priorité alors que c'est un objectif concernant l'exécution ; l'idéal serrait de mettre la modélisation et l'exécution du modèle de procédé logiciels sur le même niveau d'intérêt ; d'exploiter en plus des autres caractéristiques de l'agent, ses capacités représentationnelles, en alimentant par exemple sa base de connaissances avec son fragment de procédé, son rôle serrait de « gérer » sa modélisation et son exécution en même temps.

L'idée que nous développerons est de modéliser le procédé logiciels sous forme d'un système multi-agents ; tel que l'agent représentera un fragment de procédé, son rôle est de gérer sa modélisation, de prendre en charge toute modification, et de fournir les conditions nécessaires à son exécution.

I- Introduction :

D'après les travaux effectués dans le génie logiciel, le concept d'agent a été utilisé particulièrement pour le développement d'environnements de programmations et précisément pour l'exécution des modèles de procédé logiciels. Les travaux se sont focalisés sur l'exploitation des capacités d'interactions et la possibilité de distribuer les tâches, par spécialisation ou par redondances des agents. Par contre, peu de travaux ont été effectués concernant la modélisation des procédés logiciels basés agents **[13]**.Les capacités représentationnelles des agents sont ainsi faiblement mises à profit.

Nous nous proposons dans ce qui suit de concevoir un modèle de procédé sous forme d'un système multi-agents hiérarchique doté d'outils, modèles et connaissances nécessaires dans le but de fournir un procédé logiciel dynamique distribué et évolutifs.

La comparaison succincte des caractéristiques «générales » des systèmes multi-agents et des modèles de procédé logiciels fait apparaître plusieurs points en commun qui montrent ainsi les possibilités de représenter les modèles de procédés par un système multi-agents. Ces points peuvent être résumés comme suit :

Les caractéristiques du modèle de procédés.	Ce qui peut être sa représentation dans un SMA
▪ Modèle de procédé.	▪ Système multi-agents.
▪ Activité.	▪ Agent : son objectif est de réaliser cette activité. Si l'activité est complexe l'agent qui la réalise serrait aussi complexe.
▪ L'activité a besoin de ressources.	▪ L'agent possède et utilise des ressources personnelles ; il a des mécanismes tel que la négociation pour avoir celle qui lui manque.
▪ Les activités ont des liens de précédence, elles peuvent s'exécuter Séquentiellement ou en parallèles.	▪ Ces liens peuvent être modéliser par les modèles d'interactions de l'agent afin de respecter l'ordre d'exécution de ces activités.
▪ Les activités «peuvent » s'exécuter dans des espaces de travail différents.	▪ Les agents sont par nature distribuée et peuvent assurer cette condition sans problème.
▪ Les activités utilisent des informations en entrés et produisent des informations en sortie.	▪ Les agents utilisent et produisent des informations et des connaissances.
▪ Le modèle de procédé logiciel peut être mobile.	▪ Les agents qui représente les activités du modèle de procédé peuvent être mobiles.
▪ Le modèle de procédé logiciel peut être évolutif : il permet d'introduire des changements après le début de son exécution	▪ L'agent à des capacités de perception de son environnement et de planification, il peut ainsi introduire tous changement nécessaire.

II- Objectifs du modèle de procédé orienté agents logiciels :

Pour réussir la conception du modèle de procédé orienté agents, nous nous somme fixer les objectifs suivants :

- Pour que le travail distribué soit faciliter, le modèle de procédé sera distribué, et donc fragmenté en un ensemble de fragments où chaque partie du modèle va s'exécuter dans un espace de travail. La fragmentation du modèle de procédé doit être effectuée par un agent en respectant des règles qui vont être étudiés ultérieurement.

- Représenter le modèle global ou partiel du procédé logiciel dans la base de connaissance de l'agent,

- Respecter les spécificités locales des espaces de travail et donner une liberté de décision locale ; cela en initialisant des agents qui ont une perception de leur environnement limité à leurs espaces de travail, ce qui se passe de l'autre côté ne doit pas être leurs «problème ».

- Développer des mécanismes de prise en charge de toute déviation ou exception qui peut se présenter pendant l'exécution du modèle de procédés. Tous les agents auront la responsabilité de surveiller en permanence l'exécution des tâches et introduire des changements dans le modèle de procédé logiciels si cela s'avère nécessaire.

- Le contrôle et la prise de décision doivent être distribué mais non pas anarchique ; il faut que le contrôle de l'exécution des activités soit structuré. Notre choix est porté sur une structure hiérarchique ; chaque niveau de l'hiérarchie a certains type de contrôle a effectuer et certain type de décisions à prendre.

III. L'analyse fonctionnelle :

*« L'analyse fonctionnelle d'une organisation conduit à l'identification des principales fonctions que les composants de cette organisation doivent remplir ; à ce niveau l'objectif de l'analyse est d'identifier les agents, leurs rôles et leurs objectifs dans le système »***[4]**.

Dans notre approche nous définissons les fonctions de système multi-agents modèle de procédé à travers les types d'agents suivants.

1- Les agents du modèle de procédés logiciels.

- **Agent superviseur :** Son rôle est de modéliser et de stocker le procédé logiciel global, de le fragmenter et l'affecter aux agents fragments dans les différents espaces de travail pour qu'ils s'exécutent. Il a une vue globale de l'exécution des activités du modèle de procédé dans tous les espace de travail ; ainsi son rôle est de garder la cohérence et la consistance du procédé logiciel.

  L'agent superviseur est en interaction directe avec le chef de projet «humain » ; ainsi il est possible à l'être humain d'avoir une vue globale de l'exécution du modèle de procédés ou d'introduire des changements même pendant l'exécution.

- **Agent fragment :** cet agent est localisé dans un espace de travail et possède le fragment de procédé logiciels qui doit s'y exécuter.

Son objectif est de gérer l'exécution de son fragment de procédé ; son rôle consiste à assigner les activités du modèle de procédé sur les différents agents tâches qu'il initialise pour l'exécution de ses tâches.

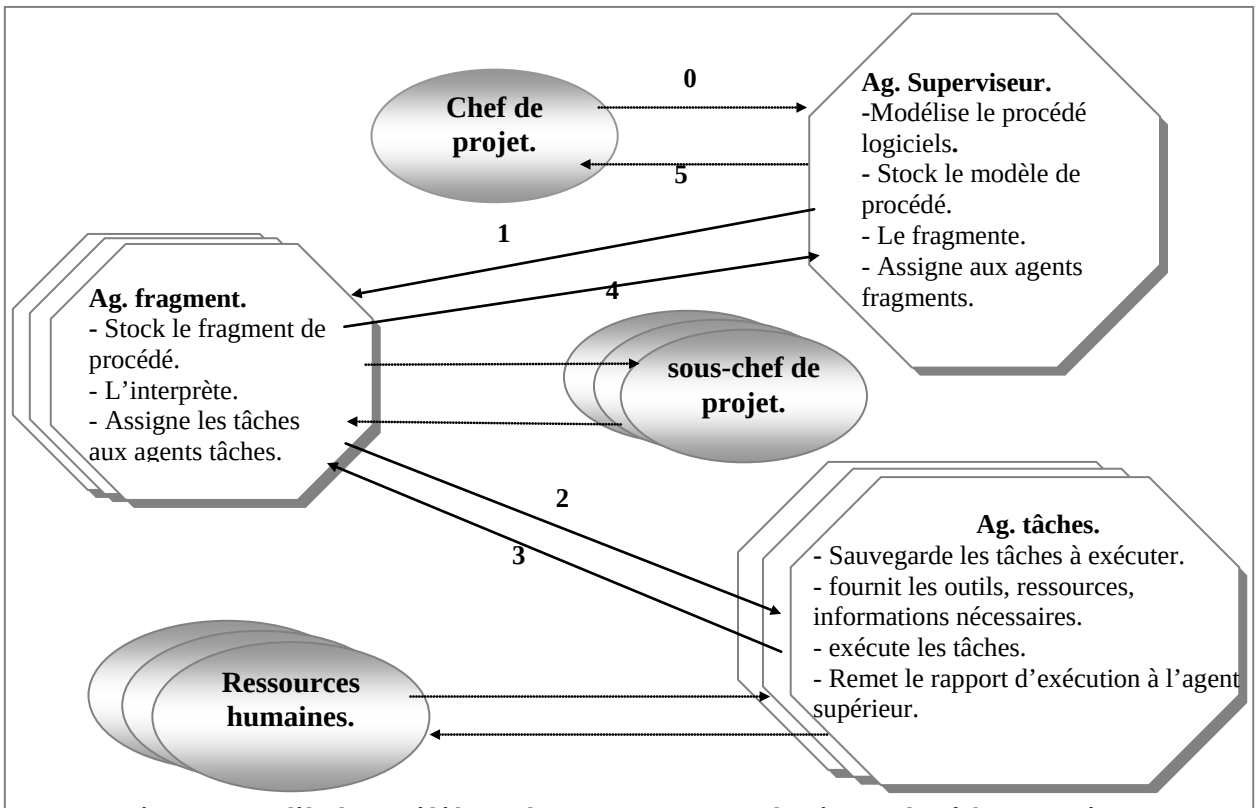
L'agent fragment a une interface avec le sous-chef de projet (le premier responsable de l'espace de travail). Ainsi cela permet au responsable «humain » d'introduire des changements dans le modèle de procédés si cela s'avère indispensable.

- **Agent tâches :** Son rôle est de réaliser les activités du modèle de procédé qu'on lui a assigné ; de fournir les ressources requises, soit directement si elles sont disponibles à son niveau ou par demande de service ou négociation dans le cas contraire, et de les exécuter. A la fin de l'exécution un rapport d'exécution est rendu à l'agent fragment.

L'agent en haut de la hiérarchie a une vue globale du modèle de procédés sans avoir les détails de l'exécution des tâches. Il peut toutefois avoir accès à toute information du modèle de procédé en exécution.

Les agents en bas de la hiérarchie sont des exécutants des tâches du modèle de procédés ; leurs décisions ne concerne que leur espace de travail dans lequel ils ont cependant le plus d'autorités car ayant une vue plus précise de l'exécution du procédé. Leurs décisions sont ainsi prioritaires par rapport à celles des agents hiérarchiques du SMA.

Contrairement aux fonctions de modélisation et d'exécution des procédés logiciels, le contrôle de la cohérence du développement, la consistance des données et le suivie de l'exécution du modèle de procédé (vérifier que le modèle instancié et bien le même que celui qui est exécuté) sont les responsabilités de tous les agents, à tous les niveaux de la hiérarchie. Chaque niveau de la hiérarchie a ses propres facteurs d'évaluations (facteurs temps, coût…), des capacités de raisonnement afin évaluer la cohérence du développement ; chaque niveau a aussi la possibilité d'introduire des modifications dans le modèle de procédé logiciels pour entretenir la consistance du développement.



0 :caractéristiques Modèle de procédé logiciels
1 : Fragment de procédé.
2 :les tâches à exécuter.
3 : Rapport d'exécution des tâches assignées.
4 : Rapport d'exécution du fragment de procédé assigné.
5 : Rapport d'exécution du modèle de procédé.

Figure 4-1- : Distribution et exécution du modèle de procédé logiciels.

2- Spécialisation et redondance des agents modèle de procédé logiciels.

Afin d'obtenir une distribution efficace des rôles sur les agents, nous utiliserons une décomposition des taches suivant une méthode «hybride » basée sur une décomposition «verticale» et une décomposition «horizontale ».

- Dans la décomposition verticale, les procédures ou les fonctions auront leur propre agent comme, par exemple, l'évaluation et la prise en charge de la cohérence du modèle de procédé logiciels,
- Dans la décomposition «horizontale », chaque «type » d'activité est assignée à un agent comme, par exemple des fonctions interactionnelles ou représentationnelles.

  Dans notre cas, la modélisation est effectuée par l'agent superviseur et l'exécution par les agents tâches.

Pour la conception de notre SMA modèle de procédé, nous avons opté pour un degré de spécialisation et un taux de redondances élevé mais pas maximale. Ces taux ne sont pas fixes et peuvent augmenter ou diminuer selon les besoins du système.

Notons que le SMA est hiérarchique. Les agents de chaque niveau de la hiérarchie ont un taux de spécialisation très faible et un taux de redondances très élevé, en d'autre terme les agents d'un même niveau ont la même structure, les mêmes rôles et les même objectifs.

La spécialisation des agents est plus explicite si on analyse le SMA de manière verticale ; car chaque niveau à ses propres fonctions et ses propres activités.

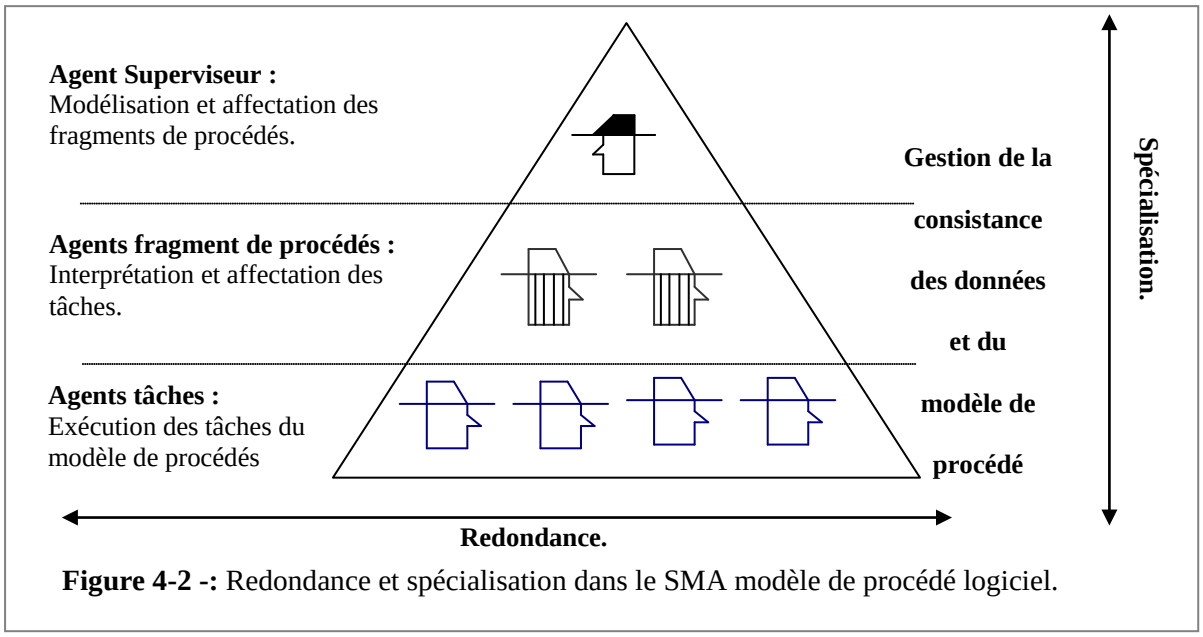


Figure 4-2 -: Redondance et spécialisation dans le SMA modèle de procédé logiciel.

IV. L'analyse structurale :

*« L'analyse structurale tente de donner un ordre à l'ensemble des interactions possibles entre agents en dégageant les relations abstraites qui les relie et la manière dont elle évolue au cours du temps »***[4]**.

Elle permet de définir le type de l'organisation désiré. Ainsi l'organisation peut être hiérarchique telle qu'illustrée par les relations « maître esclave » entre agents ou égalitaire lorsque les agents ne donnent pas d'ordre mais demandent des services. Ces relations peuvent être fixes lorsque les interactions sont connues et prédéfinies, et que le changement n'est pas permis , ou évolutives avec le changement du système et des rôles des agents.

Notre système agents modèle de procédé va être hiérarchique évolutif. Hiérarchique, tel que la distribution des fragments de procédés et des tâches est sous forme d'ordre de l'agent supérieure vers ses agent subordonnés, et évolutif, tel que le rôle des agents peut évoluer et le degré de spécialisation augmenter particulièrement pour les agents du même niveau.

1 - Les interactions entre les agents du modèle de procédé logiciels :

- **Interactions (inter agents tâches) (inter agents fragments) :**

Ce sont des relations d'accointances ; chaque agent tâche connaît ses accointances dans son espace de travail, il peut communiquer, demander et offrir des services à ses voisins.

Les relations peuvent varier de la coopération en cas d'intérêt commun, en passant par la compétition dans le cas ou les buts sont incompatibles et de ressources suffisantes et aller jusqu'aux conflits en cas de ressources insuffisantes ou objectives contradictoires.

Tous ces cas de figures sont gérés en utilisant des algorithmes et stratégies de négociations les plus adéquats aux situations qui se présentent.

L'agent tâche n'a pas de représentation des agents tâches des autres espaces de travail ; ainsi il ne sait pas qu'ils existent et ne peut interagir avec eux qu'indirectement à travers les agents fragments.

Les interactions des agents fragment sont similaires à celles des agents tâches ; cependant la fréquence des échanges et des négociations est plus faible et moins complexe car la recherche des ressources, d'informations ou de services doivent être en priorité dans l'espace de travail local avant de passer à d'autres espaces de travail qui peuvent être éloignés géographiquement et de conditions de communications difficiles.

La négociation entre les agents fragments peut porter sur l'envoie d'un agent tâche dans un autre espace de travail à la recherche de ressources non disponible au niveau de son espace de travail cependant cette éventualité reste comme une perspective car il faut développer des mécanismes de localisation d'agents mobiles ce qui n'est pas abordé dans ce travail.

- **Interactions (agent fragment- agents tâches) (agent superviseur- agents fragments) :**

Les relations entre l'agent fragment et les agents tâche du même espace de travail sont des relations de subordination maître/esclave. L'agent fragment initialise les agents tâches leurs assigne les activités à effectuer ; il reçoit les produits et les rapports d'exécution la fin de l'exécution des tâches affectées, et peut être consulté pour trancher dans une négociation entre agents tâches.

L'agent tâche peut demander des services ou des ressources à l'agent fragment, ce dernier doit tout faire pour satisfaire la demande, même si la décision finale lui revient.

Les relations agent superviseur et agents fragments sont similaires à ceux de l'agent fragment et agents tâche avec cependant avec un flux d'échange et de communications moins importantes.

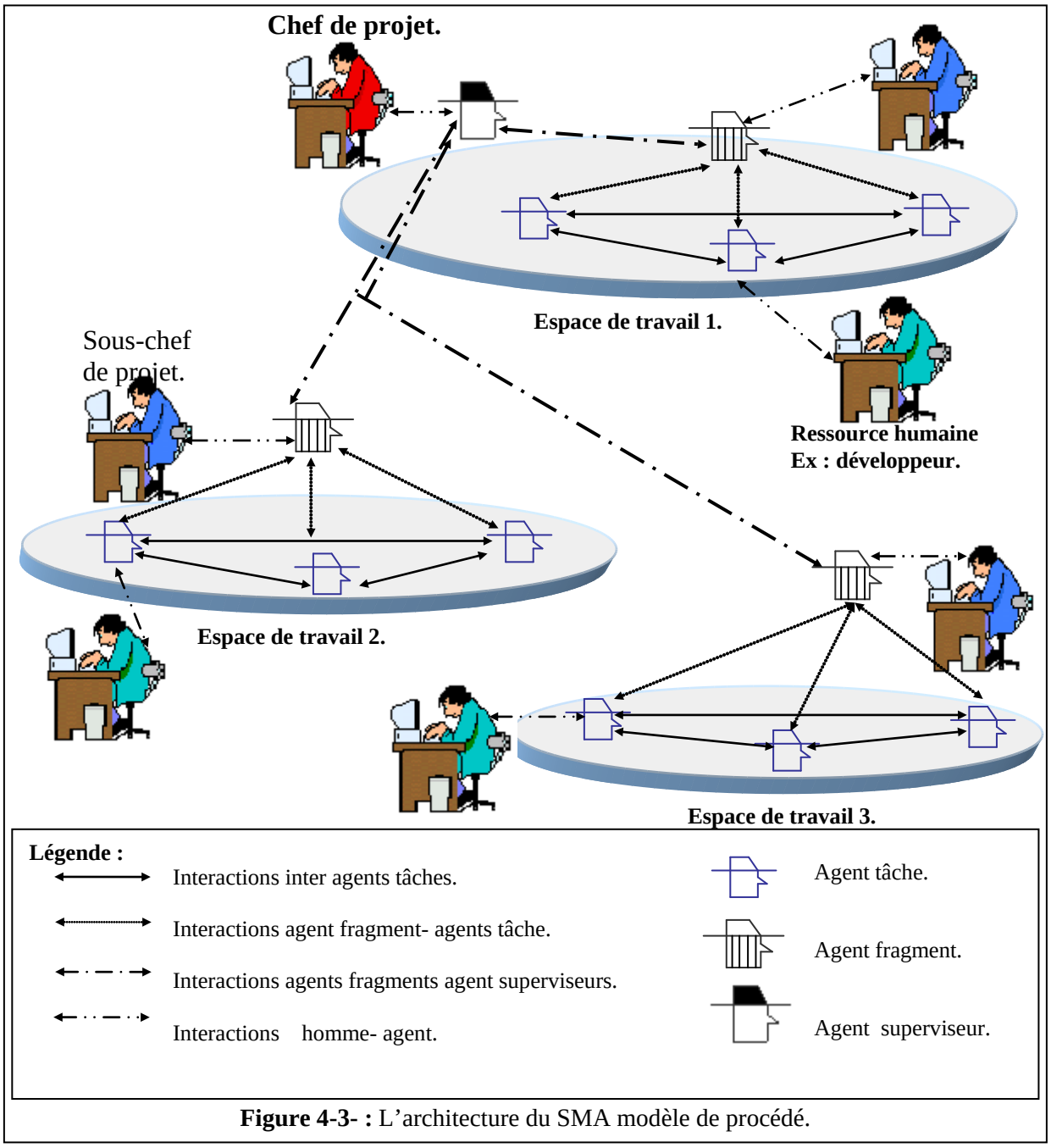


Figure 4-3- : L'architecture du SMA modèle de procédé.

V. Distribution du modèle de procédé logiciels :

Comme il a été déjà noté, la définition des agents s'est effectuée selon le modèle de procédé logiciel : l'agent superviseur pour modéliser le modèle de procédé, agent tâche pour exécuter les activités du modèle de procédé logiciels et Agent fragment est un intermédiaire entre la modélisation et l'exécution des tâches qui joue le rôle d'un coordinateur.

Les agents ont une certaine autonomie, car chaque agent est responsable d'une partie du modèle de procédé (sa modélisation ou son exécution) ; en plus, les agents procédé logiciel peuvent être distribués géographiquement ce qui permet la distribution du modèle de procédé ce qui facilitera le travail distribué.

Il est possible d'avoir un travail distribué tout en gardant le modèle de procédé « non distribué », stocké dans une base de données accessible par tous les agents ; ceci nous rappelle les systèmes à tableau noir : une base de connaissances accessible par les agents et un système de contrôle pour contrôler l'accès à ces connaissances – dans notre cas les connaissances sont le modèle de procédé logiciels-.

Cette solution n'est pas convenable à notre système : en plus des inconvénients des systèmes à tableau noir (la difficulté d'avoir un contrôle d'accès efficace), la distribution géographique nous oblige à avoir des protocoles de communication très efficace ; l'accès à la base de connaissance par des agents éloignés n'est pas toujours sûr ; la qualité du modèle de procédé et son exécution dépendra de la qualité des protocoles de communication utilisés.

La solution la plus adéquate est de distribuer le modèle de procédé logiciels, d'envoyer une seule fois le fragment de procédé à l'agent responsable de son exécution, et, à la fin recevoir le résultat final de son exécution; cette solution minimisera les interactions à longue distance et nous évitera ou du moins réduira les problèmes de communication (émission ou réception de message).

Les interactions dans un espace de travail sont plus fréquentes que celle entre les espaces de travail ; cela est dû au faite que chaque agent tâche connaît que les agents tâche de son espace de travail et par ceux des autres espaces de travail. La plus part des messages changés dans un espace de travail sont des messages de demande ou offre de services ou des messages de négociation pour l'obtention de ressources ; les autres messages peuvent concerner l'ajout ou la suppression d'un Agent tâche.

Autre avantage de la distribution du modèle de procédé est la possibilité d'effectuer une évaluation «locale »de l'exécution ; et d'introduire certaines modifications locales «en respectant des règles prédéfinis » ; elle nous permet d'avoir une certaine « autonomie locale » pour l'adaptation du modèle de procédé aux rythmes et aux traditions de travail de chaque espace de travail.

1- Technique de fragmentation du modèle de procédé :

Pour que la fragmentation du procédé logiciels soit la plus d'avantageuse possible, elle doit respecter certaines règles :

- La fragmentation doit permettre le plus de parallélismes dans l'exécution des tâches, ainsi tous les outils des espaces de travail serons exploités de manière optimale.
- Le choix de la technique de fragmentation doit minimiser les interactions inter-espace de travail et de respecter les spécificités locales ; Ainsi par exemple il est plus rationnel d'affecter le fragment « test des résultats » à l'espace de travail qui à des outils de tests.
- La taille des fragments ne doit être trop petite car cela influera négativement sur le flux des interactions inter-espaces de travail, ni trop grande car on perdra les avantages de la fragmentation et risquera de minimiser le parallélisme de l'exécution des tâches.

2- Fragmentation du procédé logiciels au niveau agent superviseur :

A ce niveau nous nous basons sur le taux d'interaction entre les activités du modèle de procédé : il faut qu'elle soit minimale entre les fragments.

Une des solutions les plus sensées est de fragmenter le modèle de procédé en respectant le cycle de vie du logiciel comme il est illustré dans la figure suivante :

Les interactions entre les phases du cycle de vie se limitent au passage des résultats d'une phase à la phase suivante ; par contre les interactions dans une phase sont plus denses ; Par exemple, dans la figure 4-4- les interactions entre les tâches « l'édition du texte », « compilation » et «exécution» sont plus fréquentes que celles entre les phases.

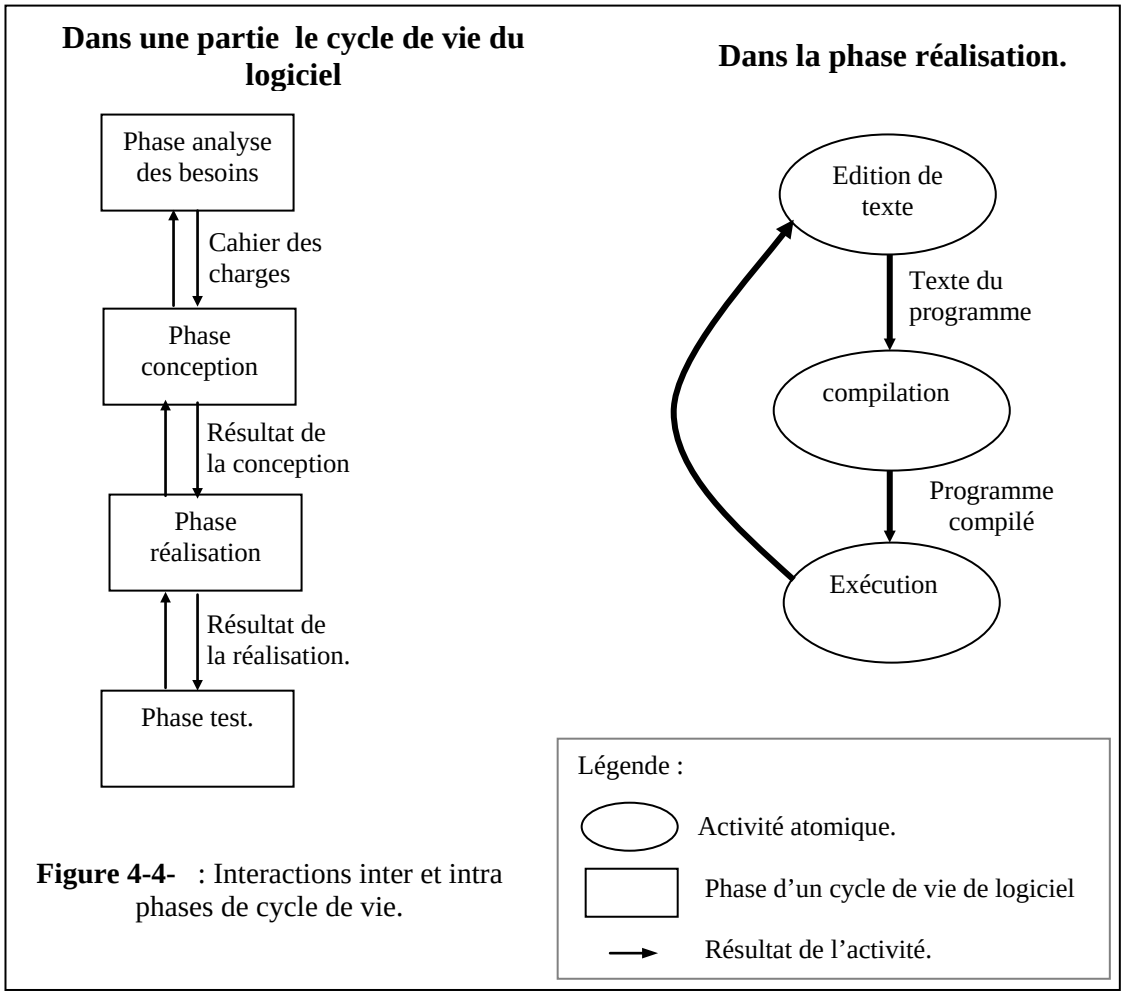


Figure 4-4- : Interactions inter et intra phases de cycle de vie.

Chaque « fragment phase » du cycle de vie du logiciel est affecté à un espace de travail.
Les interactions inter espaces de travail se limitent à des envois de résultats d'exécution finaux de l'espace de travail à la prochaine phase du cycle de vie.
Le parallélisme est plus explicite dans des cycles de vie ou des phases distinctes s'exécutent en parallèles tel que les cycles de vie en V ou en spirale.

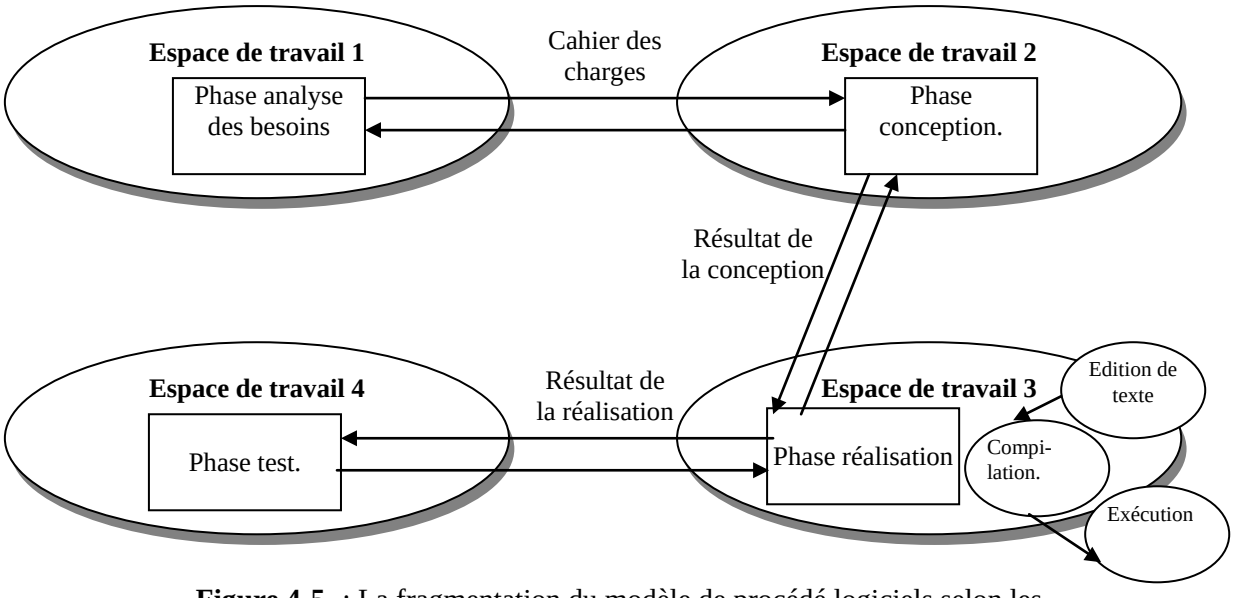


Figure 4-5- : La fragmentation du modèle de procédé logiciels selon les phases de cycle de vie

L'autre technique de fragmentation est de respecter les spécialités des espaces de travail et de fragmenter le modèle de procédé logiciels selon les ressources et outils existants dans des espaces de travail comme illustré dans la figure suivante :

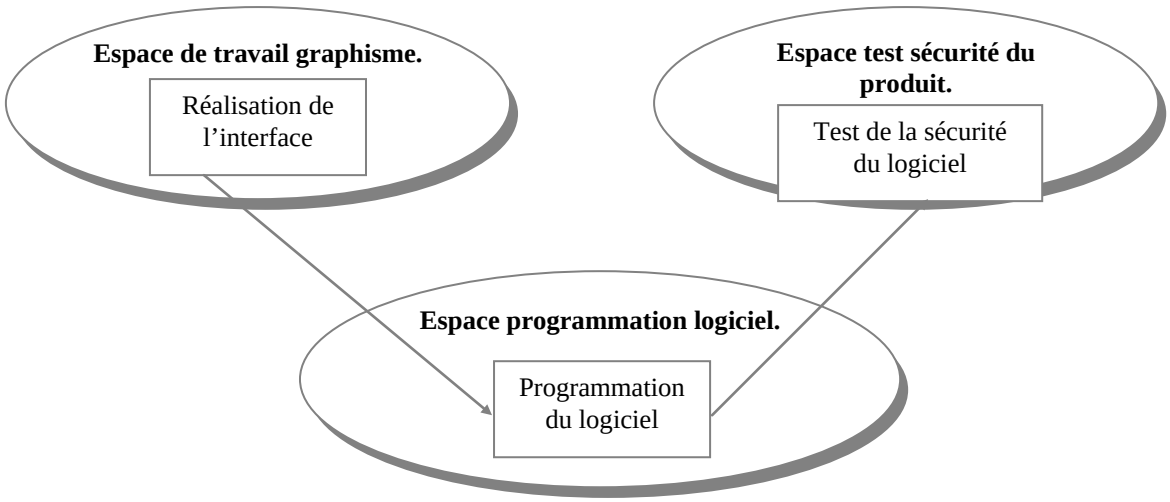


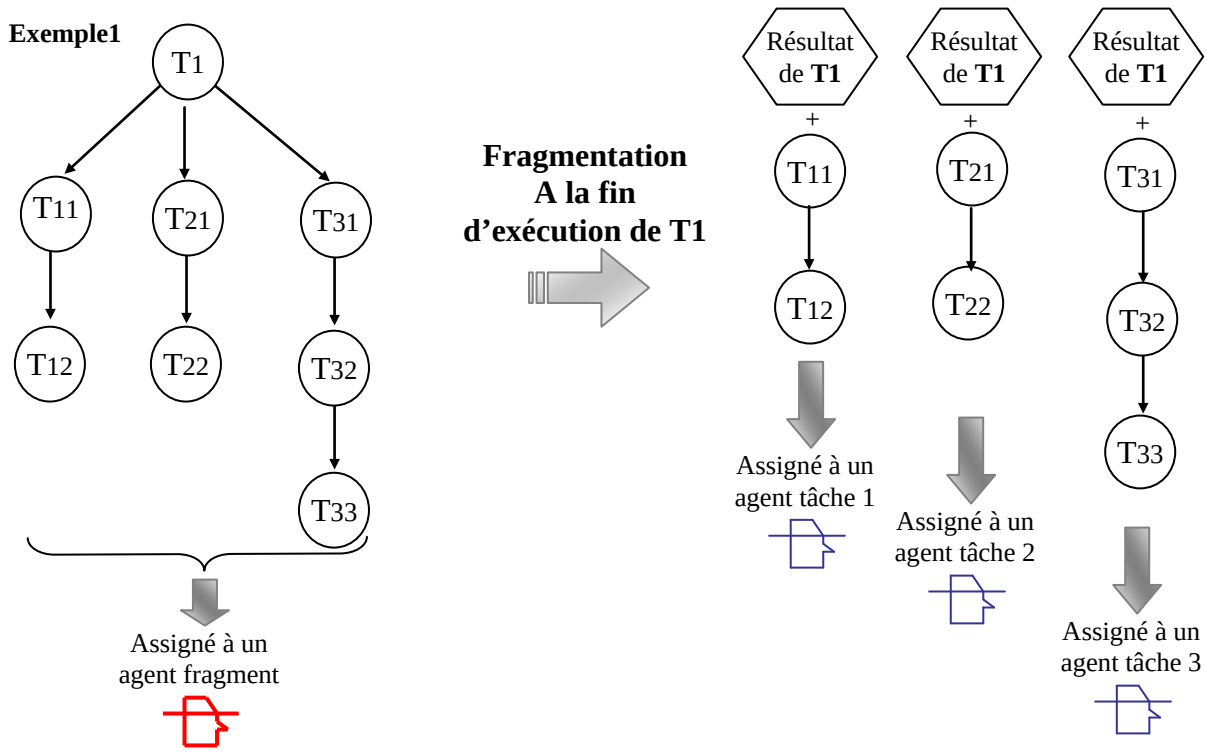
Figure 4-6- : Fragmentation du modèle de procédé selon les spécialités des espaces de travails.

3- Fragmentation du procédé logiciels au niveau agent fragment :

Le principe de fragmentation du modèle de procédé se base sur le principe de maximisation du parallélisme des tâches du modèle procédé logiciels.

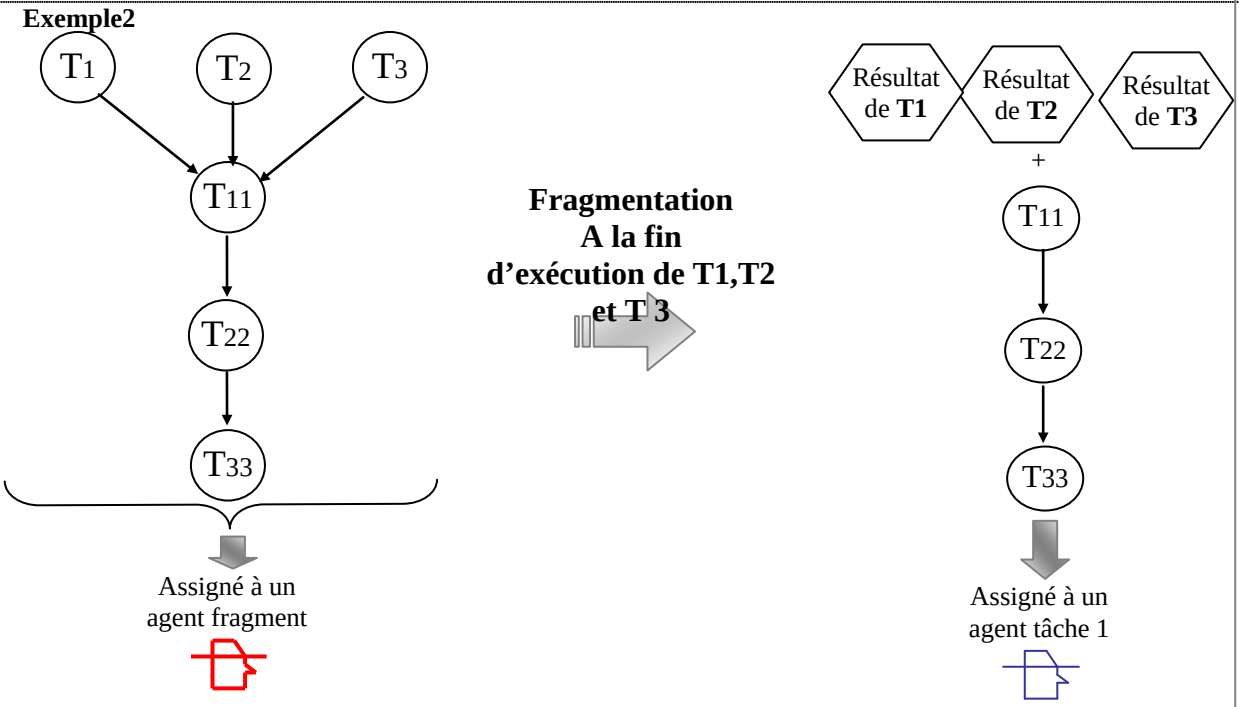
Ainsi tant qu'une série de tâches est séquentielle, elle constitue une série de tâches séquentielle assignée à un seul agent tâches.

Si la fin de la tâche T1 déclenche le début de « N tâches » qui s'exécute en parallèle, ce fragment est divisé en « N parties »,ou chaque partie de tâches séquentielles est affectée à un agent tâche -figure 4-7-.



- A la fin d'exécution de la tâche T1 (par un agent tâche1), le résultat d'exécution est envoyé à l'agent fragment ; Ce dernier consulte la suite de son fragment, établie les fragmentations nécessaires et assigne les serties de tâches séquentielles au agent tâches de son espace de travail.

- Agent tâche 1, Agent tâche 2, Agent tâche3 travaillent en parallèle. A la fin de l'exécution de leurs tâches ils envoient les résultats d'exécutions à leur agent fragment.



- A la fin d'exécution de la tâche T1, T2,T3 (par des agents tâches différents), les résultats d'exécution sont envoyés à l'agent fragment ; Ce dernier consulte la suite de son fragment, établie les fragmentations nécessaires et assigne les serties de tâches séquentielles au agent tâches de son espace de travail(dans ce cas il a qu'un agent tâche qui sera initialisé).

- Agent tâche 1 réalise les tâches assignées, a la fin de l'exécution il envoie les résultats d'exécutions à son agent fragment.

Figure 4-7- : Fragmentation du modèle de procédé logiciel au niveau de l'agent fragment.

4- Le facteur « humain » dans le modèle de procédé logiciels multi-agents :

Dans ce système l'interaction homme- agent est inévitable : l'opinion, la réflexion humaine restent irremplaçable, particulièrement dans la modélisation du procédé logiciels ou la « décision » et le suivie de la modélisation lui appartient.

Le facteur humain est exploité de deux manières :

- Soit c'est un responsable ou son rôle principale est de « suivre » la modélisation du procédé logiciel car la modélisation ne peut être effectuée par l'agent superviseur seul, et l'intervention humaine reste fondamental(cette tâche revient au chef de projets).
Le chef de projet peut aussi introduire des changements dans le modèle de procédé après son début d'exécution et donner des consignes de modélisation en interaction avec l'agent superviseur.
Le sous-chef de projet suit l'exécution des activités et peut intervenir dans des décisions concernant des négociations qui n'aboutissent pas.

-de l'autre coté l'humain peut être considéré comme «une ressource », son rôle est d'exécuter les ordres qu'on lui donne et de réaliser les tâches qu'on lui attribue, tel que la conception ou la programmation d'un module ou d'une interface. Dans ce cas ce sont ses qualités de créateurs ou de programmeurs qui sont sollicitées et pas ses capacités de gestionnaire de projet.

5- Facteur d'évaluation de l'exécution du modèle de procédé logiciels :

Une des caractéristiques principales de notre modèle de procédé (en dehors qu'il est distribué) est sa dynamicité ; le modèle de procédé est en constante observation et peut être modifié pendant son exécution dans le but de l'adapter à de nouvelles conditions d'exécution.

Le modèle de procédé instancié n'est pas forcément le même que celui qui est exécuté ; des paramètres imprévisibles ou des anomalies peuvent enter en jeu et modifier les résultats attendus.
Si par exemple la durée « prévue » d'une tâche T1 est de 'X'temps, et pour des raisons non prises en considération tel que l'absence d'un développeur ou la panne d'un outil, la tâche T1 est exécutée en '2X' temps, ce retard risque d'influer sur tout le modèle de procédé. Si cela se répète pour d'autres tâches, la fin du projet risque d'être retardée et les délais de réalisations non respectés.

Ces évènements imprévisibles ne peuvent pas être prise en compte lors de la modélisation, ils doivent être pris en considération ' détectés' lors de leurs apparitions c.a.d lors de l'exécution du modèle de procédé ; des décisions de modifications doivent être prises à temps afin d'éviter une grande déviation par rapports aux résultats attendus.

Une évaluation régulière et permanente doit être effectuée par les agents modèle de procédé ; la particularité de l'agent « d'être en activité continuelle » est exploitée pour déceler toute déviation du modèle de procédé logiciels en exécution.

L'évaluation de l'exécution des fragments modèle de procédé se base sur des facteurs d'évaluations, ces facteurs d'évaluation sont les mêmes que celles utilisés pour confirmer la réussite (ou pas)d'un projet, ces facteurs sont : le temps, le coût et la qualité de réalisation.

Chaque agent à ses propres facteurs d'évaluations, ainsi les facteurs d'évaluation de l'agent superviseur sont destinés à évaluer le modèle de procédé en entier, ils sont établis par le chef de projets et le client, et se sont celles qui détermineront la réussite du projet.

Les facteurs d'évaluation de l'agent fragment sont destinés à évaluer le fragment de procédé logiciel assigné, ils sont établis par l'agent hiérarchique (l'agent superviseur) lors de la

fragmentation du modèle de procédé en collaboration avec le chef de projets, puis transmis à l'agent fragment.
Les facteurs d'évaluation de l'agent tâche sont établis par l'agent fragment puis envoyé avec la série de tâche concernée par ces facteurs d'évaluations à l'agent tâche.

Exemple de facteurs d'évaluations :

- Le modèle de procédé a un temps de réalisation '**TpsM**', un coût de réalisation **'CtM'**, et une qualité de réalisation **' QM'** ;
- Chaque fragment du modèle de procédé a un temps de réalisation '**TpsF**', un coût de réalisation **'CtF'**, et une qualité de réalisation **' QF'** ;
  Les conditions suivantes doivent toujours être vérifiées :

➢ $\mathbf{TpsM} > \sum_{I:=1}^{N} \mathbf{Tps\ F}$ (N est le nombre de fragment du modèle de procédé logiciels).
(Le temps de communication et de collaboration doit être pris en considération lors de la détermination des Tps F).

➢ $\mathbf{CtM} > \sum_{I:=1}^{N} \mathbf{CtF}$

➢ $\mathbf{QM} > \mathrm{MAX} \left(\mathbf{QF} \right)_{I:=1}^{N}$

- Chaque série de tâches séquentielles du modèle de procédé a un temps de réalisation '**TpsS**', un coût de réalisation **'CtS'**, et une qualité de réalisation **' QS'** ;
  Les conditions suivantes doivent toujours être vérifiées :

➢ $\mathbf{Tps\ F\ i} > \sum_{I:=1}^{M} \mathbf{TpsS}$ (M est le nombre de série de tâches séquentielles du fragment 'i' du modèle de procédé logiciels).
(Le temps de communication et de collaboration doit être pris en considération lors de la détermination des Tps F).

➢ $\mathbf{CtF\ i} > \sum_{I:=1}^{M} \mathbf{CtS}$

➢ $\mathbf{QF\ i} > \mathrm{MAX}_{I:=1}^{M}\ \mathbf{QS}$

Chaque paramètre d'évaluation à une échelle de –2 à +2 pour évaluer l'exécution ;

➢ Ainsi pour le facteur temps nous avons :
-2 : très en retard ;
-1 : retard acceptable ;
0 :dans les temps ;
+1 : Avance ;
+2 : Bonne avance ;
A la fin d'exécution d'une série de tâche ou d'un fragment une valeur de –2 à +2 est donné au résultat, puis selon cette valeur des dispositifs peuvent être pris par l'agent concerné.

➢ pour le facteur coût nous avons :
-2 : très coûteux ;
-1 : coût acceptable ;
0 : Dans les normes ;
+1 : Economie ;
+2 : Bonne économie ;

A la fin d'exécution d'une série de tâche ou d'un fragment une valeur de –2 à +2 est donné au résultat, puis selon cette valeur des dispositifs peuvent être pris par l'agent concerné.

- pour le facteur qualité :

-1 : mauvaise qualité;
0: qualité acceptable;
+1 : Bonne qualité;

A la fin d'exécution d'une série de tâche ou d'un fragment une valeur de –1 à +1 est donnée au résultat, puis selon cette valeur des dispositifs peuvent être pris par l'agent concerné.
La qualité peut être évaluée en prenant en considération le langage utilisé sa compatibilité avec les plates-formes et le langage demandé, la qualité le taux de réalisation.

I- Architecture interne des agents modèle de procédé logiciels :

Après étude de différentes architectures internes connues de l'agent, nous avons opté pour une architecture modulaire qui est très répandue et a fait ses preuves dans le cadre tant de travaux théoriques que de travaux pratiques.

L'architecture modulaire est conçue comme un assemblage de modules où chacun de ceux-ci réalise une fonction horizontale particulière. Cette architecture convient parfaitement à notre agent modèle de procédé où sont exécutées plusieurs fonctions horizontales en parallèle ; elle nous permet aussi de prédéfinire les échanges des informations entre ces modules et nous donne ainsi un moyen de définir le comportement de l'agent face aux évènements qui peuvent se produire.

L'architecture à base de règles ne convient pas à notre agent modèles de procédé ; en effet, la nature de notre modèle de procédé logiciels -dynamique et évolutive- ne nous permet pas d'avoir en permanence une base de règles pertinente dont la mise à jour régulière et indispensable peut nuire à la qualité de fonctionnement notre agent ; par ailleurs, une architecture interne identique à tous les agents du modèle de procédé présente le grand avantage de ne pas avoir à programmer individuellement tous les types d'agents.

L'architecture les agents est modulaire horizontale ; chaque agent est composé de 6 modules qui travaillent en parallèle et en collaboration. Chaque module est responsable de certaines tâches 'internes' de l'agent. La représentation de l'architecture de l'agent modèle de procédé logiciels peut être comme suit :

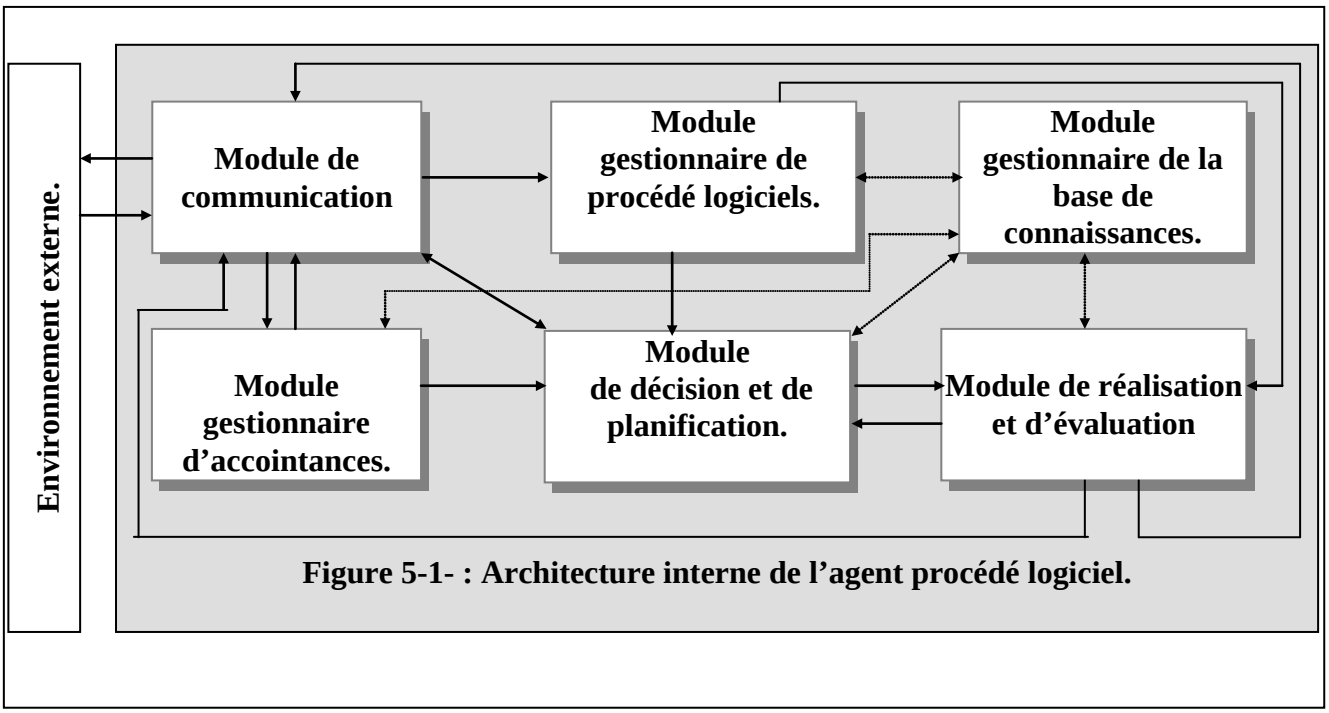


Figure 5-1- : Architecture interne de l'agent procédé logiciel.

L'agent modèle de procédé est composé de deux types de modules :

Les modules gestionnaires de l'état interne de l'agent : le rôle de ces modules est la gestion du fonctionnement interne de l'agent. Ce sont les modules: « communication », « gestionnaire de la base de connaissance », et « gestionnaire d'accointances » . Les fonctions de ces modules sont similaires pour les trois types d'agent (superviseur, fragment et tâche).

Les autres modules sont des modules de modélisation et d'exécution du modèle de procédé ; leur rôle principal est la gestion du procédé logiciel aux niveaux de la modélisation, de la planification, de l'exécution, et de l'introduction des changements. Ces modules sont : le module gestionnaire de procédé logiciels, le module décision et planification et le module exécution et évaluation de l'exécution.

Les fonctions de ces modules varient d'un type d'agent à un autre ; par exemple, la fonction principale du « module réalisation et évaluation de la réalisation » de l'agent tâche est d'exécuter les tâches assignées, par contre la fonction principale de ce module (dans l'agent fragment) est d'évaluer les résultats d'exécutions et de décider des changements à effectuer au niveau du modèle de procédé.

Les agents modèle de procédé ont donc *inévitablement* la même architecture interne car ils ont le même *comportement* ; chaque type est responsable d'un fragment de procédé logiciel et a la responsabilité de gérer sa représentation et l'évolution de son exécution (à différent niveau bien sûr).

1- Module communication :

Ce module s'occupe essentiellement de la communication avec l'environnement externe. L'environnement externe peut être composé d'agents logiciels ou d'agents humains (ressources ou chefs de projets).

Les fonctions essentielles de ce module sont l'émission et la réception des messages ainsi que leur interprétation.

L'interprétation des messages dépend des protocoles utilisés et de règles d'interprétation prédéfinis.

Lors de la réception d'un message, le module défini le type de message, identifie le module destinataire concerné et le lui transmet.

Le parallélisme des modules se matérialise et apparaît lors de la réception des messages ; le module communication peut recevoir plusieurs messages simultanément destinés aux différents modules de l'agent, ces derniers peuvent recevoir leurs messages internes et travailler en parallèle sans difficulté.

Les messages reçus ou envoyer par le module communication peuvent être classés comme suit :

Message.	Contenue.
Consignes de modélisation.	Tâches, objectifs, ressources, techniques, facteurs d'évaluations… tous ce qui est nécessaire pour la modélisation du modèle de procédé logiciels.
Fragment assigné.	Le fragment du modèle de procédé logiciel assigné par un agent «superviseur » ou «fragment » à son subordonné.
Messages d'interactions.	Ils contiennent des messages de négociation ou de demande de services, tous ce qui concerne une coopération entre agents.
Changements de valeurs d'évaluations.	Ces messages sont toujours envoyés d'un agent supérieur à son subordonné. Les changements sont toujours pour élargir ces valeurs d'évaluations.
Rapport d'exécution.	Rapport qui contient les résultats d'exécutions
Modification fragment de procédés.	Ce message est envoyé par l'agent « tâche » ou « fragment » à son supérieur. C'est un ordre de modification qui concerne la partie du modèle de procédé prise en charge par la source du message.
Demande de modifications	Ce message est envoyé par l'agent « superviseur » ou agent « fragment » à son subordonné. Cette demande peut être acceptée ou refusée après négociation des parties concernées.
Rapport de modifications effectuées.	Contient les modifications effectuées sur le modèle de procédé logiciels dont il est responsable.

2- Module gestionnaire de la base de connaissances :

Comme son nom l'indique, ce module gère la base de connaissances de l'agent, en termes d'accès et de cohérence des informations stockées ; sachant que cette base est accessible à la plupart des modules, il est en effet nécessaire d'établir des règles et des priorités d'accès. Ces priorités peuvent être résumées comme suit :

Module.	Type d'accès.	Priorité.
Décision et planification.	L	**2** ou **3** selon la priorité du gestionnaire d'accointances.
Gestionnaire d'accointances et de ressources.	L/E	**- 3** si demande d'allocation de ressources. **- 2** sinon.
Gestionnaire du modèle de procédé logiciel.	L/E	**1**
Réalisation et évaluation de la réalisation.	L	**4**
Communication.	**Pas d'accès.**	

La base de connaissances de l'agent contient les informations suivantes :

- Le modèle de procédé logiciels actuels et toutes les versions historique ou logique utilisées par cet agent.
- Les modèles d'interactions ;
- Les stratégies de négociations et de collaboration suivie par cet agent, ainsi que leur historique d'utilisation.

- Les paramètres d'évaluation d'exécution actuels, et les versions précédantes.
- Les rapports et résultats d'exécutions ou d'assignations.
- Les rapports de changements.
- Le modèle d'accointances et le modèle du moi de l'agent, ainsi que l'état d'allocation actuel des ressources.

3- Module gestionnaire d'accointances et des ressources :

Ce module veille à la mise à jour des états internes de l'agent et de ses voisins.

L'état interne de l'agent illustre l'état de ses ressources (alloué, libres, en panne, …), de ses objectifs actuels ainsi que ses contrats avec ses voisins.

L'état des voisins est caractérisé par les capacités de ceux-ci, l'état de leurs ressources ainsi que leurs objectifs actuels.

Le module gestionnaire d'accointances et des ressources est le premier informé par le module communication d'une libération de ressources.

Il était possible de laisser ces activités au module « gestionnaire de base de connaissances », néanmoins nous avons préféré avoir un module pour gérer l'état interne de l'agent par soucie d'efficacité et pour avoir l'accès direct à l'état des accointances et des ressources.

Quoique fonctionnant tous trois de manière identique et effectuant des tâches simples, les modules « communication », « gestionnaire de base de connaissances » et « gestionnaire d'accointances », sont indispensables à chacun des agents définis dont ils contrôlent le fonctionnement interne.

4- Module décision et planification :

Ce module est considéré comme le cerveau de l'agent, c'est cette partie qui s'occupe de la «réflexion » sur les décisions à prendre concernant les négociations en cours, les tâches à exécuter et de leur ordre d'exécution.

Tous les autres modules sont à son service, ils doivent lui fournir toutes les informations «pertinentes » nécessaire pour prendre les bonnes décisions.

Ces décisions portent entre autres sur les allocations de ressource et l'ordre d'exécution des tâches.

5- Module réalisation et évaluation de la réalisation :

L'objectif de ce module est d'exécuter les tâches planifiées par le module précédent; ces tâches peuvent, par exemple, être les activités du modèle de procédé logiciels, si l'agent est un agent tâches, ou une initialisation d'agents subordonnés, s'il s'agit d'un agent fragment ou d'un agent superviseur.

L'autre mission de ce module est d'évaluer l'exécution :

Si l'agent est un agent tâche, l'évaluation portera sur son propre travail, suivant ses propres valeurs d'évaluations ; chaque partie exécutée est ainsi systématiquement évaluée. La décision d'effectuer des changements ne concerne que ses propres tâches qui ne sont pas encore exécutés. L'agent tâche ne doit prendre en considération que ses ressources personnelles disponibles.

Les conditions d'implémentation de changements sur le modèle de procédé sont très strictes à ce niveau ; ainsi aucun autre agent ne peut contester l'application d'une décision de changement du modèle de procédé prise par cet agent qui envoie un rapport de changement à l'agent supérieur. Ainsi l'autonomie locale est vérifiée et la prise de décisions locales est possible.

Si l'agent est un agent fragment, l'évaluation sera portée sur les rapports d'exécution reçue par ses agents tâches ; son travail consiste à les assembler et comparer leurs résultat avec ce qui était prévu. Les déviations constatées sont alors évaluées avec des paramètres interne à l'agent fragment et des décisions de changements pourraient être prises en conséquences ; ces décisions concernent la partie du modèle de procédé assignée et qui n'est pas encore exécutée, en ne prenant en considération que les ressources disponibles des ses agents tâches .

Ces modifications sont envoyées aux agents tâche concernées et peuvent être refusées pour des raisons telles que le fait que cette partie a été exécuté. Si les agents tâche acceptent les changements le rapport des changements est envoyé à l'agent superviseur.

En ce qui concerne l'agent superviseur l'évaluation s'effectue de la même manière que pour l'agent fragment ; cependant la prise de décision de changement doit prendre en considération tous les agents fragment ainsi que l'avis personnel du chef de projet.

Le chef de projet peut introduire des changements ; cela permet de répondre aux demandes «changeantes et évolutives» des clients. Ce type de changement n'est pas très négocié particulièrement si les valeurs d'évaluation sont revues et allégées.

Un des autres avantages de ce module est la possibilité d'introduire le modèle de procédé partie par partie.

6- Module gestionnaire du procédé logiciels.

Le rôle de ce module est de formaliser le modèle de procédé de la meilleure manière possible à partir des informations reçue par le chef de projets. Il doit permettre d'introduire les changements décidés par le module « réalisation et évaluation de la réalisation » et vérifier la consistance de ces modifications par rapports au fragment ou au modèle de procédé global de l'agent. Ce module prend également en charge le suivi de l'exécution du modèle de procédé en fixant la partie à exécuter avant de la faire passer à l'agent « décision et planification » ; ainsi, si une partie est sélectionnée, elle doit être exécutée même si elle est concernée par un rapport de changement.

Le module de l'agent superviseur est ainsi celui qui a le plus de responsabilité car il a pour tâche la modélisation globale du procédé. Les décisions de changements du procédé logiciels sont rares à son niveau, elles se résument à l'introduction de nouvelles conditions ou des facteurs d'évaluations demandées par le client ou le chef de projet.

II- Architecture interne de l'agent superviseur :

La figure 5-2- montre les activités internes de l'agent superviseur, ainsi nous pouvons constater que les fonctions principales du module « réalisation et évaluation de la réalisation » est l'initialisation des agents fragments, assignation des fragments de procédé logiciel et l'évaluation des résultats reçus. Le module « gestionnaire du modèle de procédé logiciels » est le module le plus surchargé ; il a devoir de modéliser tout le procédé logiciels et particulièrement d'introduire tous les changements décidés par les agents fragments en conservant la cohérence du modèle de procédé logiciels.

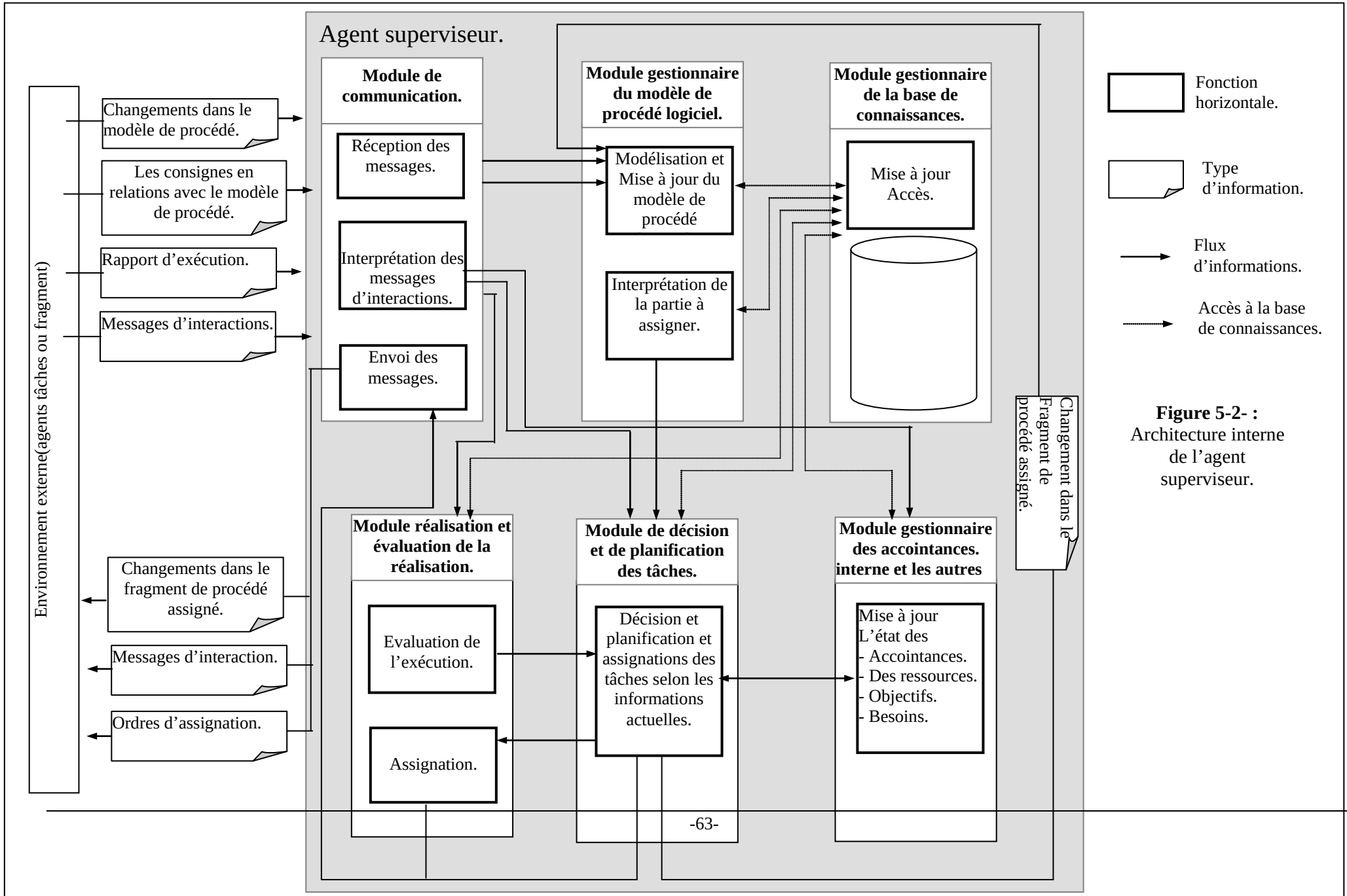


Figure 5-2- : Architecture interne de l'agent superviseur.

1- Algorithme de fonctionnement de l'agent superviseur:

DEBUT

- **Module communication :**

Début :

➢ **Si** *message_reçu* **= «**consigne de modélisation de procédé logiciels » et
source = « chef de Project ».
Alors 1- envoi *message_interne* **= «** Consignes de modélisation de procédé logiciels »au **Module gestionnaire du modèle de procédé logiciel.**
2- Aller à : Etiq0.
Fin de si.

➢ **Si** *message_reçu* **= «** Changement dans le procédé logiciels »
source = « Agent fragment ».
Alors 1- envoi *message_interne* **=«** Changement dans le procédé logiciels » au **Module gestionnaire du modèle de procédé logiciel**.
2- Aller à : Etiq7.
Fin de si ;

➢ **Si** *message_reçu* **= «**Message d'interaction »
Alors :
Si *source* = « Chef de Project »
Alors :
Si *message d'interaction* = «changement du modèle de procédé ou des facteurs d'évaluations »
Alors 1- envoi *message_interne* **=«**Changement dans le procédé logiciels » au **Module gestionnaire du modèle de procédé logiciel.**
2- Aller à : Etiq10.
Sinon;

Si *message d'interaction* = «Ajout de ressources »

Alors 1- *envoi _message _interne* = «Ajout de ressources » au **Module gestionnaire des accointances et de ressources (interne et des autres).**
2- Aller à : Etiq11.
Fin de si ;

Fin de si ;
Sinon

Si *message d'interaction* = «libération de ressources »

Alors 1- *Envoi_message_interne* = « libération de ressources »» au **Module gestionnaire des accointances et de ressources (interne et des autres).**
2- Aller à : Etiq11.
sinon;

3 **2** **1**

3 2 1 **Si** *message d'interaction* = «Demande ou offre de services »

Alors 1- envoi *message_interne* = «Demande, offre, négociation de services » au **Module de décision et de planification des tâches :**
2- Aller à : Etiq14.
Fin de si ;

Fin de si ;

Fin de si ;

- **Si** *message_reçu* **= «**Rapport d'exécution » et *source = « Agent fragment»*
  Alors 1- *envoi_message_interne* = «Rapport d'exécution » au **Module gestionnaire de la base de connaissances.**
  2- Aller à : Etiq19.
  Fin de si ;

- **Si** *message_reçu* **= «**Demande de consultation de résultats d'exécutions » et *source = «Chef de projects»*
  Alors 1- *envoi_message_interne* = «Demande de consultation de résultats d'exécutions » au **Module gestionnaire de la base de connaissances.**
  2- Aller à : Etiq24.
  Fin de si ;

- **Etiq20 : 1-** *envoi_message_interne* = «Rapport d'exécution » au **Module réalisation et évaluation de la réalisation.**
  2- Aller à : Etiq21.

- **Etiq6 :** *Envoi_message* = «Assignation des fragments et facteurs d'évaluations».
  Destination = «Tous les agents fragments ».

- **Etiq9 :** *Envoi_message* = «Changement effectué ».
  Destination = «Agent fragment qui a effectué les changements ».

- **Etiq13 :** *Envoi_message* = «MAJ d'état de ressources effectuée ».
  Destination = «Chef de projets ou agents qui a ajouter ou libérer des ressources ».

- **Etiq15 :** *Envoi_message* = «Réponse message d'interaction ».
  Destination = «source du message d'interaction ».

- **Etiq22 :** *Envoi_message* = «Proposition de changement dans le modèle de procédé logiciel ».
  Destination = «Chef de projets ».

- **Etiq25 :** *Envoi_message* = «Résultat final ou partiel de l'exécution du modèle de procédé logiciel ».
  Destination = «Chef de projets ».

Fin ;

- **Module gestionnaire du modèle de procédé logiciel**.

Début :

- **Etiq0 :1-** Modéliser le modèle de procédé logiciels en interaction avec le chef de projet jusqu'à modélisation finale.
  2- Etablir les facteurs d'évaluation du modèle de procédé modélisé.
  3- Envoi *message_interne* = «modèle de procédé logiciel global » au **module gestionnaire de la base de connaissances.**
  4- Aller à : étiq1.

- **Etiq3 : 1-** Sélectionner la partie du modèle de procédé à exécuter.
  2- Envoi *message_interne* = « partie du modèle à exécuter » au **Module de décision et de planification des tâches.**
  3- Aller à : Etiq4.

- **Etiq7 : 1-** Introduire les changements dans le modèle de procédé global.
  2- Envoi *message_interne* = «changement du modèle de procédé effectué » au **Module gestionnaire de la base de connaissances.**
  3- Aller à : Etiq8.

- **Etiq10 : 1-** Introduire les changements dans le modèle de procédé global en interaction avec le chef de projets.
  2- *Envoi_message_interne* = «changement du modèle de procédé effectué » au **Module gestionnaire de la base de connaissances.**
  3- Aller à : Etiq8.

- **Etiq23 :<u>Si</u> fin d'exécution de tous le modèle de procédé**

  <u>Alors</u> : 1- *Envoi_message_interne* = «Envoi du résultat final de l'exécution»au **module gestionnaire de la base de connaissances**
  2- Allez à Etiq24 :
  <u>sinon</u>
  Allez à Etiq3 :
  <u>Fin de si ;</u>

<u>Fin ;</u>

- **Module gestionnaire de la base de connaissances.**

Début :

- **Etiq1 : 1-** Sauvegarde du modèle de procédé global et des facteurs d'évaluations de l'agent superviseur.
  2- Envoi *message_interne* = « modèle de procédé prêt à s'exécuter » au **Module gestionnaire du modèle de procédé logiciel.**
  3- Aller à Etiq3 :

- **Etiq8 : 1-** Sauvegarde des changements du modèle de procédé global et des facteurs d'évaluations de l'agent superviseur.
  2- Envoi *message_interne* = «Changement effectué » au **Module communication.**
  3- Allez à Etiq9 :

1

- **Etiq12 : 1-** Sauvegarde des mise à jour des états des ressources.

1

Si Maj à partir de message externe

Alors **1-** Envoi *message_interne* = «MAJ effectuée » au **Module communication ;**
2- Allez à Etiq13 : ;
sinon
1- *Envoi_message_interne* = «MAJ effectuée » au **Module de décision et de planification des tâches ;**
2- Allez à Etiq26 : ;
Fin de si ;

- **Etiq16 : 1-**Sauvegarde des Assignations des fragments aux agents fragments
  2- *Envoi_message_interne* = «assignation des fragments sauvegardées » au **Module réalisation et évaluation de la réalisation.**
  3- Allez à Etiq17 :

- **Etiq19 : 1-**Sauvegarde du Rapport d'exécution.
  2- *Envoi_message_interne* = «Rapport d'exécution sauvegardé » au **Module communication.**
  3- Allez à Etiq20 :

- **Etiq24 : 1-** *Envoi_message_interne* = «Résultat final ou partiel d'exécution » au **Module communication.**
  2- Allez à Etiq25 :

Fin ;

- **Module de décision et de planification des tâches :**

Début :

- **Etiq4 : 1-** Fragmenter le modèle de procédé logiciels
  2- Etablir les facteurs d'évaluation de chaque fragment.
  3-envoi *message_interne* modèle de procédé prêt à s'exécuter au **Module réalisation et évaluation de la réalisation.**
  4- Aller à Etiq5.

- **Etiq14 : 1-** Consulter la stratégie utilisée.
  2- Vérifier l'état des ressources.
  3- Décision selon les objectifs à suivre et les priorités.
  Si changement dans l'état des ressources ou des accointances

  Alors **4-** envoi *message_interne* = «Changement de l'état des ressources ou des accointances » au **Module gestionnaire des accointances et de ressources (interne et les autres) ;**
  5- Allez à Etiq11 :
  Fin de si ;

- **Etiq26 : 1-** *Envoi_message_interne* = «Réponse message d'interaction » au **Module communication.**
  2- Allez à Etiq15 :

Fin ;

- **Module réalisation et évaluation de la réalisation.**

Début :

- **Etiq5 : 1-** Initialiser les agents fragment nécessaires**.**
  2- assigner les fragments et leurs facteurs d'évaluations aux agents fragment correspondant .
  3- *Envoie message interne* = « Assignation des fragments aux agents fragments» au **Module gestionnaire de la base de connaissances.**
  4- Allez à Etiq16 :

- **Etiq17 :1-** *Envoi_ message_interne* = «Assignation des fragments et facteurs d'évaluations» au **Module communication.**
  2- Allez à Etiq6 :

- **Etiq 1 : 1-** *Envoi_message_ interne* = «Assignation des fragments et facteurs d'évaluations» au **Module gestionnaire du modèle de procédé logiciel**.
  2-Allez à Etiq3.

- **Etiq 21 : 1-** Evaluation des rapports d'exécutions.
  Si déviation du modèle instancié
  Alors
  1- Consultation de l'état des ressources,
  2- Consultation des facteurs d'évaluations,
  3- Consultation des priorités d'exécution.
  4- Proposition de changements dans le modèle de procédé logiciels.
  5- *Envoi_message_interne* = «proposition de changement du modèle de procédé logiciels» au **Module communication.**
  3- Allez à Etiq22 :
  Sinon
  Si Dernier Rapport d'exécution de la partie assignée
  Alors
  1- *Envoi_message_interne* = «fin d'exécution de la partie assignée» au **Module gestionnaire du procédé logiciel.**
  2- Allez à Etiq23 :
  Fin de si ;
  Fin de si ;

Fin ;

- **Module gestionnaire des accointances et de ressources (interne et les autres).**

Début :

- **Etiq11 : 1-** MAJ de l'état des ressources.
  2- *Envoie message interne* = «Etat des ressources ou des accointances mis à jour» au **Module gestionnaire de la base de connaissances.**
  3- Allez à Etiq12 :

Fin ;

FIN.

II- Architecture interne de l'agent fragment :

L'agent fragment est un intermédiaire entre la modélisation du procédé logiciel au niveau de l'agent superviseur et l'exécution du modèle de procédé au niveau de l'agent tâches. Son rôle est de revoir le fragment de procédé et de le distribuer sur des agents tâches pour les exécuter. La figure suivante illustre les principales fonctions de l'agent fragment.

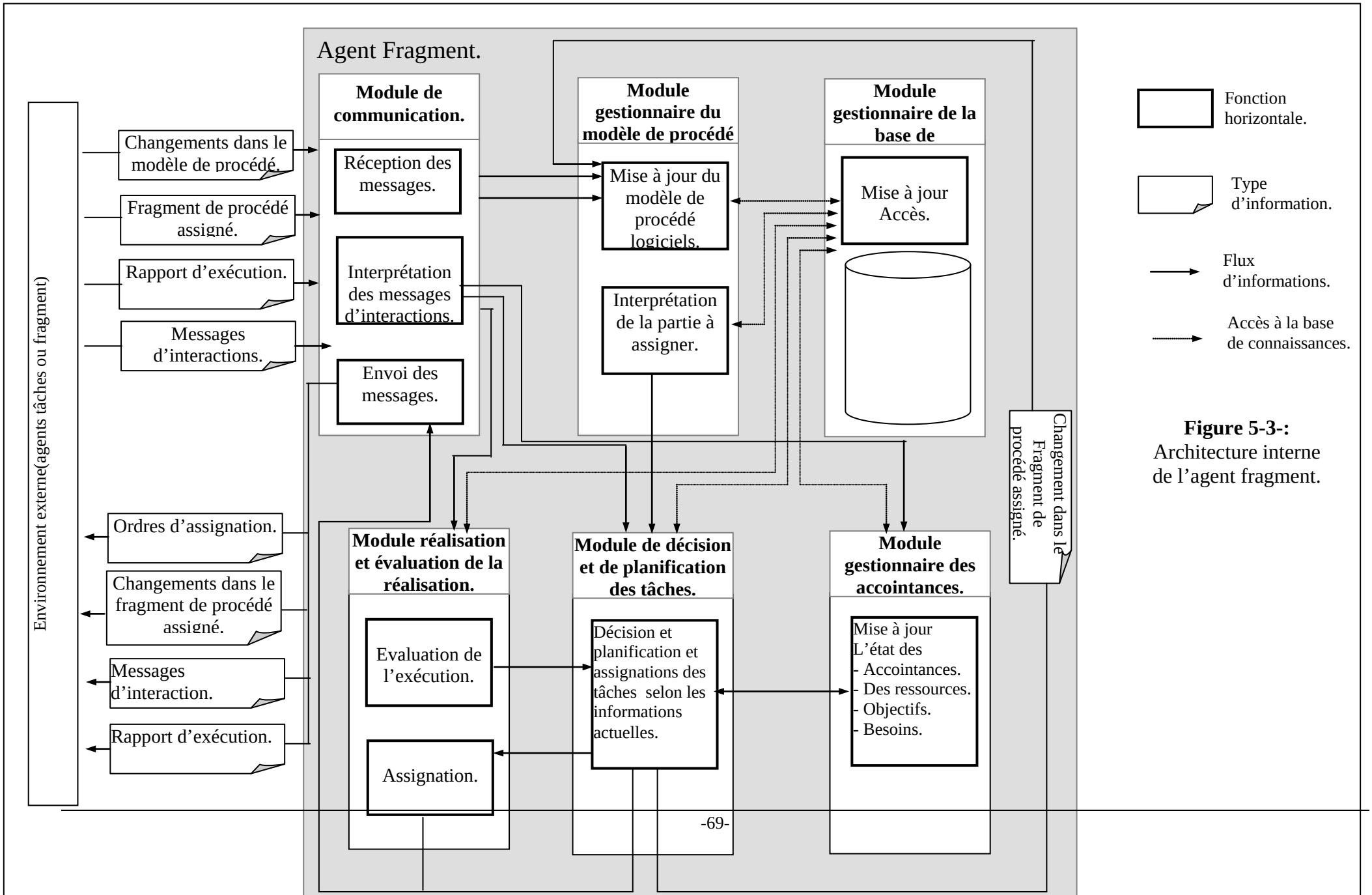


Figure 5-3-:
Architecture interne de l'agent fragment.

1-Algorithmes de fonctionnement de l'agent fragment :

DEBUT :

- **Module communication :**

Début :

- **Si** *message_reçu* **= «**Assignation des fragments et facteurs d'évaluations» et *source = « Agent superviseur».*
  Alors 1- Envoi *message_interne* **= «** fragment de procédé et facteurs d'évaluations »au **Module gestionnaire du modèle de procédé logiciel**.
  2- Aller à : Etiq0.
  Fin de si.

- **Si** *message_reçu* **= «** demande de changement dans le fragment procédé logiciels » *source = « Agent superviseur ».*
  Alors 1- *envoi_message_interne* **=«** demande de changement dans le fragment de procédé logiciels » au **Module gestionnaire du modèle de procédé logiciels**.
  2- Aller à : Etiq9.
  Fin de si ;

- **Si** *message_reçu* **= «** Changement dans le procédé logiciels » *source = « Agent tâche ».*
  Alors 1- envoi *message_interne* **=«** Changement dans le procédé logiciels » au **Module gestionnaire du modèle de procédé logiciel**.
  2- Aller à : Etiq13.
  Fin de si ;

- **Si** *message_reçu* **= «**Message d'interaction »
  Alors :
  Si *source* = «Agent superviseur »
  Alors :
  Si *message d'interaction* = «demande de changement du modèle de procédé ou des facteurs d'évaluations »
  Alors 1- *envoi_message_interne* **=«**Changement dans le procédé logiciels » au **Module gestionnaire du modèle de procédé logiciel**.
  2- Aller à : Etiq9.
  Sinon;

  Si *message d'interaction* = «Ajout ou libération de ressources »

  Alors 1- envoi *message_interne* = «Ajout de ressources » au **Module gestionnaire des accointances et de ressources (interne et des autres).**
  2- Aller à : Etiq13.
  Fin de si ;

  Fin de si ;
  Sinon

1 2 3

3 2 1

Si *message d'interaction* = «libération de ressources »

Alors 1- *Envoi_message_interne* = « libération de ressources »» au **Module gestionnaire des accointances et de ressources (interne et des autres).**
2- Aller à : Etiq13.

sinon;

Si *message d'interaction* = «Demande ou offre de services ou réponse négociation».

Alors 1- envoi *message_interne* = «Demande, offre, négociation de services » au **Module de décision et de planification des tâches :**
2- Aller à : Etiq16.

Fin de si ;

Fin de si ;

Fin de si ;

- **Si** *message_reçu* **=** «Rapport d'exécution » et *source = « Agent tâche»*
  Alors 1- *envoi_message_interne* = «Rapport d'exécution » au **Module gestionnaire de la base de connaissances.**
  2- Aller à : Etiq19.
  Fin de si ;

- **Si** *message_reçu* **=** «Demande de consultation de résultats d'exécutions » et *source = «Chef de projects»*
  Alors 1- *envoi_message_interne* = «Demande de consultation de résultats d'exécutions » au **Module gestionnaire de la base de connaissances.**
  2- Aller à : Etiq24.
  Fin de si ;

- **Etiq8 :** *Envoi_message* = «Assignation des série de tâches séquentielles et facteurs d'évaluations».
  Destination = «Tous les agents tâche de l'espace de travail de l'agent fragment ».

- **Etiq10 :** *Envoi_message* = «Changement refusés + cause du refus ».
  Destination = «Agent superviseur».

- **Etiq12 :** *Envoi_message* = «Changements effectués».
  Destination = «Agent qui a demander les changements(superviseur ou tâche) ».

- **Etiq15 :** *Envoi_message* = «MAJ d'état de ressources effectuée ».
  Destination = «Chef de projets ou agents qui a ajouter ou libérer des ressources».

- **Etiq18 :** *Envoi_message* = «Réponse de la négociation ».
  Destination = «Agent négociateur».

3

3

- **Etiq20 :1-** *envoie_message*= « Rapport d'exécution sauvegardé »
  Destination= agent tâche émetteur du rapport d'exécution
  2- *envoi_message_interne* = «Rapport d'exécution » au **Module réalisation et évaluation de la réalisation.**
  3- Aller à : Etiq21.

- **Etiq22** : *envoie_message*= «demande ou ordre de modification du modèle de procédé logiciels » ;
  Destination= « si demande de modification alors destination =agent tâche sinon destination= agent fragment ».

- **Etiq25 :** *Envoi_message* = «Résultat final ou partiel de l'exécution du modèle de procédé logiciel ».
  Destination = «Agent superviseur ou sous chef de projets ».

Fin ;

- **Module gestionnaire du modèle de procédé logiciel**.

Début :

- **Etiq0 : 1-** Vérification du fragment modèle de procédé et les facteurs d'évaluations reçus.
  2- Envoi_*message_interne* = «modèle de procédé logiciel et facteurs d'évaluation du fragment » au **module gestionnaire de la base de connaissances.**
  3- Aller à : Etiq1.

- **Etiq3 : 1-** Sélectionner la partie du fragment du modèle de procédé à exécuter.
  2-Envoi_*message_interne* =« partie du fragment modèle prêt à exécuter » au **Module de décision et de planification des tâches.**
  3- Aller à : Etiq4.

- **Etiq9 : 1-** Consulter l'état d'avancement du fragment modèle de procédé

Si la partie concernée par les changements à été exécuté
Alors
1-*Envoie_message_interne*= « changements refusés + cause du refus : ici partie exécutée » Au **module communication.**
2-Aller à **Etiq10** ;
Sinon
1- Consulter les facteurs d'évaluations.
2- Consulter l'état des ressources,
3- consulter les agents tâches concernés,
4- Vérifier la possibilité d'introduire les changements.
5- Négocier les changements.
Si changements acceptés
Alors
1- Introduire les changements dans le fragment modèle de procédé.
2- Envoi *message_interne* = «changement du modèle de procédé effectué » au **Module gestionnaire de la base de connaissances.**
3- Aller à : Etiq11.
sinon
1-*Envoie_message_interne*= « changements refusés + cause du refus ici partie exécuté » Au **module communication.**
2-Aller à **Etiq10** ;
Findesi ;

1

Findesi ;

1

- **Etiq13 : 1-** Introduire les changements dans le fragment modèle de procédé.
  2- Envoi *message_interne* = «changement du modèle de procédé effectué » au **Module gestionnaire de la base de connaissances.**
  3- Aller à : Etiq11.

- **Etiq23 :<u>Si</u> fin d'exécution de tous le fragment modèle de procédé logiciel**

  <u>Alors</u> : 1- *Envoi_message_interne* = «Envoi du résultat final de l'exécution»au **module gestionnaire de la base de connaissances**
  2- Allez à Etiq24 :
  <u>sinon</u>
  Allez à Etiq3 :
  <u>Fin de si ;</u>

<u>Fin ;</u>

- **Module gestionnaire de la base de connaissances.**

<u>Début</u>

- **Etiq1 : 1-** Sauvegarde du fragment modèle de procédé global et des facteurs d'évaluations de l'agent fragment.
  2- *Envoi_message_interne* = « fragment modèle de procédé prêt à s'exécuter » au **Module gestionnaire du modèle de procédé logiciel.**
  3- Aller à Etiq3 :

- **Etiq6 : 1-**Sauvegarde des Assignations des séries de tâches séquentielles aux agents tâches
  2- *Envoi_message_interne* =«Assignation des fragments -des séries de tâches séquentielles- sauvegardées » au **Module réalisation et évaluation de la réalisation.**
  3- Allez à Etiq7 :

- **Etiq11 : 1-** Sauvegarde des changements du fragment modèle de procédé et des facteurs d'évaluations de l'agent superviseur.
  2- Envoi *message_interne* = «Changement effectué » au **Module communication.**
  3- Allez à Etiq12 :

- **Etiq14 : 1-** sauvegarde des mise à jour des états des ressources.
  <u>Si</u> Maj à partir de message externe

  <u>Alors</u>
  1- Envoi *message_interne* = «MAJ effectuée » au **Module communication.**
  2- Allez à Etiq15 :
  <u>sinon</u>
  1- Envoi *message_interne* = «MAJ effectuée » au **Module décision et planification des tâches .**
  2- Allez à Etiq17 :
  <u>fin de si ;</u>

- **Etiq19 : 1-**Sauvegarde du Rapport d'exécution.
  2- *Envoi_message_interne* = «Rapport d'exécution sauvegardé » au **Module communication.**
  3- Allez à Etiq20 :

1

1

- **Etiq24 : 1-** *Envoi_message_interne* = «Résultat final ou partiel d'exécution » au **Module communication.**
  2- Allez à Etiq25 :

Fin ;

- **Module de décision et de planification des tâches :**

Début :

- **Etiq4 : 1-** Fragmenter la partie du modèle de procédé logiciels en séries de tâches séquentielles.
  2- Etablir les facteurs d'évaluation pour chaque série de tâche séquentielle.
  3-envoi_*message_interne*= « modèle de procédé prêt à s'exécuter » au **Module réalisation et évaluation de la réalisation.**
  4- Aller à Etiq5.

- **Etiq16 : 1-** Consulter la stratégie utilisée.
  2- Vérifier l'état des ressources.
  3- Décision selon les objectifs à suivre et les priorités.

  Si changement dans l'état des ressources ou des accointances

  Alors 4- envoi *message_interne* = «Changement de l'état des ressources ou des accointances » au **Module gestionnaire des accointances et de ressources (interne et les autres) ;**
  5- Allez à Etiq13 :
  sinon
  1- envoi *message_interne* = « réponse message d'interaction »
  Au module communication.
  2- Allez à Etiq18 ;
  Fin de si ;

- **Etiq17 : 1-** envoi *message_interne* = « réponse message d'interaction »
  Au module communication.
  2- Allez à Etiq18 ;

Fin ;

- **Module réalisation et évaluation de la réalisation.**

Début :

- **Etiq5 : 1-** Initialiser les agents tâche nécessaires**.**
  2- Assigner les fragments et leurs facteurs d'évaluations aux agents tâches correspondant .
  3- *Envoie_message_interne* = « Assignation des fragments aux agents tâche» au **Module gestionnaire de la base de connaissances.**
  4- Allez à Etiq 6 :

- **Etiq7 :1-** *Envoi_ message_interne* = «Assignation des fragments et facteurs d'évaluations» au **Module communication.**
  2- Allez à Etiq8 :

1

1

- **Etiq21 : 1-** Evaluation des rapports d'exécutions.

  Si déviation du modèle instancié
  Alors
  1- Consultation de l'état des ressources,
  2- Consultation des facteurs d'évaluations,
  3- Consultation des priorités d'exécution.
  4- Proposition de changements dans le modèle de procédé logiciels.
  5- Selon la décision prise :
  - *Envoi_message_interne* = «proposition de changement du modèle de procédé logiciels» au **Module communication.**
  Destination = « agent tâche ».
  - *Envoi_message_interne* = « Changement du modèle de procédé logiciel» **Au module communication**
  Destination = « agent superviseur ».
  6- Allez à Etiq22 :
  Sinon
  Si Dernier Rapport d'exécution de la partie assignée.
  Alors
  1- *Envoi_message_interne* = «fin d'exécution de la partie assignée» au **Module gestionnaire du procédé logiciel.**
  2- Allez à Etiq23 :
  Fin de si ;
  Fin de si ;

Fin ;

- **Module gestionnaire des accointances et de ressources (interne et les autres).**

Début :

- **Etiq13 : 1-** MAJ de l'état des ressources.
  2- *Envoie message interne* = «Etat des ressources ou des accointances mis à jour» au **Module gestionnaire de la base de connaissances.**
  3- Allez à Etiq14 :

Fin ;

Fin.

IV- Architecture interne de l'agent tâche :

L'agent tâches est responsable de l'exécution du modèle de procédé logiciel, pour cela il établie des facteurs d'évaluations internes pour son usage personnel. Après exécution de ses tâches, il évalue les résultats et vérifie si les facteurs d'évaluations qu'il a établie sont respectées.

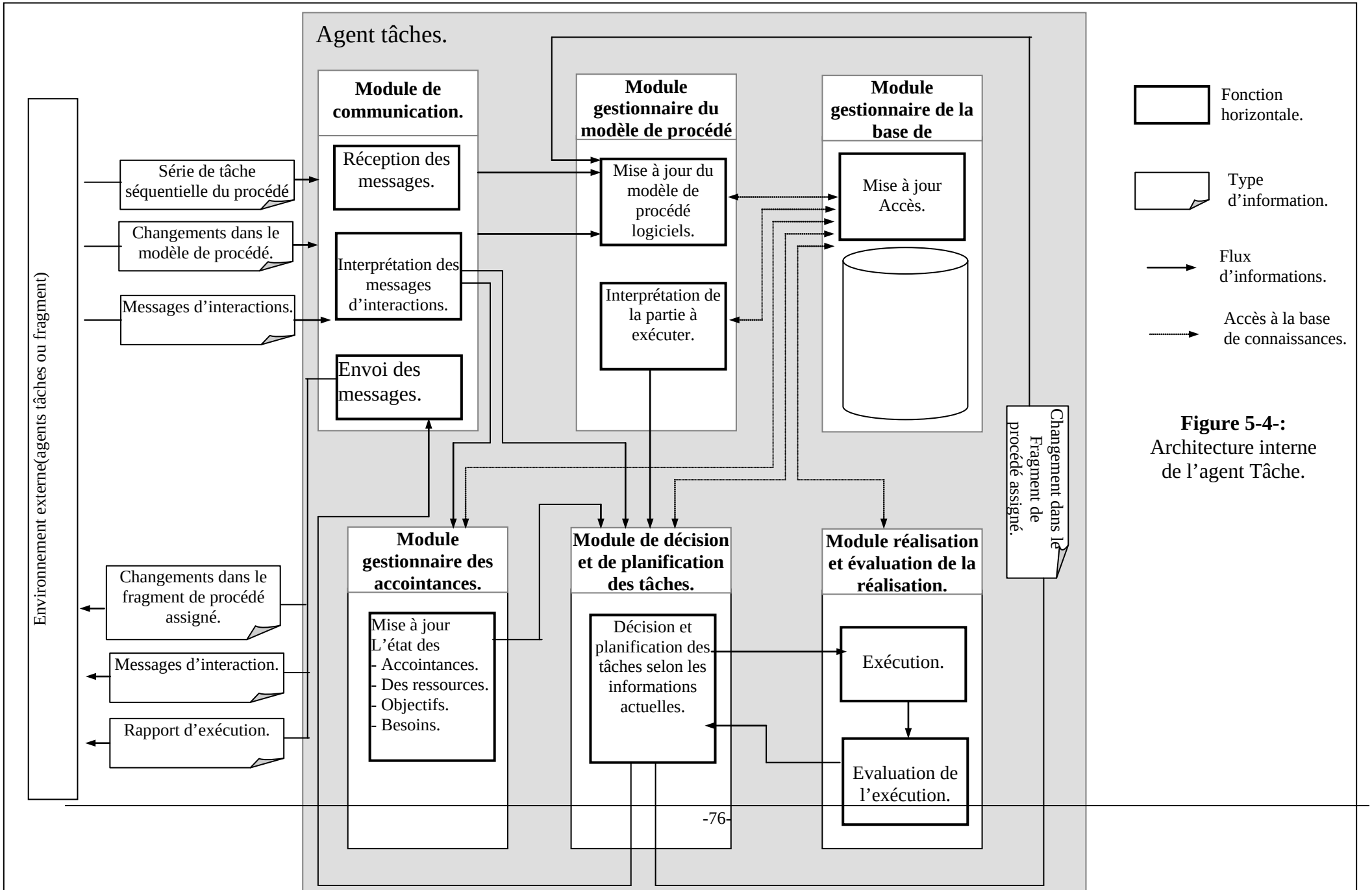


Figure 5-4-:
Architecture interne de l'agent Tâche.

1- Algorithme d'exécution de l'agent tâches :

DEBUT :

- **Module communication :**

Début :

➢ **Si** *message_reçu* **=** «Assignation de la série de tâches séquentielles et facteurs d'évaluations» et
source = « Agent fragments».
Alors 1- Envoi *message_interne* **=** «série de tâches séquentielles et facteurs d'évaluations »au **Module gestionnaire du modèle de procédé logiciel.**
2- Aller à : Etiq0.
Fin de si.

➢ **Si** *message_reçu* **=** «message d'interaction»
Alors
Si message = « libération ou ajout de ressources ou d'accointances »
Alors
1- *Envoi_message_interne* **=** «libération ou ajout de ressources »au **Module gestionnaire de ressources et d'accointances** .
2- Aller à : Etiq15.
Sinon
1- *envoi_ message_ interne*= « contenue du message » **au module décision et planification.**
2- Aller à : Etiq19.

Fin de si.

➢ **Si** *message_reçu* **=** «demande de changement du modèle de procédé logiciels»
Source = « Agent fragment »
Alors
1- *Envoie_message_interne* = « demande de changement de procédé logiciels » au **module gestionnaire du modèle de procédé logiciels.**
2- Allez à Etiq21 ;
Fin de si ;

➢ **Si** *message_reçu* **=** «Demande de consultation de résultats d'exécutions » et
source = «utilisateurs ou sous_chef de projet»
Alors 1- *envoi_message_interne* = «Demande de consultation de résultats d'exécutions » au **Module gestionnaire de la base de connaissances.**
2- Aller à : Etiq12.
Fin de si ;

➢ **Etiq6:** *Envoi_message* = «Demande de services ou de ressources, Réponse à une négociations».
Destination = «Agent tâche voisin ou agent fragment».

➢ **Etiq10:** *Envoi_message*= « message de déviation »
Destination = « Agent fragment».

1

1

- **Etiq13:** *Envoi_message*=« Rapport d'exécution final ou partiels de la série de tâche séquentielle »
  Destination = « Agent fragment».

- **Etiq17 :** *Envoi_message*=« état des ressources ou des accointances mis à jours »
  Destination = « Agent fragment ou tâche».

- **Etiq18** *: Envoi_message*=«changement dans la série de tâche de procédé logiciels »
  Destination = « Agent fragment».

- **Etiq20** *: Envoi_message*=«réponse message d'interaction »
  Destination = « Agent fragment ou agent tâche».

- **Etiq22** *: Envoi_message*= « changements refusés + cause du refus »
  Destination= « agent fragment ».

Fin ;

- **Module gestionnaire du modèle de procédé logiciel**.

Début :

- **Etiq0 : 1-** Vérification de la série de tâche séquentielle du modèle de procédé et les facteurs d'évaluations reçus.
  2- Envoi_*message_interne* = «modèle de procédé logiciel et facteurs d'évaluation du fragment » au **module gestionnaire de la base de connaissances.**
  3- Aller à : Etiq1.

- **Etiq3 : (si N est le nombre de tâches séquentielles de cette série alors)**
  i :=1 (la première tâche de la série séquentielles);

- **Etiq11 : Si** i : = 1 <= N
  Alors:
  1- Sélectionner la tâche 'i' de la série de tâches du modèle de procédé à exécuter.
  2- Envoi *message_interne* = « tâche prête à exécuter » au **Module de décision et de planification des tâches.**
  3- Aller à : Etiq4.
  Sinon
  1- Envoie message interne= « demande d'envoi du rapport d'exécution final de la série de tâche séquentielle au **Module gestionnaire de la base de connaissances.**
  2- Allez à Etiq12 ;
  Fin de si ;

- **Etiq9 : 1-** vérification des changements effectués et des nouveaux facteurs d'évaluations
  2- Introduction des changements ;
  en parallèle A et B :
  A) 1- *Envoie_message_interne* = « sauvegarde des changements dans la série des tâches » **au module gestionnaire de la base de connaissances;**
  2- Allez à Etiq14 ;
  B) 1- *Envoie_message_interne* = « Changements dans la série des tâches » **au module communication;**
  2- Allez à Etiq18 ;

1

1

- **Etiq21 : 1-** Consulter l'état d'avancement du fragment modèle de procédé
  Si la partie concernée par les changements à été exécuté
  Alors
  1-*Envoie_message_interne*= « changements refusés + cause du refus : ici partie exécutée » Au **module communication.**
  2-Aller à **Etiq22** ;
  Sinon
  1- Consulter les facteurs d'évaluations.
  2- Consulter l'état des ressources,
  3- consulter les agents tâches concernés,
  4- Vérifier la possibilité d'introduire les changements.
  5- Négocier les changements.
  Si changements acceptés
  Alors
  1- Introduire les changements dans le fragment modèle de procédé.
  En parallèle **A)** et **B)**

  A) 1- Envoi *message_interne* = «changement du modèle de procédé effectué » au **Module gestionnaire de la base de connaissances.**
  2- Aller à : Etiq14.

  B) 1- Envoi *message_interne* = «changement du modèle de procédé acceptés » au **Module communication.**
  2- Aller à : Etiq18.

  sinon
  1-*Envoie_message_interne*= « changements refusés + cause du refus ici partie exécuté » Au **module communication.**
  2-Aller à **Etiq22** ;

  Findesi ;

  Fin de si ;

Fin ;

- **Module gestionnaire de la base de connaissances.**

Début :

- **Etiq1 : 1-** Sauvegarde série de tâche séquentielle et facteurs d'évaluations»
  2- *Envoi_message_interne* = « série de tâches séquentielles sauvegardées » au **Module gestionnaire du modèle de procédé logiciel.**
  3- Aller à Etiq3 :

- **Etiq7 : 1-** Sauvegarde du rapport d'exécutions + résultats intermédiaires
  2- *Envoi_message_interne* = «rapport d'exécutions + résultats intermédiaires» au **Module réalisation et évaluation de la réalisation.**
  3- Aller à Etiq8 :

- **Etiq12 :1-** *Envoi_message_interne***=** « rapport d'exécution final de la série de tâche séquentielle » **au Module communication.**
  2- Allez à Etiq13 ;

- **Etiq14 : 1-** Sauvegarde des changements dans la série de tâche séquentielle et

1

les nouveaux facteurs d'évaluations»

2- **si** Changement à partir des rapports d'exécution internes
alors
1- *Envoi_message_interne* = « série de tâches séquentielles sauvegardées » au **Module gestionnaire du modèle de procédé logiciel.**
2- Aller à Etiq11 :
Fin de si ;

- **Etiq16 :** **1-** sauvegarde des mise à jour des états des ressources.
Si Maj à partir de message externe
Alors
1- Envoi *message_interne* = «MAJ effectuée » au **Module communication.**
2- Allez à Etiq17 :
Fin de si ;

fin ;

- **Module de décision et de planification des tâches :**

Début :

- **Etiq4 :**
1- consulter les ressources requises pour l'exécution de cette tâche ;
2- consulter les facteurs d'évaluation ;
3- consulter les ressources disponibles
si toutes les conditions d'exécution sont vérifiées
alors :
en parallèle A et B
A) 1- envoi *message_interne= «* mise a jour de l'état des ressources » au **Module Gestionnaire des ressources.**
2- Aller à Etiq15.

B)1-envoi *message_interne= «* tâche prête à s'exécuter » au **Module réalisation et évaluation de la réalisation.**
2- Aller à Etiq5.

sinon
1- Demande de ressources ou de services .
2- *Envoie_message_interne=* « demande de services ou de ressources » au **Module Communication** ;
3- Allez à Etiq6 ;
fin de si ;

- **Etiq19** **1-** Lire le contenue du message ;
Si confirmation acquisition de ressources
Alors
Allez à Etiq4 ;
Sinon
1- Consulter la de négociation stratégie utilisée.
2- Vérifier l'état des ressources.
3- Décision selon les objectifs à suivre et les priorités.

2 1

2 1 **Si** changement dans l'état des ressources ou des accointances
Alors
En parallèle A) et B)
A) 1- envoi *message_interne* = «Changement de l'état des ressources ou des accointances » au **Module gestionnaire des accointances et de ressources (internes et les autres) ;**
2- Allez à Etiq15 :
B) 1- *envoi_message_interne* = « réponse message d'interaction »
Au module communication.
2- Allez à Etiq20 ;
sinon
1- envoi *message_interne* = « réponse message d'interaction »
Au module communication.
2- Allez à Etiq20 ;
Fin de si ;

Fin de si ;

Fin ;

- **Module réalisation et évaluation de la réalisation.**

Début :

- **Etiq5 :**
  1- Exécution de la tâche ;
  2- *envoi_message_interne* = « rapport d'exécutions + résultats intermédiaires »
  au **Module gestionnaire de la base de connaissances.**
  3- Allez à Etiq7 ;

- **Etiq8 : 1-** Evaluation du rapport d'exécution et des résultats intermédiaires ;
  Si déviation par rapport au facteurs d'évaluations
  Alors
  Traitement de le déviation du modèle exécuté
  Si solution trouvé
  alors
  1- Modification dans la série des tâches du modèle de procédés,
  2- Envoie message interne = « modifications dans la série de tâche séquentielles »
  au **Module gestionnaire du modèle de procédé logiciels.**
  3- Allez à Etiq9 ;
  sinon
  1- *Envoie_message_interne* = « message de déviation » **au Module communication.**
  2- Allez à Etiq10
  Fin de si ;

  Sinon
  i := i+1 ;
  Allez à Etiq11 du module ;
  Fin de si ;

Fin ;

- **Module gestionnaire des accointances et de ressources (interne et les autres).**

Début :

- **Etiq15 : 1-** MAJ de l'état des ressources.
  2- *Envoie message interne* = «Etat des ressources ou des accointances mis à jour» au **Module gestionnaire de la base de connaissances.**
  3- Allez à Etiq16 :

Fin ;

FIN.

Conclusion :

Les algorithmes définis dans cette partie sont écris de manière globale; Plusieurs point reste à développer tel que les stratégies de négociations utilisé, le chois des stratégies, la planifications des tâches, la manière d'introduire les changements...etc.

Nous n'avons pas approfondie ces points de manière délibérée, car c'est la structure générale qui nous intéresse et ces points peuvent être développés ultérieurement.

Introduction :

Dans ce dernier chapitre nous présentons les connaissances que doit avoir un agent modèle de procédé.

Les connaissances présentes dans la base de connaissances de l'agent modèle de procédé concernent le modèle de procédé logiciels et l'état interne de l'agent et de ces voisins.

Pour la présentation des connaissances de l'agent nous avons opté pour le langage de modélisation UML ; La diversité de ces diagrammes et la possibilité de modéliser un système pris de différents angles est un avantage non négligeable pour notre système; Il nous permet de représenter les concepts utilisés (les modèles de procédés logiciels, l'agent et les interactions de l'agent) avec le même formalisme tout en conservant une cohérence de représentation.

Le modèle de procédé est représenté par deux diagrammes de UML (le diagramme de classe et le diagramme de séquence) ; le choix de ces diagrammes s'est basé sur les travaux effectués par. Cependant il faut souligner que c'est le concept et le choix des diagrammes qui nous intéressent, ainsi des modifications sont apportées à la plupart des diagrammes afin d'introduire la notion d'agent et de ces interactions.

Chaque type d'agent a son propre diagramme de classes qui représente la vue statique d'un fragment du modèle de procédé assigné. L'union de ces diagrammes de classes représente le modèle de procédé logiciels dans tous ces détails.

Le diagramme de séquences est important au niveau de l'agent superviseur car il permet de définire les interactions inter espaces de travail.

Les interactions de base de l'agent modèle de procédé sont représentées avec le diagramme d'activités ; ce diagramme est le plus approprié pour montrer les agents en interactions et le contenue des messages et les activités effectués par chaque agent.

I. Définition des connaissances de l'agent modèle de procédé logiciels :

La base de connaissance de l'agent modèle de procédé contient :

- Le diagramme de classes représentant la vue statique du modèle de procédé logiciels.
- Le diagramme de séquences représentant la vue dynamique du modèle de procédé logiciels.
- Le diagramme états/transitions pour définir les l'états des agents subordonnés.
- Le diagramme d'activités pour modéliser les interactions de base de l'agent modèle de procédé, ses voisins et ses activités.
- L'agent tâche à un diagramme d'états/transitions pour représenter les états de ses ressources, et un autre pour représenter l'état de l'activité du modèle de procédé logiciels.

II. Les diagrammes de classes des agents modèle de procédé :

1-Diagramme de classes du modèle de procédé logiciels de l'agent superviseur:

Les objets définis dans ce diagramme représentent une vision globale du modèle de procédé, la vision de l'agent superviseur ; Les détails ne sont pas tous représentés, seul les informations manipulable à ce niveau sont nécessaires.

Les agents fragments sont le seul type d'agent définis dans ce diagramme, puisque les agents tâches n'ont aucun lien direct avec l'agent superviseur, il n'est pas utile de les définir à ce niveau.

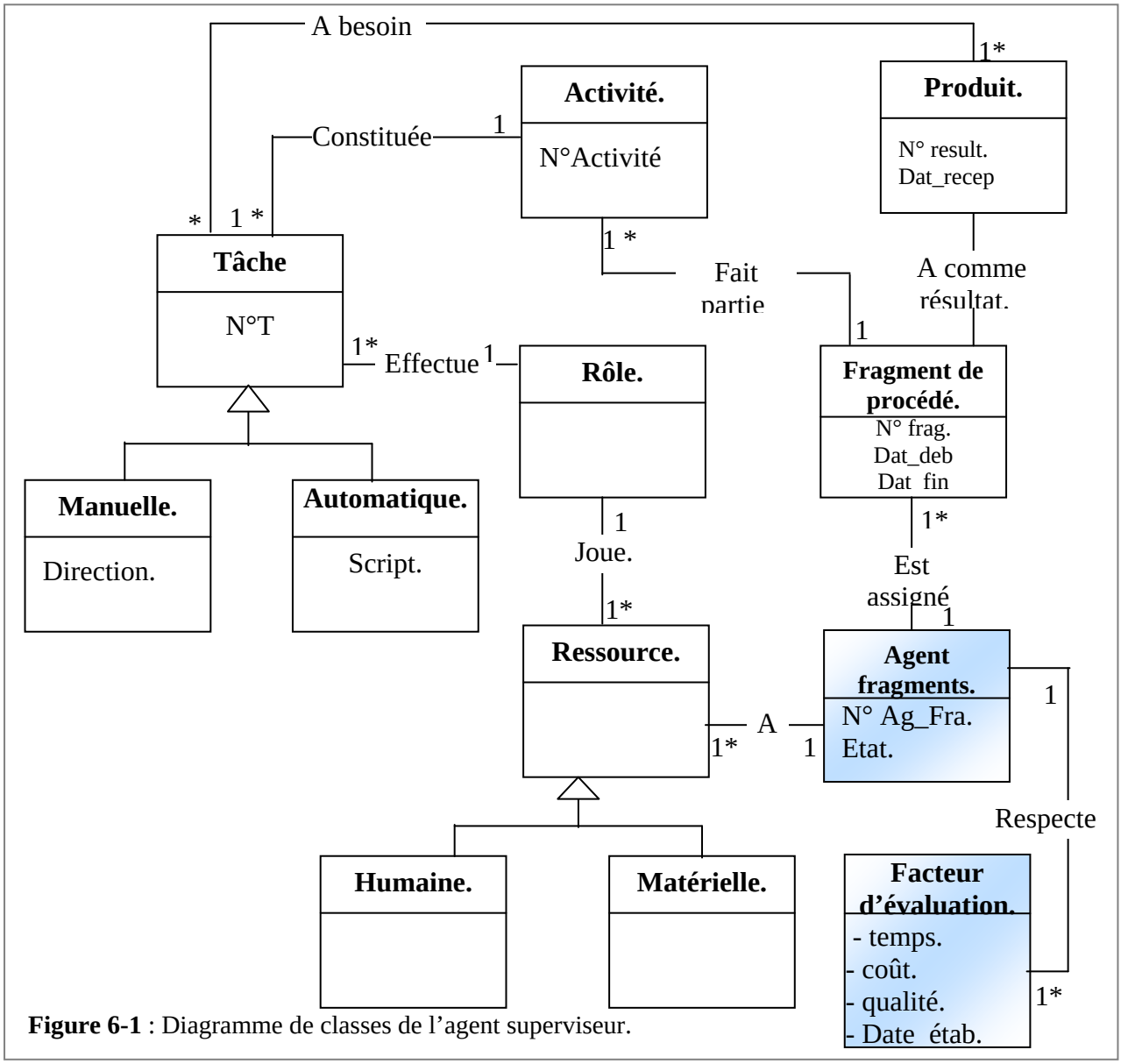


Figure 6-1 : Diagramme de classes de l'agent superviseur.

La classe « facteur d'évaluation » contient les facteurs d'évaluation initialement établis par l'agent superviseur (pour chaque agent fragment), ils concernent chaque fragment de procédé assigné et désignent les conditions d'exécution à respecter. Ces facteurs d'évaluation ne sont pas statiques et peuvent être mis à jour par les agents fragments pendant l'exécution de leur fragment de procédé logiciel.

La classe fragment de procédé contient toutes les instances de fragment de procédé définis même celles qui déjà exécutées.

2- Diagramme de classes de l'agent fragments:

Ce diagramme fait partie des connaissances de l'agent fragment. Il représente la vue statique d'un fragment modèle de procédé logiciel assigné à un agent fragment.

La classe « agent tâches » apparaît, elle représente les agents tâches initialisées par l'agent fragment.

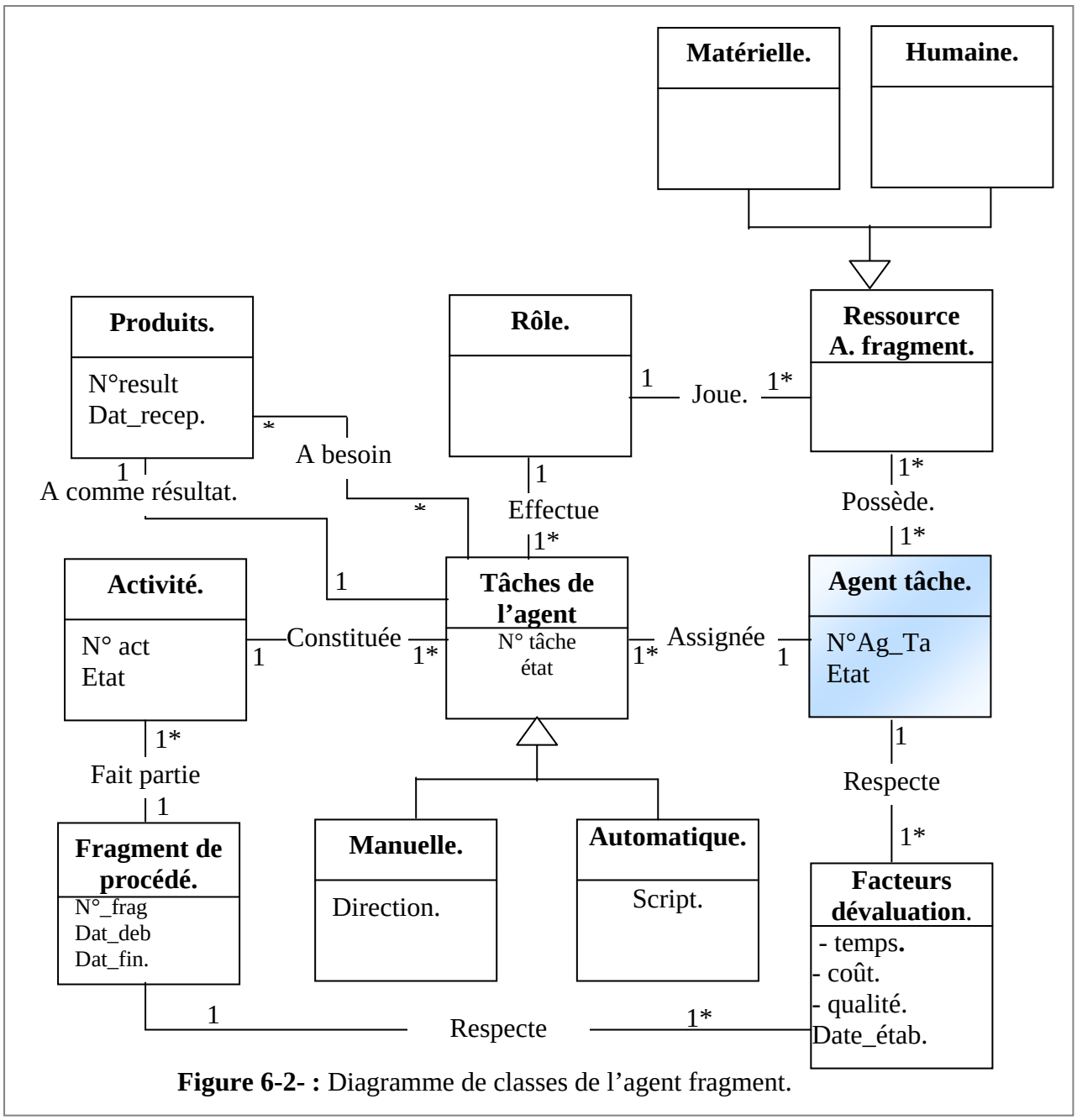


Figure 6-2- : Diagramme de classes de l'agent fragment.

Les classes « fragment de procédé », « activité tâches » et « ressources » contiennent que les instances qui concerne cet agent fragment.

La classe facteurs d'évaluation contient en plus des facteurs passés par l'agent superviseurs des facteurs d'évaluations établis par l'agent fragment pour ses agents tâches. Ces facteurs concernent les séries de tâches séquentielles assignées aux agents tâches.

3- Diagramme des classes de l'agent tâche :

Ce diagramme de classes fait partie des connaissances de l'agent tâches, il représente la vue statique des séries de tâches séquentielles du modèle de procédé assignées ainsi qu'une vue partielle de l'environnement de l'agent tâches (voisins et leurs ressources).

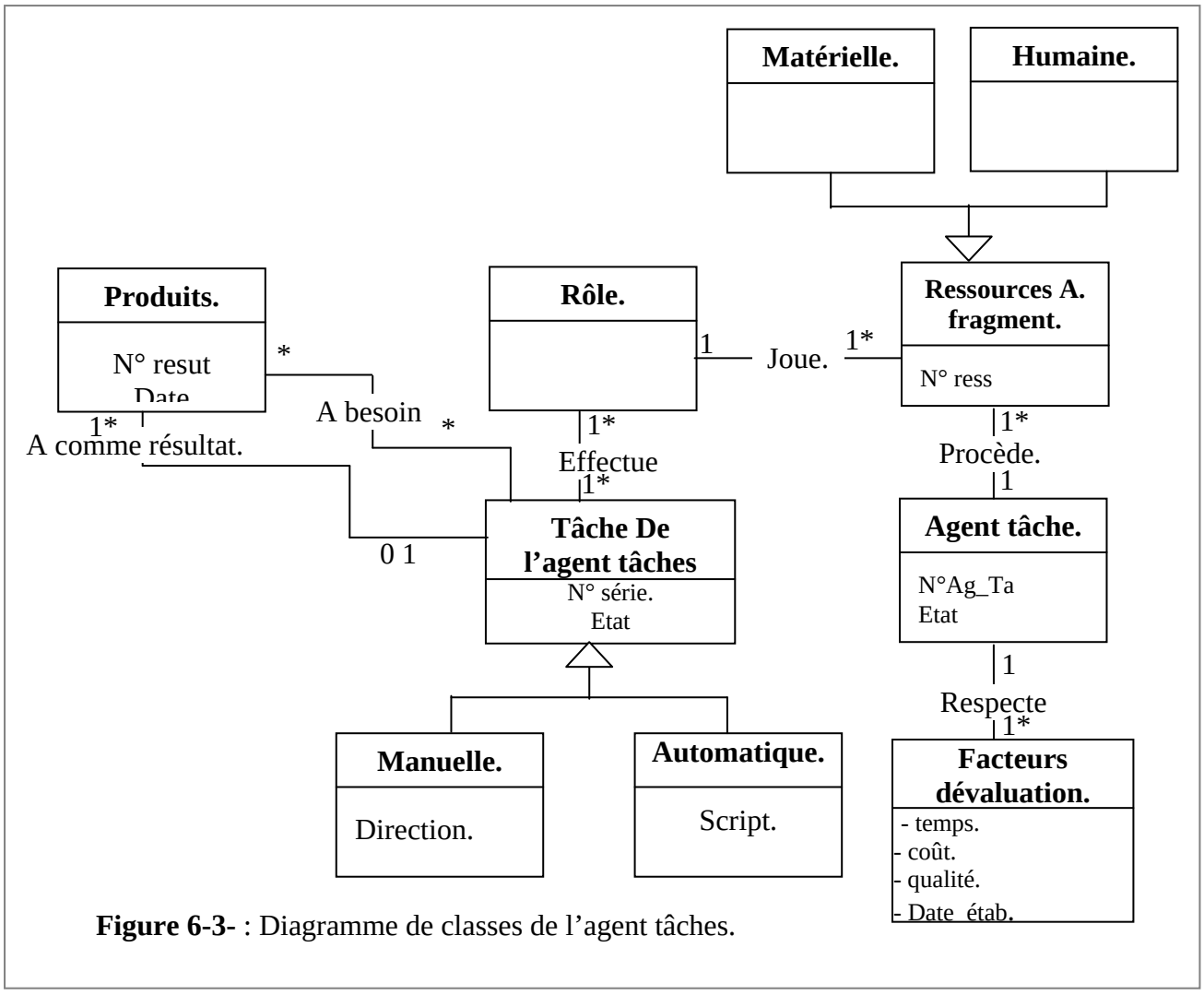


Figure 6-3- : Diagramme de classes de l'agent tâches.

La classe « agent tâches » contient tous les agents tâches voisins et leurs assignations. Elle représente les accointances de l'agent tâches et leurs états.

Les classes « activité », «tâches » et « ressources » contiennent les instances qui concerne cet agent tâches et des instances qui concerne ces voisins.

III. Le diagramme de séquences des agents modèles de procédé logiciels :

1- Diagramme de séquences de l'agent superviseur :

Le diagramme de séquence permet de représenter des collaborations entre objets selon un point de vue temporel. On y met l'accent sur la chronologie des envois de messages. Il décrit le scénario du déroulement et du séquencement des différentes activités **[30].**

En plus de la définition des enchaînements des activités du modèle de procédé, le diagramme de séquences permet à l'agent superviseur de définir les fragments de procédé logiciels et les facteurs d'évaluations de chaque fragment, en respectant les contraintes temporelles. Aussi il contribue dans la pré définition des interactions agents fragment, agent superviseur.

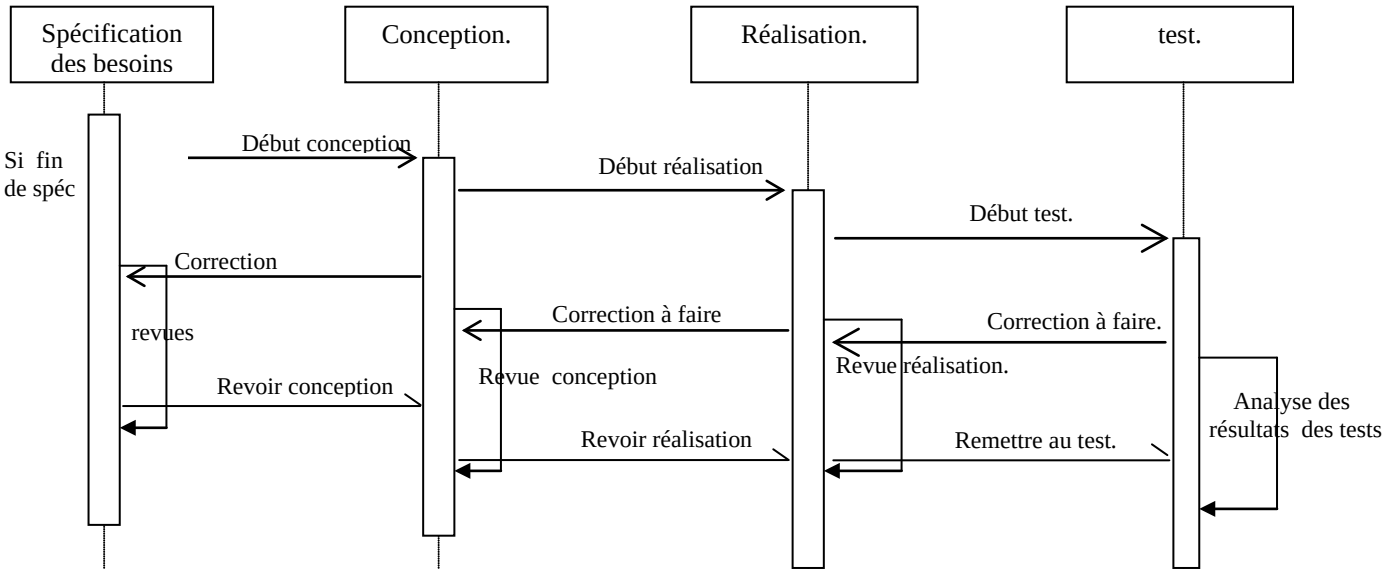


Figure 6-4- : Diagramme de séquences de l'agent superviseur.86

2- Diagramme de séquences de l'agent fragment :

Le diagramme de séquences de l'agent fragment n'est qu'une partie de celui de l'agent superviseurs. Il évoque les activités affectées à l'agent fragment. Il contribue dans la déterminations des facteurs d'évaluations des séries de tâches séquentielles des agents tâches.

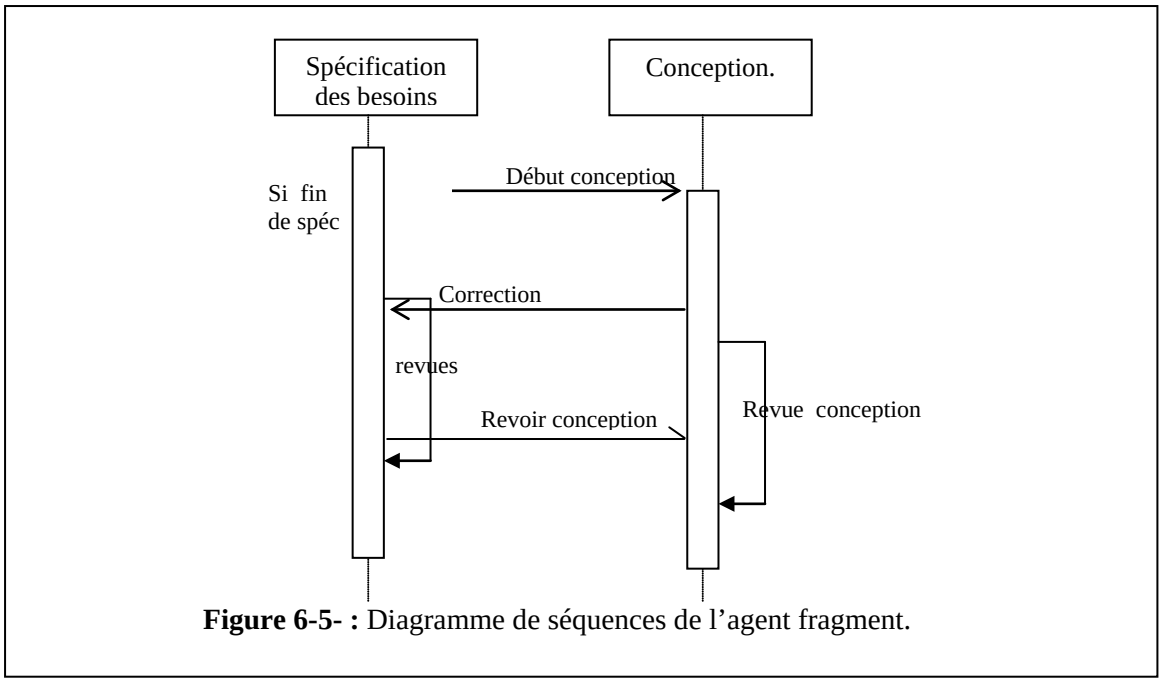


Figure 6-5- : Diagramme de séquences de l'agent fragment.

IV. Diagrammes d'activités des agents modèle de procédé logiciels :

1-Diagramme d'activités de l'agent superviseur :

UML permet de représenter graphiquement le comportement d'une méthode ou le déroulement d'un mécanisme à l'aide de diagrammes d'activités.

Le diagramme d'activités permet de représenter les activités internes de l'agent modèle de procédé et les interactions essentielles avec ses voisins.

Les couloirs d'activités représentent dans la plupart des cas les voisins de l'agent qui peuvent être des agents, chefs ou sous-chefs de projet.

Sachant que chaque type d'agent a des objectifs qui régissent son comportement et des voisins distincts, il est naturel d'avoir des diagrammes d'activités qui varie d'un type d'agent à un autre. Ainsi les couloirs d'activités de l'agent superviseurs sont les agents fragments et le chef de projets.

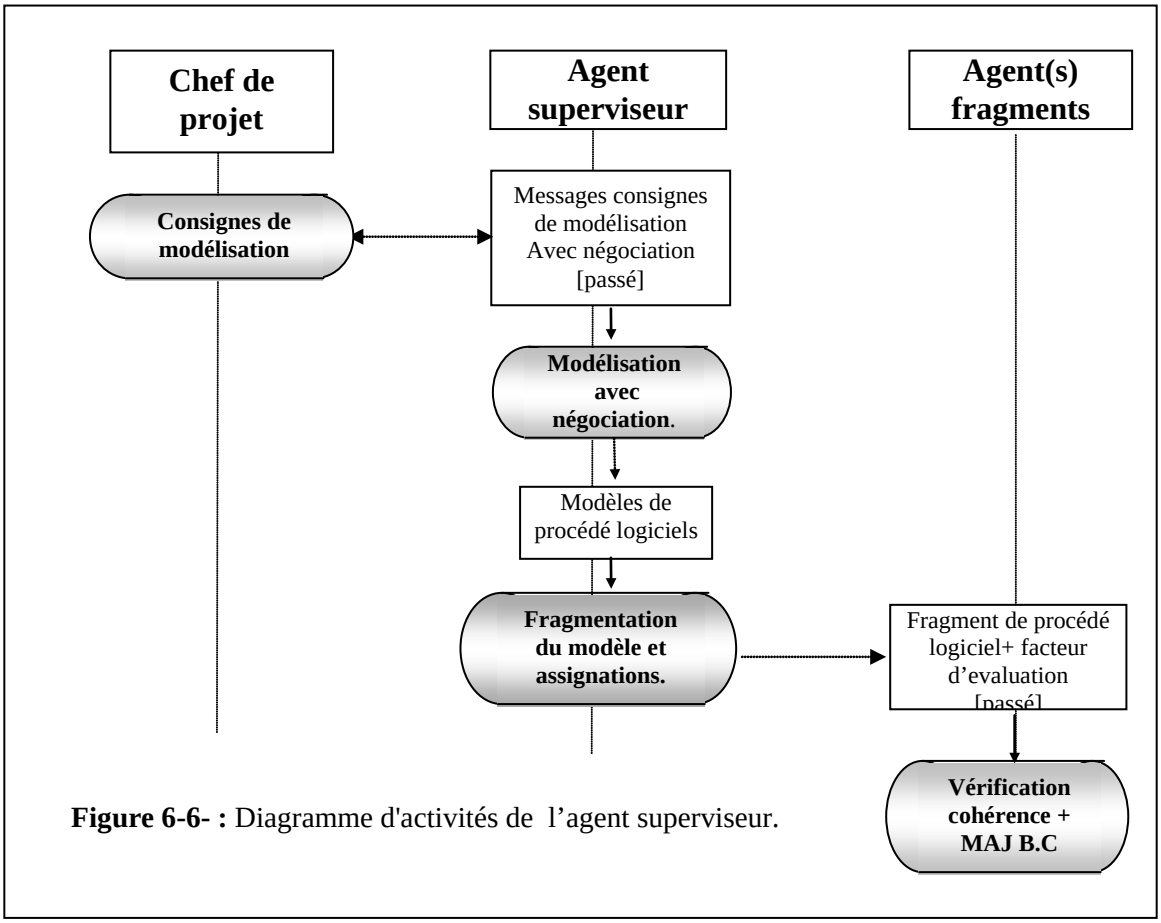


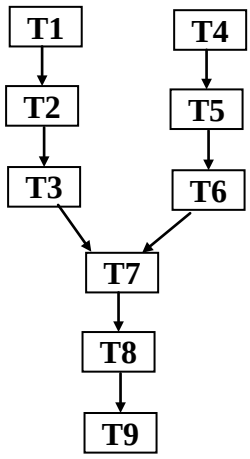
Figure 6-6- : Diagramme d'activités de l'agent superviseur.

2-Diagramme d'activités de l'agent fragment:

Le diagramme d'activités de l'agent fragment dépend du fragment assigné, ainsi Le nombre de « fragmentation » varie d'un agent fragment à un autre selon le nombre de tâches séquentielles et parallèles.

Le nombre d'agent tâches initialisés découle du nombre maximum de branches parallèles du fragment de procédé.

Exemple : si l'enchaînement des tâches du fragment de procédé assigné est comme suit :



Deux agents tâches sont initialisés, et deux activités de fragmentation sont effectuées ; le diagramme d'activités est comme suit :

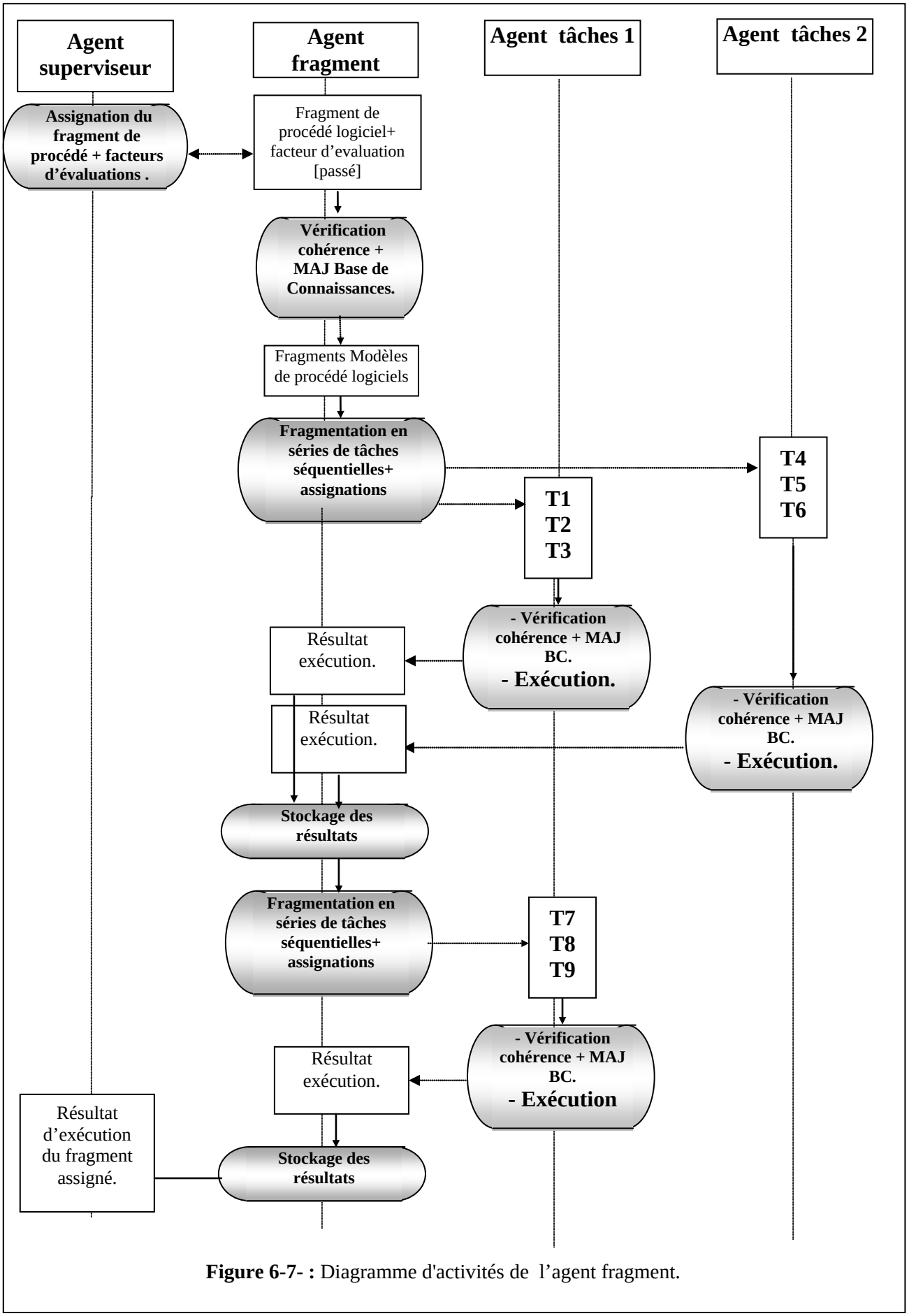


Figure 6-7- : Diagramme d'activités de l'agent fragment.

3- Diagramme d'activités de l'agent tâches :

Ce diagramme montre les activités internes de l'agent tâches, ces activités sont les activités de base de l'agent tâches les négociations, demandes et offres de services ne sont pas illustrés dans ce diagramme car elles ne sont pas prévisibles et dépendent de l'état actuelle de l'environnement et des ressources disponibles.

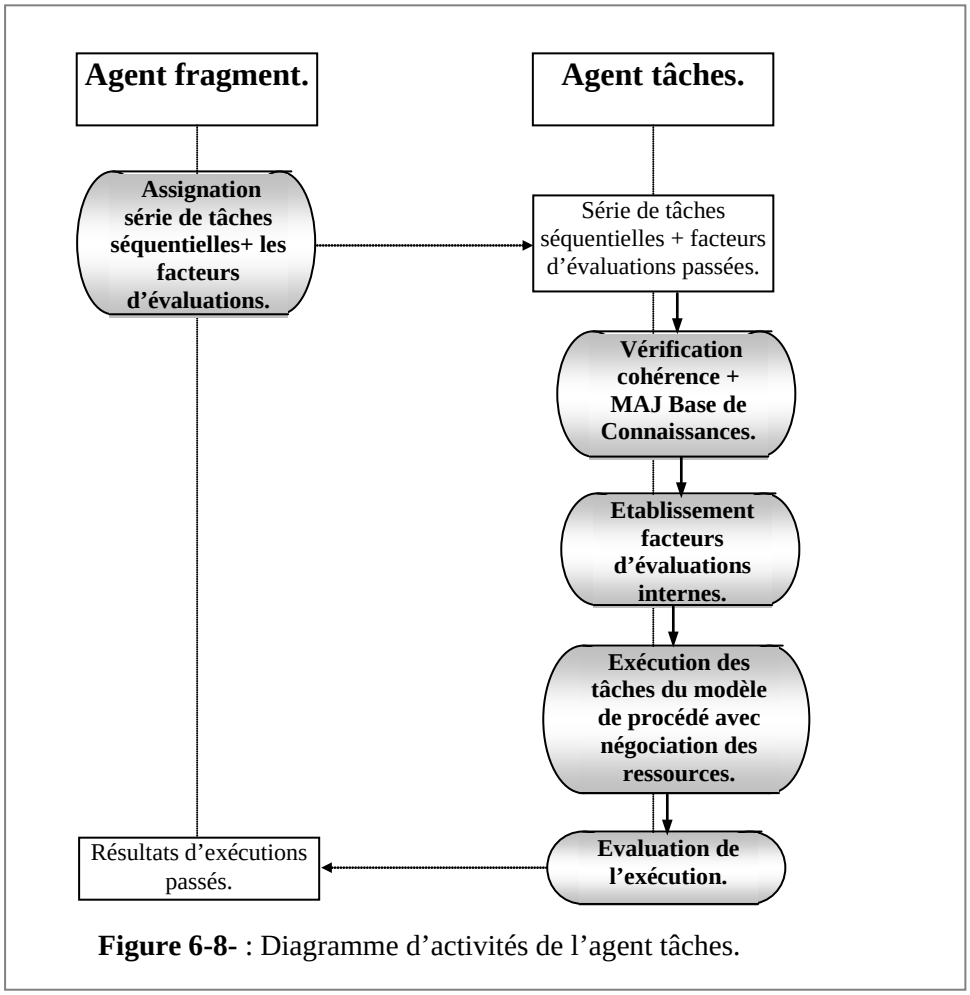


Figure 6-8- : Diagramme d'activités de l'agent tâches.

V- Diagramme états /transitions décrivant l'état des agents modèle de procédés:

1-Diagramme d'états/transitions de l'agent fragment (fait partie des connaissances de l'agent superviseur).

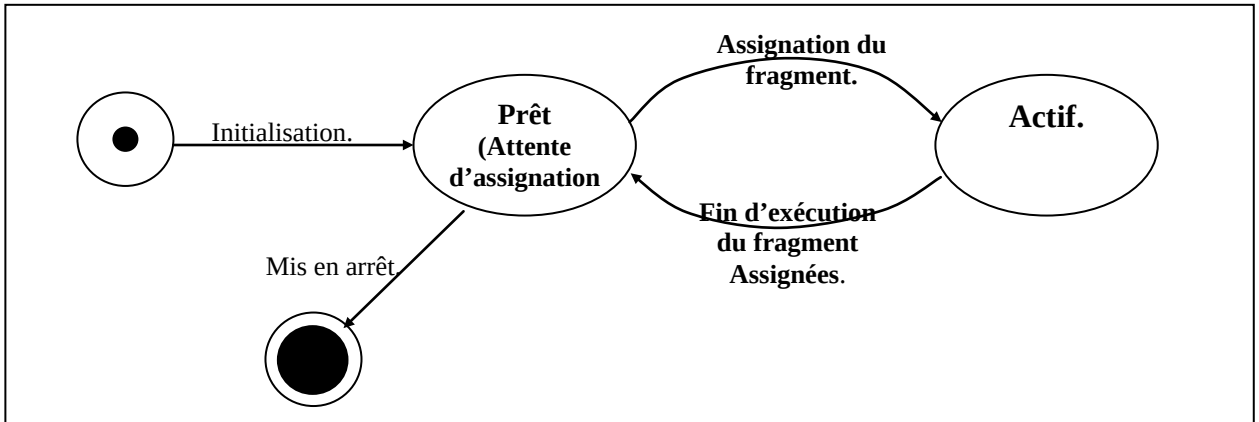


Figure 6-9- : Diagramme d'états/transitions de l'agents fragment (fait partie des connaissances de l'agent superviseur).

L'agent fragment peut être prêt ou actif il ne peut pas être bloque ou en attente de ressources car l'allocation des ressources ne se fait pas à son niveau mais au niveau inférieur.

Le diagramme état/transitions de l'agent superviseurs est similaire à celui de l'agent fragment et fait partie des connaissances de l'agent superviseur, l'intérêt de ce diagramme et de permettre au chef de projet d'accéder aux informations relatives à l'état de l'agent superviseur.

2- Diagramme états/transitions de l'agent tâches (partie des connaissances de l'agent fragment).

Par rapport au diagramme « état/transitions » précédent, un nouvel état est introduit (bloqué en attente de ressources) ; dans cet état l'agent n'est pas passif il reste toujours actif ; il peut interagir avec ses voisins et mettre à jour ces connaissances de manière continue. C'est l'état d'avancement de l'exécution du modèle de procédé qui est bloqué en attente de ressources.

Ce diagramme peut être nommé diagramme « états/transitions » des activités du modèle de procédés logiciels car il reflète parfaitement l'état des activités du modèle de procédé logiciel.

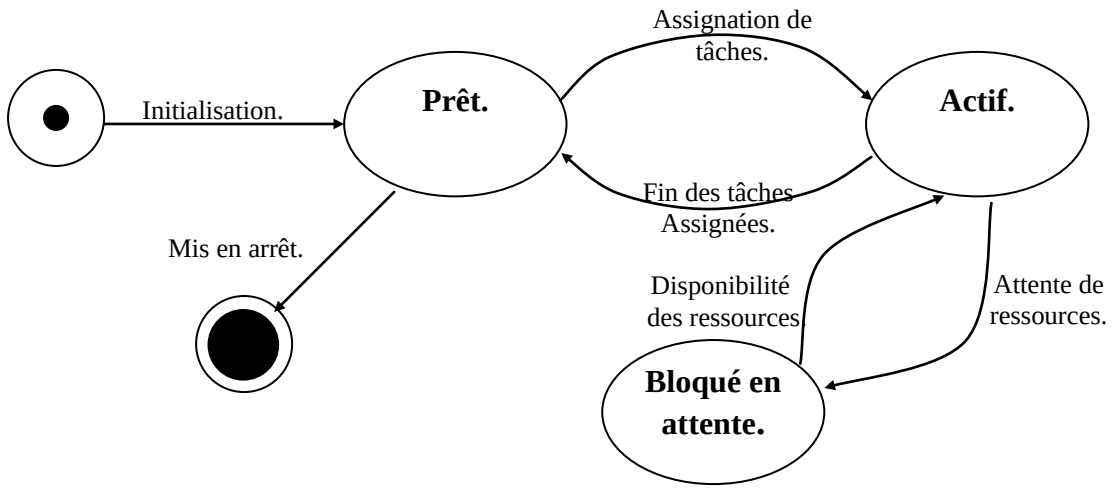


Figure 6-10- : Diagramme états/transitions de l'agent tâches (il fait partie des connaissances de l'agent fragment).

3- Diagramme états/transitions pour définir l'état des ressources de l'agent tâche :

Ce diagramme reflète l'état des ressources de l'agent tâches. En plus des états standards « allouées et disponibles », les ressources peuvent être « réservées » en d'autres termes, retirées des ressources disponibles pour une utilisation ultérieure ou non disponibles cad en panne ou en réparation.

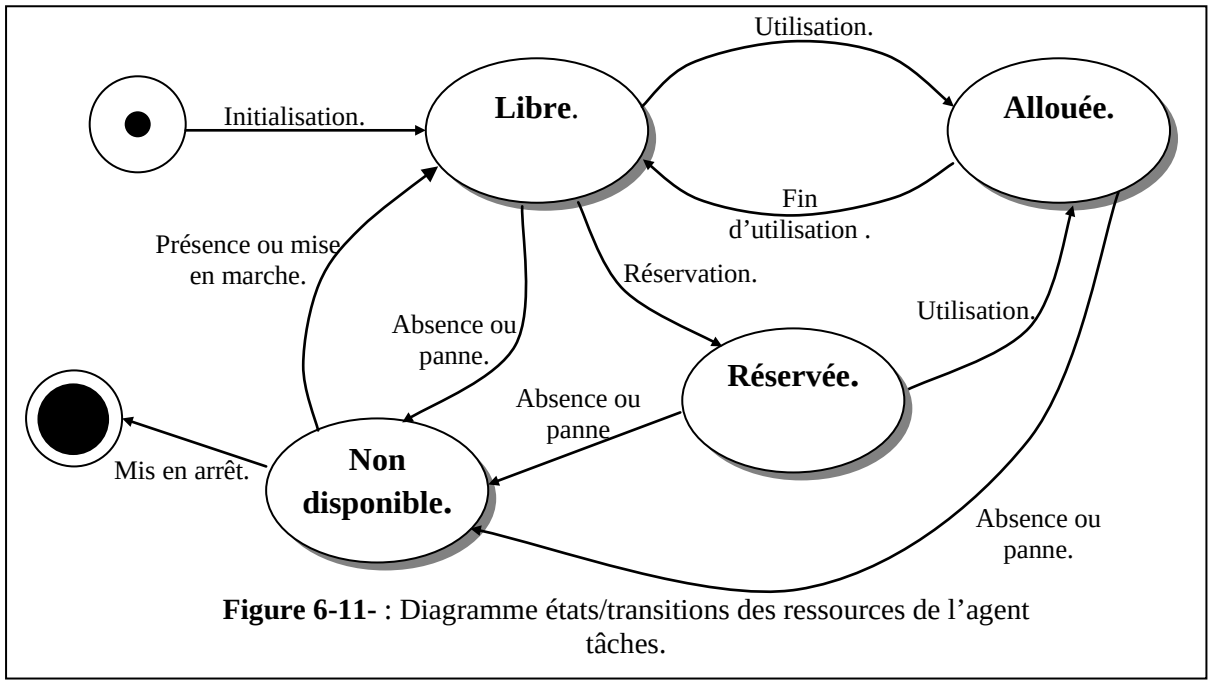


Figure 6-11- : Diagramme états/transitions des ressources de l'agent tâches.

Les diagrammes présentés montrent les connaissances de base que doit avoir l'agent modèle de procédé logiciel, les stratégies de négociations ne sont pas illustrées par ces diagrammes vu que Beaucoup de travaux ont été effectués dans ce domaine, de plus, les stratégies de négociations des agents représentent à eux seul un domaine de recherches très vaste qui ne peut être maîtrisé en quelques paragraphes.Ce qui est certain, c'est que les diagrammes présentés (particulièrement les diagrammes d'activités) doivent être en connexion avec ces stratégies de négociations, les digrammes d'activités doivent permettre l'invocation et exécution des stratégies de négociations en toute aisance.

CONCLUSION GENERALE :

La complexité croissante des logiciels, insuffisamment prise en charge par les méthodes "traditionnelles" et centralisées de développement, conjuguée à l'évolution tout aussi croissante de la technologie, ont contribué à l'arrivée de nouveaux modes et méthodes de développements basés sur l'utilisation optimale des ressources humains et logicielles-environnementales distribuées. A la centralisation réponds la distribution géographique des équipes de développeurs, à la "compatibilité" réponds "l'hétérogénéité" des environnements de développements.

Parmi ces modes, l'utilisation d'agents intelligents pour l'ingénierie des logiciels présente un intérêt certain par ses capacités d'autonomie, d'action et d'interactions entre les espaces de travail. ; ce mode nous offre l'opportunité de l'étudier en un de ses aspects qui a jusque là rencontré peu d'intérêt : ses capacités représentationnelles pour la modélisation des procédés logiciels .

Un étude de "l'existant", effectuée sur quatre environnements, PEACE+, CAGIS, ALLIANCE, et ADEPT nous montre que les priorités de ces systèmes tendaient vers la gestion de l'exécution du modèle de procédés et non pas vers sa modélisation, alors que des capacités de l'agent qui peuvent prendre en charge la modélisation restent inexploitées.

Notre propre contribution présente un modèle de procédé système multi-agents dans lequel l'intégration, partielle ou totale, du modèle de procédé dans les bases de connaissances des agents permet de représenter chaque agent comme une parie du procédé Ainsi, aux capacités d'action et d'interaction de l'agent, s'ajoutent des capacités d' interprétation et d'exécution de ce modèle de procédé.

L'objectif de notre approche est d'utiliser l'agent en tant que support et non en tant que formalisme de modélisation, car si l'agent ne peut rivaliser avec des formalismes de modélisation puissants tel que UML, il peut par contre être très efficace pour le contrôle de la cohérence de la modélisation et utiliser des formalismes de modélisations existant.

Notre système multi-agents est hiérarchiquement structuré en vue de la distribution du contrôle et des responsabilités sur tous les agents du système. Ainsi chaque agent a une liberté de décision et d'action sur ses tâches. Cette structure est avantageuse grâce à l'aspect «coopératif » de l'agent, tous les agents étant par ailleurs «sincères » et coopèrent pour réaliser un objectif en commun, ce qui n'est pas toujours vérifié pour une hiérarchie d'agents «humains ».

La représentation des connaissances de l'agent modèle de procédé logiciel que sont les fragments de procédés, des modèles d'interactions des agents, l'état des ressources et des agents dans les bases de connaissances sont définis en UML qui offre une grande diversité de diagrammes. Plusieurs concepts sont utilisés et définis en même temps et UML nous donne plus de liberté dans la modélisation des connaissances de notre système.

Dans notre thèse, nous définissons une nouvelle approche pour la modélisation et l'exécution des modèles de procédé logiciels ; une structure générale du modèle de procédé logiciel système multi-agents est établie, cependant les détails de programmations reste à définir.

Comme perspective de recherche d'autres types d'agents seront définis pour gérer d'autres aspects du modèle de procédé logiciels tel que l'hétérogénéité des outils ou des langages de communications. Les stratégies de négociations et les algorithmes de fragmentation doivent être définis. La mobilité des agents peut être exploitée pour la recherche de ressources ou pour supporter la mobilité des fragments de procédés logiciels. Enfin, et parmi les points les plus importants, une simulation de ce système reste à réaliser afin de valider de travail effectué, ce qui peut être réalisé en attribuant des projets de PFE aux étudiants de 5 emme années informatique.

[1] Ilham ALLOUI, Sami Beydéda, S. Cimpam, Flavio Oquendo.
« service avancé pour l'évolution des procédés : support de décision et de contrôle »
2001 Society for Design and Process SciencePrinted in the United States of America
Transactions of the SDPS DECEMBER 2001, Vol. 5, No. 4, pp. 39-53

[2] Ilham Alloui,Sorana Cîmpan,Flavio Oquendo,Hervé Verjus
« Une structure logiciel pour la modélisation, l'exécution et le contrôle flou des procédés logiciels intensif ».
LLP/CESALP Laboratory
ESIA Engineering School
University of Savoie at Annecy, France, 2001.

[3] Akira AIBA, Kazamasa Yokota et Hirochi Tsuda
« système résolveurs de problèmes coopérative distribué et hétérogène HELIOS et ses mécanismes de coopération ».
institute of new generation computer technology.
Shiba Tokyo 1995.

[4] Olivier Boissier
« La communication et la coopération dans les systèmes à agents La coordination dans les SMA ». 02 décembre 1999.

[5] Olivier Boissier :
« *La communication et la coopération dans les systèmes à agents :L'interaction dans les SMA ». 02 décembre* 1999.

[6] Noureddine Belkhatir, Jacky Estublier et Walcelio L.Melo :
« Travail coopératif dans une large gamme de systèmes logiciels ».

[7] M.Belmesk, N.Hank, H.Khelifa
« Architecture multi-agents pour le contrôle de conduite de processus ».1996.

[8] S.Dami, J.Estublier, A.Amiour :
« APEL : formalisme graphique et exécutable pour la modélisation de procédés ». Mars 1997.

[9] Jacques Ferber : Laforia, université de Pierre et Marie Curie,
« Les systèmes multi-agents : vers une intelligences collective »
interEditions 1997.

[10] Nickoolas R. Jennings
« Approche basée agent pour la construction des systèmes logiciels complexes »
Communication d'ACM avril 2001 Vol44.no4

[11] N. R. Jennings , P. Faratin, M.J. Jhonson , T. J. Norman , P. O'Brien et M.E. Wiegand.
« Gestionnaire de procédés business basés agents ».
Int Journal of Cooperative Information Systems **5** (2&3) 105-130, 1996.

[12] N. R. Jennings 1 , T. J. Norman , P. Faratin , P. O'Brien and B. Odgers
« Agent autonome pour un gestionnaire de procédés busness »
1 Dept. Electronic Engineering, Queen Mary & Westfield College,
University of London, London E1 4NS, UK.2000.

{N.R.Jennings, P.Faratin, T.J.Norman}@qmw.ac.uk
2 BT Research Labs, Martlesham Heath, Ipswich, Suffolk IP5 7RE, UK.
{paul, briano}@info.bt.co.uk

[13] Grégor Joeris, Christoph Klauck, Otthein Herzog
« Gestionnaire de procédé distribué et dynamique basé technologie d'agents »
Université de Bremen.

[14] A.Mostfaï
« la fédération dans les environnements centré procédés logiciels »
thèse de magister (chapitre1) ; USTHB ; 2002.

[15] Jie Meng, Sum Helal et Stanley Su.
« L'architecture des systèmes workflow Ad-Hok basé sur les agents mobiles et l'exécution basée règles »
Computer of information science and engineering
Univercity of Florida Cainsville FL 32611

[16] Peiwei Mi et Walt Scacchi :
« Processus d'intégration dans les environnements CASE ».
Computer science department and decision systems department
university of southern califonia ;
Los Angeles.

[17] D.Ouelhadj :
« Système multi-agents pour le pilotage distribué de cellules flexibles de production ».
Résumé de la thèse de magister.
Institut national de formation en informatique INI. Oued smar
1999/2000.

[18] A.Tmedjdoubene, A.Melaab :
« systèmes multi-agents pour l'ordonnancement de tâches dans les cellules flexibles de production »,USTHB
septembre 1998.

[19] Alf Inge Wang :
« Implémentation d'une architecture multi-agents pour l'ingénierie logiciel coopérative ».
In twelfth international conference on software ingineering and knowledge ingineering (seke2000), chicago, USA, 68 july 2000.

[20] Alf Inge Wang .

« Un support pour les procédés logiciels mobile dans CAGIS ».
In seventh europeen workshop on software process technologie, Kaprun near salzbourg, austria, 22-25 february 2000.

[21] Alf Inge Wang.
« Utilisation d'agents logiciels pour supporter l'évolution des modèles workflow distribués ».
In Proc. International ICSC Symposium on Interactive and Collaborative Computing (ICC'2000), page 7, Wollongong (near Sydney), Australia, December 12-15 2000.

[22] Alf Inge Wang.
« Un environnement centré procédé pour l'ingénierie logiciels coopératives. »
Université norvégienne des sciences et technologies,
Dept. of Computer and Information Science,
NTNU, Trondheim, Norway, February 5th 2001.

[23] Alf Inge Wang.
« Utilisant un environnement basé agent pour supporter les procédés logiciels coopératifs » thèse de doctorat.
Université norvégienne des sciences et technologies,
Dept. of Computer and Information Science,
NTNU, Trondheim, Norway, February 5th 2001.

[24] Michael Wooldridge, Paolo Ciacarini.
« Ingénierie logiciels orientés agents : état de l'art » .
Department of computer science univercity of liverpool.
Dipartimento di scienze de ll' informazione univercity of bologna.italyss

[25] Alf Inge Wang, Reider Conradi et chunnian Liu :
« Architecture multi-agents pour l'ingénierie logiciel coopérative »
In proc. Of the eleventh international conference on software ingineering and knoledge ingineering ? seke 99 page 1-22,kaiserslautern, germany, 17-19 june1999.

[26] Alf Inge Wang, Reidar Conradi , Chunnian Liu.
« L'interaction des fragments de procédés logiciels avec les agents interagissant. »
Université norvégienne des sciences et technologies,
Dept. of Computer and Information Science,
NTNU, Trondheim, Norway, February .

[27] Alf Inge Wang, Anders Aas henssen , Bard Smi Dsrod Nymoen
« Conception principale pour une architecture de système multi-agents mobile pour l'ingénierie logiciel coopérative ».
Université norvégienne des sciences et technologies,
Dept. of Computer and Information Science,
NTNU, Trondheim, Norway.

[28] N. Lardjane, M. Ahmed-Nacer
« Méthode pour intégrer les fragments de modèle de procédés logiciels. »

[29] N. Lardjane,
« Intégration des modèles de procédés logiciels » (chapitre5)
Thèse de magister, USTHB 2002.

[30] N. R. Jennings, P. Faratin, T. J. Norman, P. O'Brien, M. E. Wiegand, C. Voudouris, J. L. Alty, T. Miah, and E. H. Mamdani,
« ADEPT: gérer les procédés business en utilisant les agents intelligents ».
Proc. BCS Expert Systems 96
Conference (Intelligent Systems Integration Programme Track),
Cambridge, UK, 5-23, 1996.

[31] Zakaria Maamar
« l'approche multiperspectives pour la modélisation des workflows »
zakaria.maamar@drev.dnd.ca
Interoperability Group
Information System Technology Section
Defence Research Establishment Valcartier
2459 Pie-XI Blvd North, Val-Bélair, QC G3J 1X5, Canada
&
Computer Science Department and Research Center in Geomatics
Laval University
Ste-Foy, QC G1K 7P4, Canada

[32] Alloui I, Oquendo F.
"*PEACE+: A Multi-Agent System for Computer-Supported Cooperative Work in Software Process Centred Environments",* 8th International Conference on Software Engineering and Knowledge Engineering,10-12/06/96, Nevada, USA, IEEE Computer Society Press.

[33] Daniel K.C. Chan, Karl R.P.H Leung
« Développement de logiciel comme un procédé workflow ».
published in procceding of Joint 1997 Asia Pacific Software engineering.
Conference(APSE 1997).
International Computer science Conférence (ICSC 97). 1997 Hong Kong , SAR, CHINA.

[34] Henri Ramampiano, Alf Inge Wange, Terje Brasethvik.
« Support pour le travail coopérative distribué dans CAGIS »
14 JUNE 2000.

1-Quelques scénarios d'inconsistances dans l' ECP CAGIS :

Dans ce qui va suivre nous allons présenter quelques scénario qui illustreront les problèmes crées par le changement sans restriction des fragments de procédés et la solution adoptée pour palier à ces problèmes.

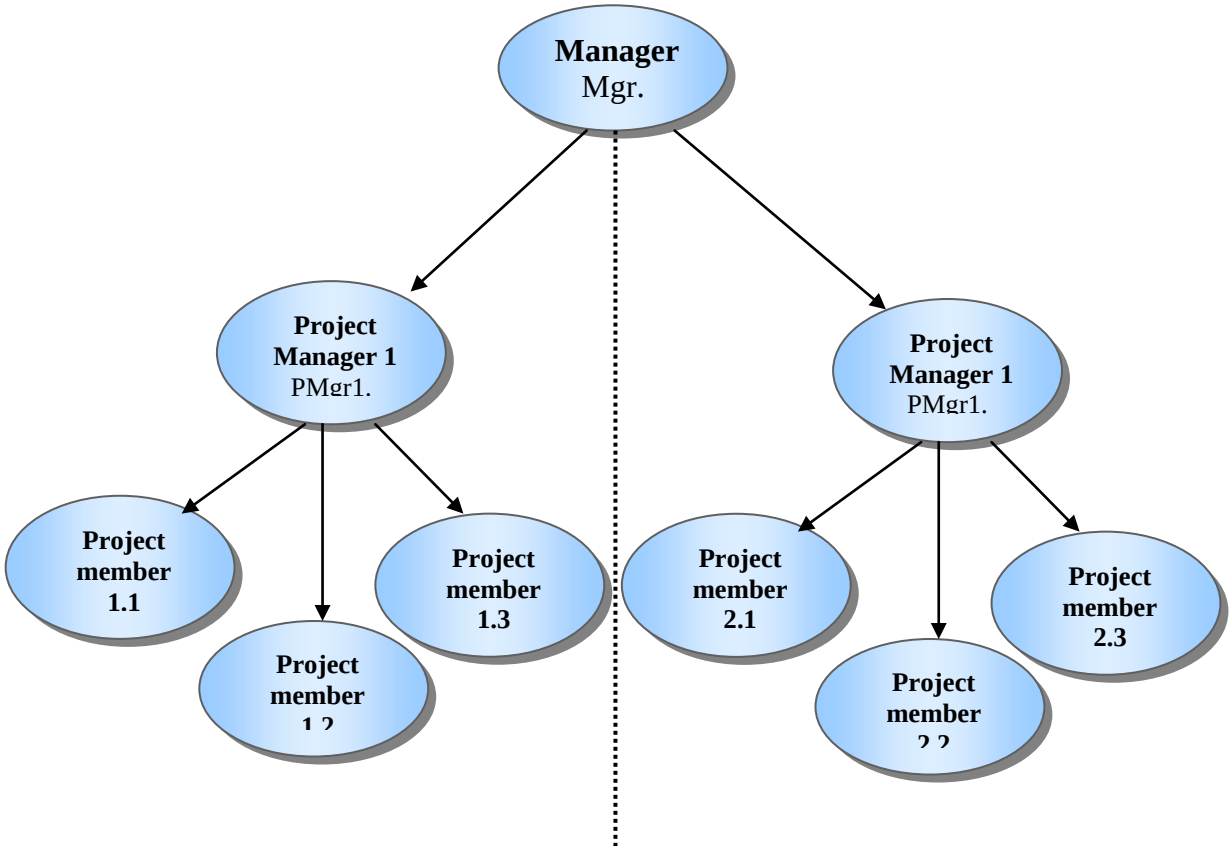


Figure 1 : Hiérarchie organisationnelle utilisée dans les scénarios.

- **Scénario1 : L'anarchie.**

Le gestionnaire de projet veut que tous les membres du projet ait le même fragment de procédé ; de ce faite, il le distribue au manager project1 et manager project2, ces derniers non satisfait du workflow reçue apportent « séparément » des modifications au modèle reçue et envoie les deux nouvelle version aux project member concernés ; ces derniers aussi non satisfait du modèle reçue effectue des modifications personnelles.

Ce cas est un cas extrême ou, on se retrouve avec plusieurs version du même fragment de procédé. Il est difficile de savoir quelle est la version correcte à utiliser.

- **Scénario2 : Mise à jour exclusif dans un seul fragment de procédé.**

Ici le scénario est le même sauf que le fragment de procédé ne peut être modifier que dans un seul espace de travail à la fois, donc si par exemple le manager Project veut effectuer des modifications, il doit avoir la permission du manager.

Le problème qui se pose dans ce cas est que si le fragment de procédé est changé pendant son exécution comment ces changements peuvent être propagés.

- **Scénario3 : mise à jour exclusif dans plusieurs fragments de procédé spécifiés.**

Dans ce scénario les trois membres du projet sont responsables de parties différentes du procédé. Cela veut dire que les fragments de procédé ne vont pas être exécuter en parallèle, mais que chaque membre de projet a des activités qui dépendent des activités d'un autre membre de projet. Le procédé est constitué de trois fragments relié entre eux, ainsi le changement dans un fragment de procédé est permis que pour l'espace de travail qui possède tout le procédé. Dans ce cas la permission est affectée qu'au projet manager 1 et projet manager 2 et le manager.

Le problème rencontré est si le projet manager 1 effectue des modifications et projet manager 1 ne fait rien ; quel est la version la plus efficace, est ce que deux version d'un procédé peut être accepter en exécution.

- **Scénario 4 : Accès exclusif.**

Le principe de ce scénario est qu'un seul espace de travail peut accéder (lire / mise à jour) simultanément à un fragment de procédés.

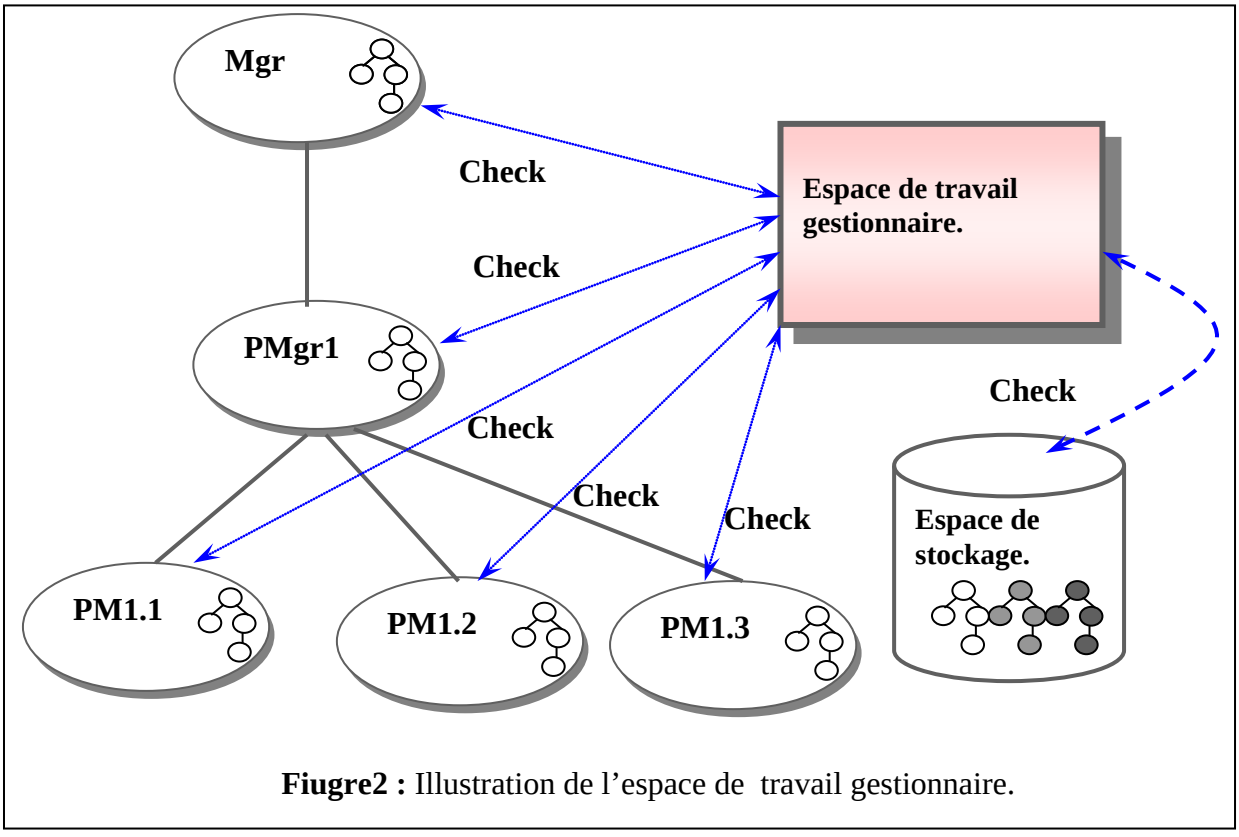
Ce protocole assure la consistance mais aussi met beaucoup de limitations, il n'y a pas de problèmes de propagation car il y a qu'une seule instance qui s'exécute à la fois.

2- Agents prise en charge s'adaptant avec l'évolution Workflow.

Pour palier à ces problèmes de consistances nous avons besoin d'un espace de travail gestionnaire qui sera au courant de toutes de versions des fragments de procédés et les restrictions d'accès.

Les espaces de stockages vont contenir les modèles workflow aussi bien que le modèle de l'organisation des espaces de travail.

Les agents logiciels de plusieurs types vont constituer un support pour la prise en charge des problèmes de consistances.



Fiugre2 : Illustration de l'espace de travail gestionnaire.

- **Support pour le scénario 1 :**

Dans le scénario 1 il n'y a aucune restriction d'accès aux fragments de procédés, n'importe quel espace de travail peut accéder à un fragment de procédé. Si des espaces de travail accèdent en même temps au même procédé en lecture ou un seul en écriture et le reste en lecture rien ne se passe. Si par contre deux espaces de travail veulent accéder au même fragment en écriture, ici il y' a un conflit : l'espace de travail gestionnaire va alors activer deux agents négociants qui vont être affecté aux deux espaces de travail en conflit pour résoudre le conflit qui les concerne.

- **Support pour le scénario 2:**

Le problème posé dans ce scénario est comment propager les changements effectués sur un fragment de procédé. Quant un fragment de procédé est mis à jour, tous les autres espaces de travail qui ont ce fragment de procédé doivent être notifier, l'espace de travail gestionnaire initie l'agent mise à jour, ce dernier va parcourir tous les espaces de travail ayant l'ancienne version du fragment de procédé mis à jour et leur demande est ce qu'ils veulent utiliser la nouvelle version du fragment de procédé. La mise à jour des fragments de procédés peut être de trois types : mise à jour ignorer, aux choix ou forcé.

- **Support pour le scénario 3 :**

Ici nous avons à résoudre le problème de la propagation des changements dans des versions de procédé (constitué de plusieurs fragments de procédés). Le principe adopté est le même que précédemment sauf que les mises à jour effectué sont faites sur tous le procédé version et pas un seul fragment ; si cela doit se faire c'est les agents négociants qui se charge de l'effectuer.

- **Support pour le scénario 4 :**

Dans ce protocole il y a qu'une seule personne à la fois qui peut accéder (lecture ou écriture) à un fragment de procédé. En d'autre terme le fragment de procédé reste verrouillé pendant tout le temps d'accès. Si un espace de travail veut effectuer un check out d'un fragment de procédé ; l'espace de travail gestionnaire envoie l'agent notification pour avoir des informations sur qui à fait un check out du fragment de procédé demandé, combien de temps durera le verrouillage… etc. L'utilisateur peut le questionner sur par exemple le temps restant pour déverrouiller le fragment.

3 - Les langages de modélisation de procédés (PML) utilisés dans l'environnement CAGIS [23]:

Les modèles de procédés CAGIS sont exprimés en 3 différents PMLs :

- Les procédés logiciels individuels modélisés dans les «*procédés simples CAGIS* » est un PML basé réseau exprimé en syntaxe XML.
  Dans ce PML le procédé est représenté comme un ensemble d'activités avec des relations d'ordre, ils sont activés quand l'utilisateur notifie la fin d'une activité.

- Les procédés coopératifs sont dans le « *CAGIS DIAS* » modélisés à travers l'API Java agent. Par contre les agents sont modélisés en JAVA. Ce PML peut être considéré comme un PML basé langages de programmation.

L'API Java agent fournis un nombre de méthodes de haut niveau qui peuvent être utilisés pour effectuer des tâches spécifiques.
KQML est choisi comme un langage de communication entre les agents logiciels car KQML est un standard pour la communication d'agent.

- Le troisième PML dans l'environnement CAGIS est utilisé pour la modélisation du modèle glue du « *Glue serveur de CAGIS* » ; ce langage définis la relation entre les éléments workflow (fragments de procédés) et les agents logiciels.

Le modèle glue peut être considéré comme une partie du modèle du procédé car il trace les événements dans le procédé et notifie les prochaines tâches à effectuer d'après les résultats reçus des interactions (agents- fragments de procédés).
Le modèle glue est modélisé avec un PML basé règles, car il spécifie l'ensemble des règles qui peuvent déclencher certaines actions.

Si nous considérons l'environnement centré procédés CAGIS dans l'ensemble ; nous pouvons déduire qu'il utilise un langage de modélisation de procédés multi-paradigmes. Car pour chaque composant il utilise un PML diffèrent, mais surtout le langage de modélisation de procédés qui convient le mieux.

4 - Scénario d'application pour l'ECP CAGIS:

Pour mieux discerner les différents composants de ECP CAGIS en interaction, nous présentons un petit scénario **[23].**

Le scénario décrit une partie du procédé dans une compagnie de développement de jeu appelé « COOL GAMES », ou nous nous focaliserons sur deux départements : le département de développement logiciel et le département de conception graphique localisé à respectivement Oslo en Norvège et San Francisco en USA.

Non seulement ces deux départements sont distants géographiquement mais aussi ils utilisent des ordinateurs pour interagir via Internet.
La figure 7 montre une partie du procédé logiciel de développement d'un nouveau jeu par les deux départements.

Les développeurs commencent à implémenter un prototype du graphique 3D du jeu (A1), alors que les concepteurs sont entrain de concevoir de nouveaux dessins (B1). Après quelque temps les développeurs et les concepteurs ont décidé d'avoir un entretien (C1), ou ils peuvent s'échanger leurs informations et décider des prochaines actions à effectuer. En se basant sur le résultat de l'entretien le procédé pourra prendre différentes directions : retourner à (A1) et (B1) ou continuer à l'activité (A2)et (B2).

Dans les activités (A2) et (B2) ; chaque département estime la quantité de ressources humaines dont il a besoin (ressources humaines interne au département), et la quantité de ressources humaines dont il a besoin des autres départements. Une négociation est initialisé (C2) entre les deux départements ou ils négocient les quantités de ressources qu'ils auront. Si le procédé de négociation échoue; les deux départements doivent revenir aux activités (A2) et (B2) et revoir leur estimations pour arriver à un arrangement . Après une allocation avec succès, le procédé peut aller aux activités (A3) et (B3), juste après la fin de (A3) le procédé peut continuer et passer à l'activité (A4).

Dans la figure 41 nous pouvons diviser le procédé en 3 parties :

- **S1 : Activités individuelles :** se sont les activités qui peuvent être effectuer par des individus sans interactions avec d'autres personnes des autres départements.

Les activités A1, A2, A3, A4, B1, B2, B3 sont des activités individuelles.

- **S2 : Activités coopératives :** se sont les activités qui peuvent être effectuer que si il y a la participation de plus d'une personne ou département.

C1, C2 sont des activités coopératives.

- **S3 : Les règles de coopération :** c'est la relation entre les activités individuelles et les activités coopératives.

Dans notre exemple ils sont représenté par les lignes entre les activités individuelles et les activités coopératives.

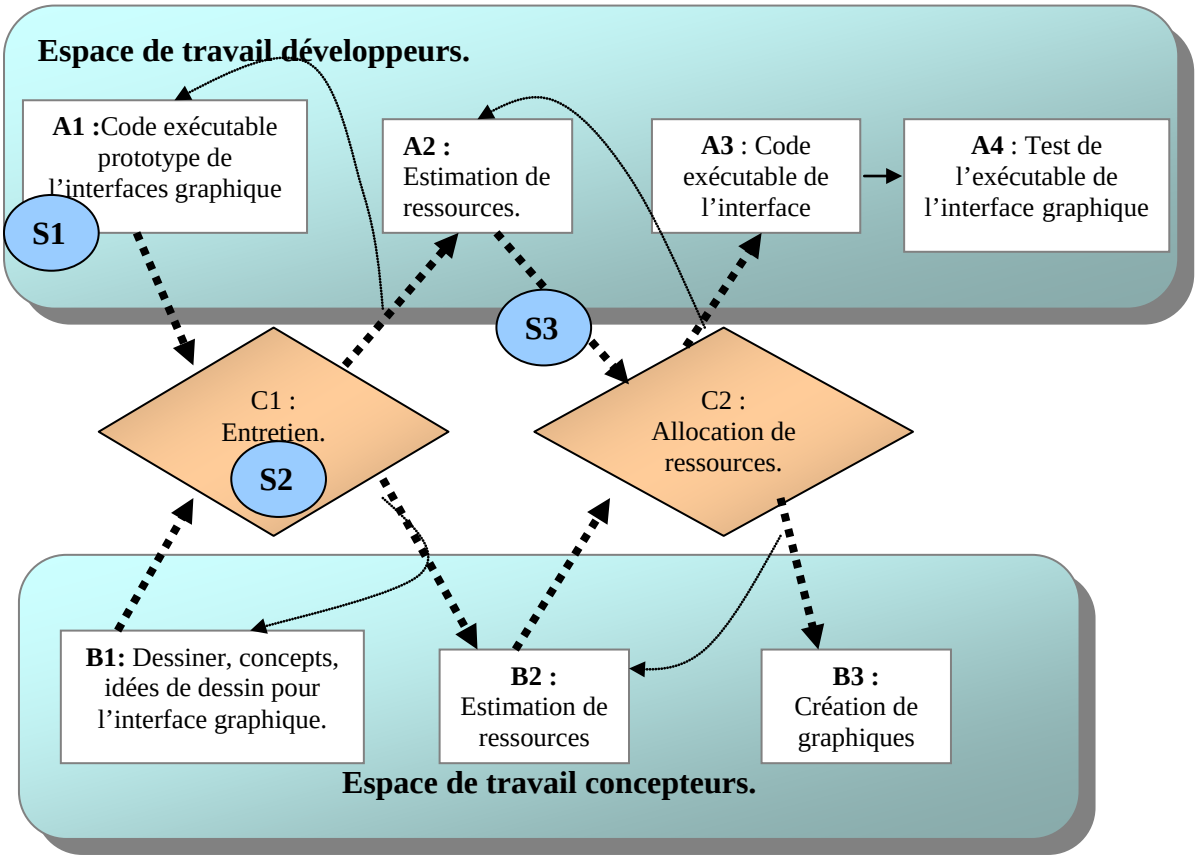


Figure 3 : Le scénario utilisé pour illustrer l'architecture ECP CAGIS. **[22]**

L'architecture CAGIS PSE [22]:

Le fonctionnement (activités) du procédé dans cet exemple se divise en deux catégories :
La coordination et les workflows coopératives.

L'arriver de l'Internet a poussé plusieurs compagnies de développement logiciels à laisser le travaille centralisé traditionnel et immigrer vers le travaille distribué.
Dans les ECPs distribués la communication, coordination, collaboration et négociation entre différents participants deviennent plus compliqué.
Car les personnes peuvent non seulement être distribuer géographiquement mais aussi travailler sur différentes plates-formes, différents horaires et différents modèles de procédés.

Les Systèmes Multi-agents offrent un meilleur moyen pour modéliser et supporter cette distribution, ces systèmes et environnements ouvert.

Chaque composant de ECP CAGIS supporte un type d'activité :

- Pour les activités simples et individuelles, c'est un composant workflow qui les représentent.
  L'architecture du système workflow utilisé est spécialisée dans la modélisation et l'exécution et de procédés workflow simples et répétitifs : *Les simples procédés CAGIS*, cela conviens surtout aux activités locales qui n'ont pas besoin d'interaction avec des activités externes à l'espace de travail.

- Les activités coopératives sont représentées par un système multi-agents (Le CAGIS système d'agents intelligents distribués : CAGIS DIAS), le SMA offre le meilleur moyen pour modéliser et supporter la distribution et hétérogénéité des environnements de travail.
  Les activités C1 et C2 sont des activités coopératives supportées par le *CAGIS DIAS*.

- Les deux composants ne sont pas connecter et n'ont aucun moyen de se connecter ; pour cela nous avons besoin d'un intermédiaire, pour «coller » entre les outils du SMA et les outils workflow ; de ce fait nous pouvons permettre aux agents du SMA et au système workflow d'interagir dans un environnement hétérogène.
  Le CAGIS glue serveur permet au outils workflow des simples procédés CAGIS d'interagir avec les agents du CAGIS DIAS tout en spécifiants les règles de coopération dans le modèle glue.
  Le modèle glue modélise les règles de coopération entre les activités individuelles et les activités coopératives.

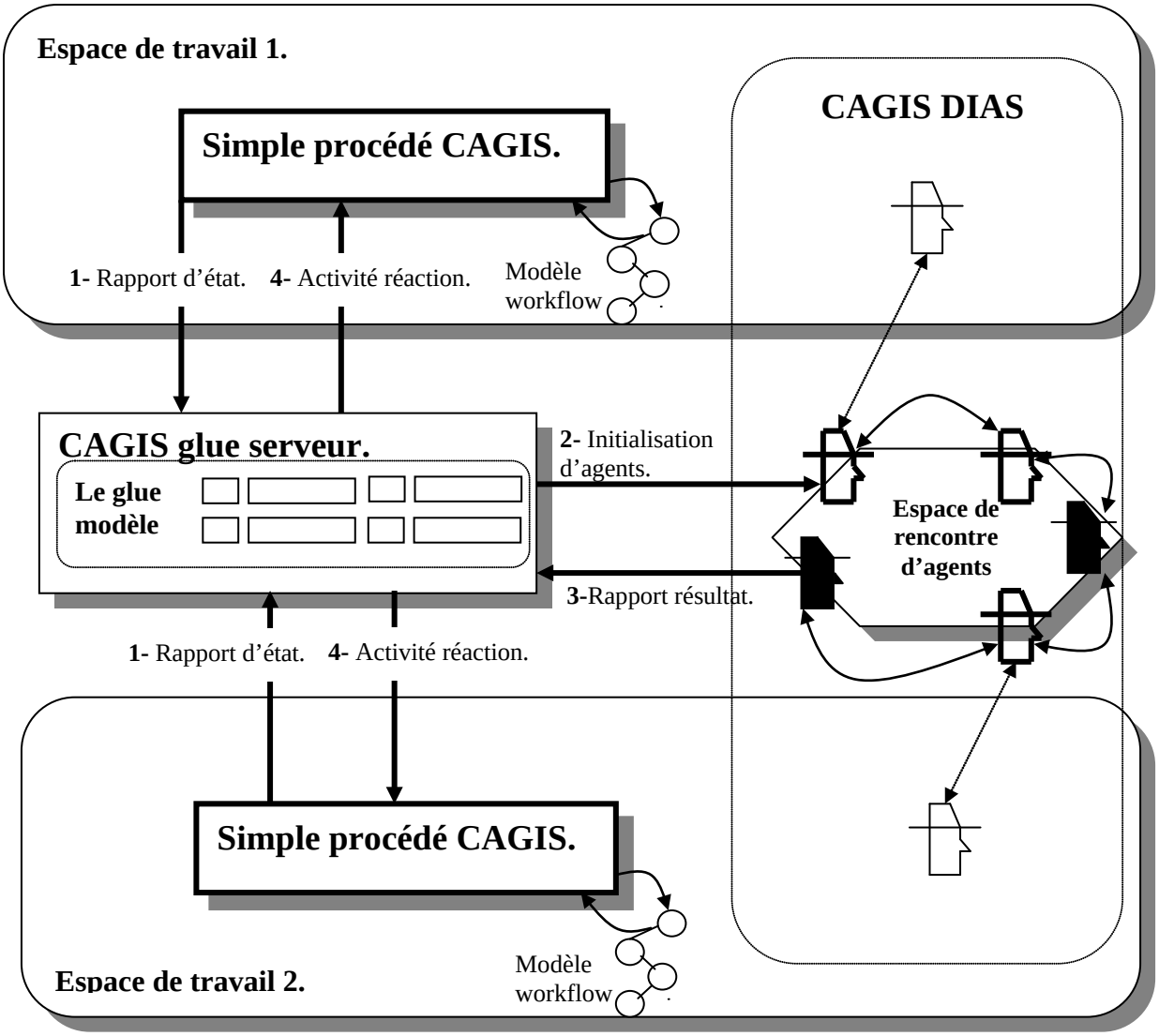


Figure 12 : l'environnement centré procédés CAGIS en interaction.[22]

Les interactions typiques des différents composants sont :

1- Les outils workflow des simples procédés CAGI reporte leurs états au glue serveur CAGIS.
exemple : fin de l'exécution de l'activité A1.

2- le glue serveur va consulter le modèle glue pour voir si quelque chose est spécifier pour les activités A1 et B1 , il initialise des agents adéquat pour le de débat d'idées « C1 » dans le CAGIS DIAS (activités coopérative).

3- l'échange d'idées ou le débat peut retourner deux résultats succès ou échec après la fin de l'activité C1 le résultat est retourné au glue serveur

4- Le modèle glue spécifie différentes réactions selon le résultat retourné au glue serveur.
Selon le résultat retourné le glue serveur va activer une réaction du simple procédé CAGIS ou du DIAS CAGIS ou du glue serveur CAGIS.

Dans le scénario, le résultat positif va activer une réaction du glue serveur CAGIS pour exécuter les activités A2 et B2 du simple procédé CAGIS des deux espaces de travail.

Le résultat négatif va activer une réaction du glue serveur pour reexécuter les activités A1 et B1.

Ces étapes se répète chaque fin d'activités, le glue serveur est consulté pour savoir quel est la prochaine activité à réaliser.

1- L'approche 6- perspectives pour la modélisation des workflows :

Comme son nom l'indique l'approche 6- p se base sur 6 perspectives qui faut prendre en considération pour la modélisation des modèles de procédés ; ces perspectives sont :

Perspective Procédé : Se focalise sur les tâches qui constitue le procédé et la technologie d'exécution de ces tâches. Elle représente *« quoi traiter »* dans le WF.

Perspective rôle : se focalise sur les acteurs organisationnelles qui participent à la tâche de l'exécution .elle représente *« qui est responsable de »* de dans le WF.

Perspective ressources : Se focalise sur les moyens software aussi bien que hardware qui sont utilisé pour exécuter un modèle de procédé. Elle représente *« quoi utiliser »* dans le WF.

Perspective informations : Se focalise sur les données qui sont manipuler dans l'exécution du modèles de procédé. Elle représente *« quoi utiliser côté information »* dans le WF.

Perspective place : Se focalise sur la distribution des contextes d'exécution du modèle de procédé. Elle représente *« ou traiter »* dans le WF.

Perspective ressources : (en interaction avec les autres perspectives) se focalise sur la progression de l'exécution des tâches du modèles de procédés . elle représente *« quand traiter »* dans le WF.

Interaction dans l'approche 6-perspectives :

Dans l'approche 6-p les différentes perspectives sont en interactions continue, ces dernières sont différentes selon les perspectives qui interagissent, ainsi, nous distinguons 9 types d'interactions, ces interactions vont être définit dans ce qui va suivre. Ces interactions ne prnd pas en considérations la concernent pas la perspective temps.

Neuf (09) relations sont définies et résumées dans ce qui suit :

- Assigner (procédé, rôle) : Un procède est assigné aux acteurs qui ont des rôles dans l'organisation.

- *Exécuter (procède, place) :* Un procédé est exécuté dans différentes places de l'organisation, grâce au concept place, l'aspect distributions, et plus particulièrement dans le cas de multi- organisations peut être illustré.

- *Manipuler (procédé, informations) :* Durant son exécution, le procédé manipule différentes informations (d'entré ou sortie) ; des informations en entré pour pouvoir s'exécuter, et produit d'autres informations et documents en sortie. De plus la technologie d'exécution du procédé dépend des valeurs prises par ces informations.

- *Besoin (procédé, ressources) :* Pour s'exécuter le procédé a besoin d'un ensemble de ressources (hardware ou software).

- *Situer (rôles - places) :* Un rôle est situer dans une place de l'organisation.

- *Gérer (ressources- informations) :* La ressource gère différentes informations qui sont disponibles dans différent format.

- *Localiser (ressources – place) :* La ressource est localiser dans une des places de l'organisation.

- *Autoriser (rôle, informations) :* Un rôle est caractérisé par des informations, ou il a l'autorisation d'accéder en lecture ou écriture ou les deux.

- *Utiliser (rôle, ressources) :* Pour effectuer ses tâches, le rôle utilise différentes ressources.

La perspectives temps représente un frame-work cadre de travail » qui rassembles les cinq autre perspective, pour cela, la perspective temps demande périodiquement leurs états internes tel que la progressions des tâches dans la perspective procédé, les ressources occupées de la perspective ressource, les acteurs exige de la perspectives rôle …etc.

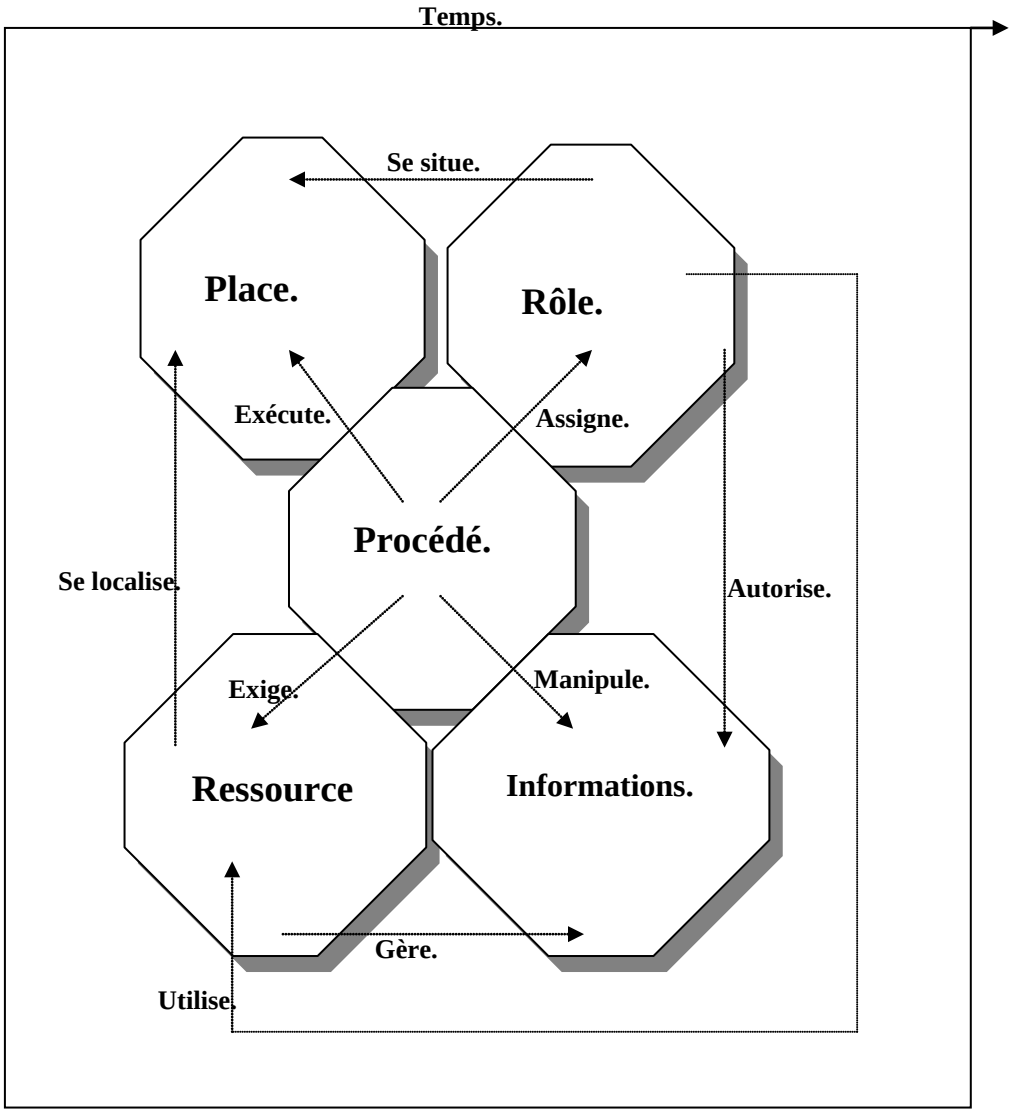


Figure 1 : La représentation de l'approche 6 perspectives.

Dans l'approche6-p les 6 perspectives sont représentées par des composants intelligents : les agents logiciels, et cela pour la gestion et l'interactions des perspectives : ainsi les 6 types d'agent sont exiger, ces agents sont :
Agent procédé ; agent rôle; agent temps ; agent ressources ; agent place ; agent- informations ; chaque agent est associé à une perspective.

L'agent procédé : Est considéré comme le moteur du workflow ; il est responsable du procédé et de la manipulation des tâches ; ainsi cet agent correspond avec tous les agents reliés au procédé en terme d'initiation ;assignation ;adaptation du guidage d'évaluation.

L'agent rôle : Agit a la place de l'acteur qui a un rôle dans l'organisation. Cet agent gère les détails qui concerne l'acteur en terme de validité ; expérience et back- grounds.

L'agent informations : Est responsable de la gestion des informations ; ainsi de toute les requêtes de L'utilisation aussi bien que la MA ;les informations doivent passer par cet agent .
de plus l'agent information gère les conflits des informations qui viennent avec les opération.

L'agent ressources : Agit comme un proxy pour les ressources ; et gère leur périodes d'utilisation.
De plus cet agent supporte le bon fonctionnement de la ressource ;en s'occupant de la maintenance en cas de besoin par exemple.

L'agent place : Identifie la place et son antenne ; en terme de rôles ;ressource et information ;de plus cet agent retrouve des informations sur la position des places tel que la floor number ;avec le respect des autre place d'organisation .

L'agent temps : Est en charge de l'invocation des autres perspectives a travers leurs agents perspectives. La bonne tenue(but) de cet invocation est de connaître la progression des WFs .

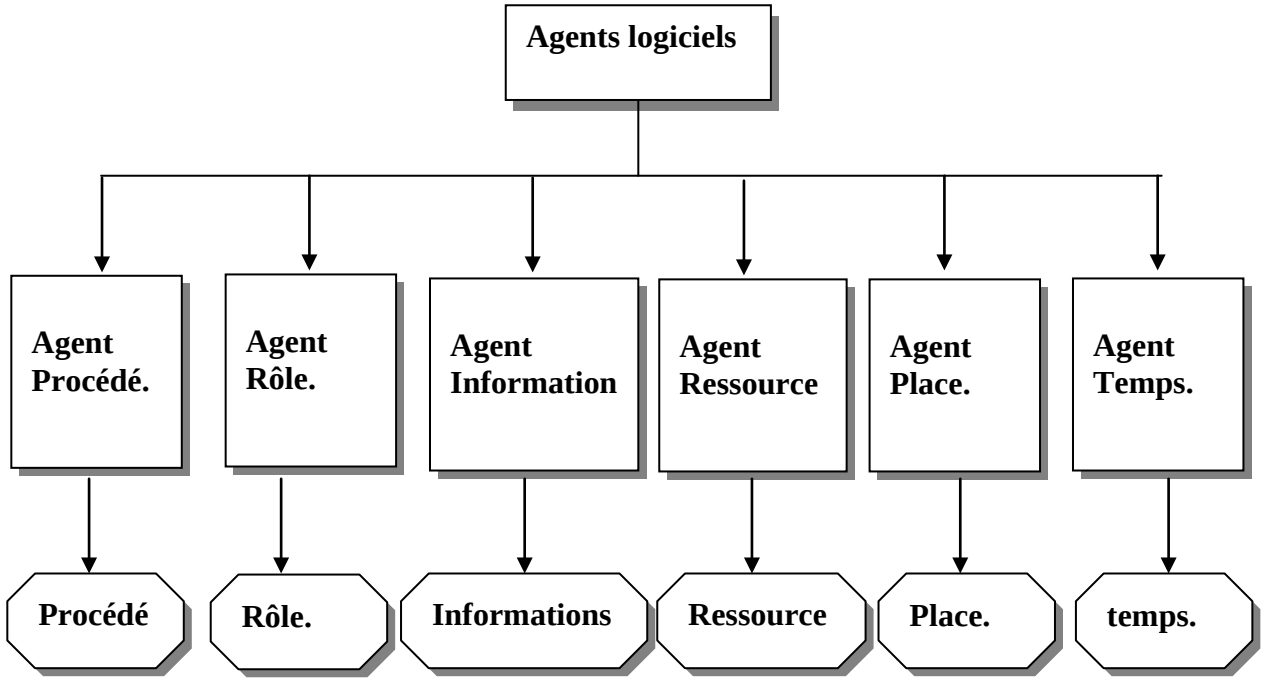


Figure 2 : Les agents logiciels dans l'approche 6-P.

1- Gestionnaire de procédés distribués et dynamiques basés technologie d'agents :

Les environnements d'ingénierie logiciels doivent souvent gérer, exécuter des modèles de procédés très complexes. Aussi, la plus part du temps, ces modèles sont distribués et dynamiques car on ne peut pas les planifier à l'avance et peuvent changer pendant l'exécution.

Dans ce qui suit nous présentons un gestionnaire de procédés basé technologie d'agents.

L'agent ayant des propriétés d'autonomie, sociabilité et réactivité convient très bien pour gérer des procédés logiciels distribués et dynamiques.

Le concept de base du gestionnaire de procédés est que :

La conception du procédé business (affaire), sa formalisation à des spécifications de l'environnement et l'exécution de ces spécifications sont totalement indépendant. Pour respecter cette indépendance le gestionnaire de procédés doit pouvoir répondre aux exigences de la modélisation et de l'exécution du modèle de procédés et respecter leurs indépendances, ces exigences sont comme suit :

La modélisation :

1- Un support pour différents scénarios de modélisation de procédés centralisés ou distribués et la participation à la modélisation doit être supporté.
   La capacité de la modélisation distribuée et l'intégration des modèles de procédé doit être assurée pour qu'elle couvre le plus grand nombre de procédé logiciels.
2- La représentation des procédés : la description du procédé doit être adaptable au besoin personnel des projets, en d'autre terme la présentation doit être indépendante du «Comment doit être fait », ce dernier peut être décidé et adapté selon l'espace ou le groupe de travail, ce qui rend le modèles de procédé très flexible.

L'exécution :

1- L'évolution de procédés : Le gestionnaire de procédé doit fournir des mécanismes pour la planification de modifications et doit prendre en charge les modifications à temps réel de l'instance d'exécution du procédé et la gestion de scénarios d'exceptions.
2- Support transactionnel : Le gestionnaire de procédé doit fournir de protocoles transactionnels, adaptables pour coordonner l'exécution concurrente des tâches manuelles ou automatiques et pour supporter l'automatisation des étapes du procédé.
3- Intégration des versions et possibilité de contrôle des espaces de travail : Le versionnement, la propagation des données, l'intégration et la cohérence des données et le travail coopérative doit être supporté de manière efficace.

2- La construction des blocs du framework :

Le framework de la modélisation et exécution des procédés est composé de 4 espaces : espaces organisationnel, espaces ressources, espaces dépôt, ces espaces couvrent respectivement les aspects organisationnels, ressources, données et informations, et l'espace procédés couvre les perspectives fonctionnelles et comportementales. Tous ces aspects sont nécessaires pour la modélisation et l'exécution d modèles de procédés et les interactions entre ces espaces ou dans l'espace lui-même sont importantes ; elles doivent être représentées et gérées.

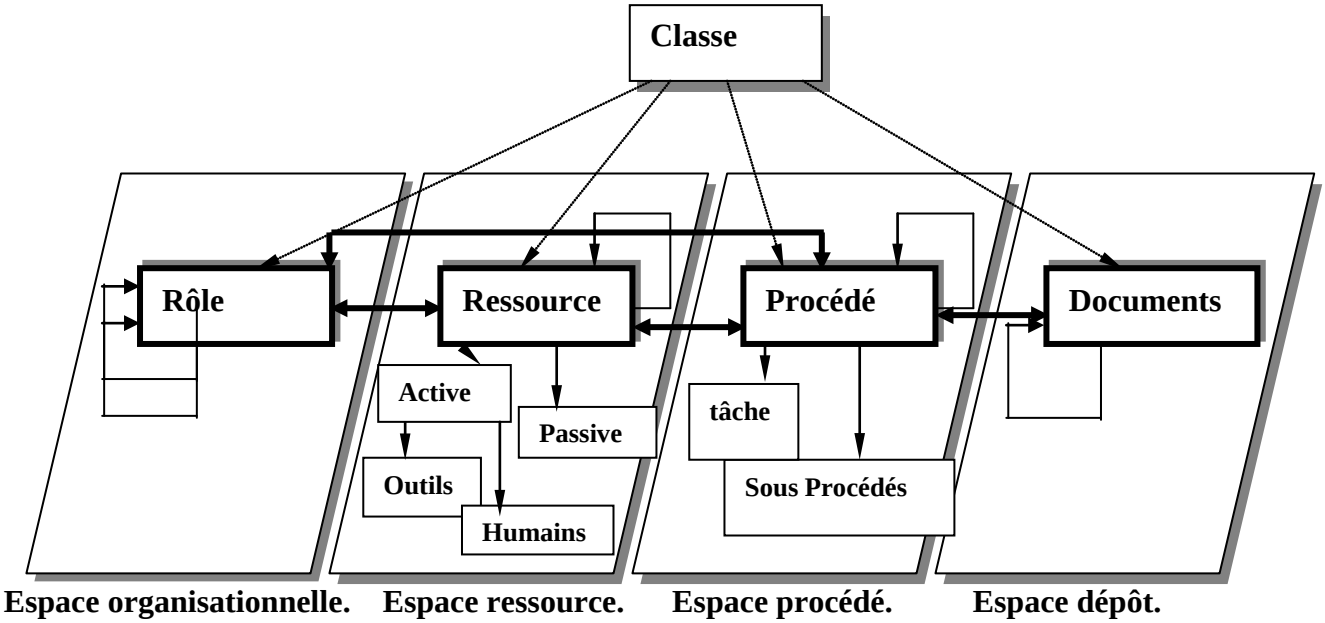


Figure 1 : La modélisation des espaces du méta modèle du procédé.

3 - Les types d'agents utilisés:

Dans ce gestionnaire de procédés nous distinguons des types d'agents pour :

La gestion du procédé, la gestion des ressources, la gestion des espaces de stockages. L'espace organisationnel n'est pas directement représenté par un type d'agent mais il impose certaines conditions sur l'architecture des agents et leurs communications.

- **Agent procédé :**

L'agent procédé agit comme un moteur d'exécution de procédé distribué. L'agent procédé interagit avec d'autres agents procédés et d'autres types d'agents pour effectuer le procédés dont il est responnsable. L'agent procédé gère le flux de travail car il coordonne entres les différentes activités complexes ou atomiques. L'agent procédé peut créer de nouveaux sous agents procédés et leur délègue des sous procédés dont il est responsable. On distingue deux types d'agents l'agent tâche qui est responsable des tâches atomiques et l'agent procédé qui est responsable des tâches complexes ou du procédé.

Agent tâche :

Il délivre les informations contextuelles dont on a besoin pour exécuter une tâche et informe les agents acteurs de tous changements tel que suspension ou report de tâche ,changement d'entré …

Il est responsable de l'allocation et le scheduling des ressources.

Les agents acteurs agissent à la place de l'être humain ou de la machine et effectue la tâche demandée, il rapporte le résultat et l'état des performances du procédé et particulièrement toutes déviation ou exception du modèle de procédé.

Les agents tâches et procédés doivent surtout réagir dans des situations de changements dynamiques ainsi, ils doivent fournir des mécanismes de modifications rapides du modèle de procédé en exécution spécialement pour les dernières mises aux points et arrangements du procédé.

L'architecture des agents procédé est flexible et adaptable aux différents espaces de travail et groupements organisationnels ce qui facilite l'extension et l'utilisation de ces agents avec des environnements hétérogènes sans trop de difficulté.

- **Agent espace de stockages ou « dépôt » :**

Les agents stockage agissent comme des gestionnaires de documents car ils fournissent des mécanismes pour le stockage uniforme des informations aussi le contrôle d'accès et accès concurrents pour tous les documents.
Ces agents sont responsables de la gestion des versions et le maintien de l'intégrité et la consistance des informations ;
Ils sont responsables de la propagation des changements et fournissent des fonctionnalités fondamentales de modélisation des procédés.

- **Agent ressource :**

Comme son nom l'indique, l'agent ressource s'occupe de la gestion des ressources utilisées par le procédé.
Les ressources sont soit passives tel que le temps ou actives tel que les outils ou les humains ;
L'agent acteur s'occupe des ressources actives de ce fait il y a deux types d'agent acteurs :

Les agents personnels :Pour assister l'être humain dans son travail.

Les agents outils :S'occupent de l'intégration des systèmes d'applications, particulièrement les applications autonomes et hétérogènes.

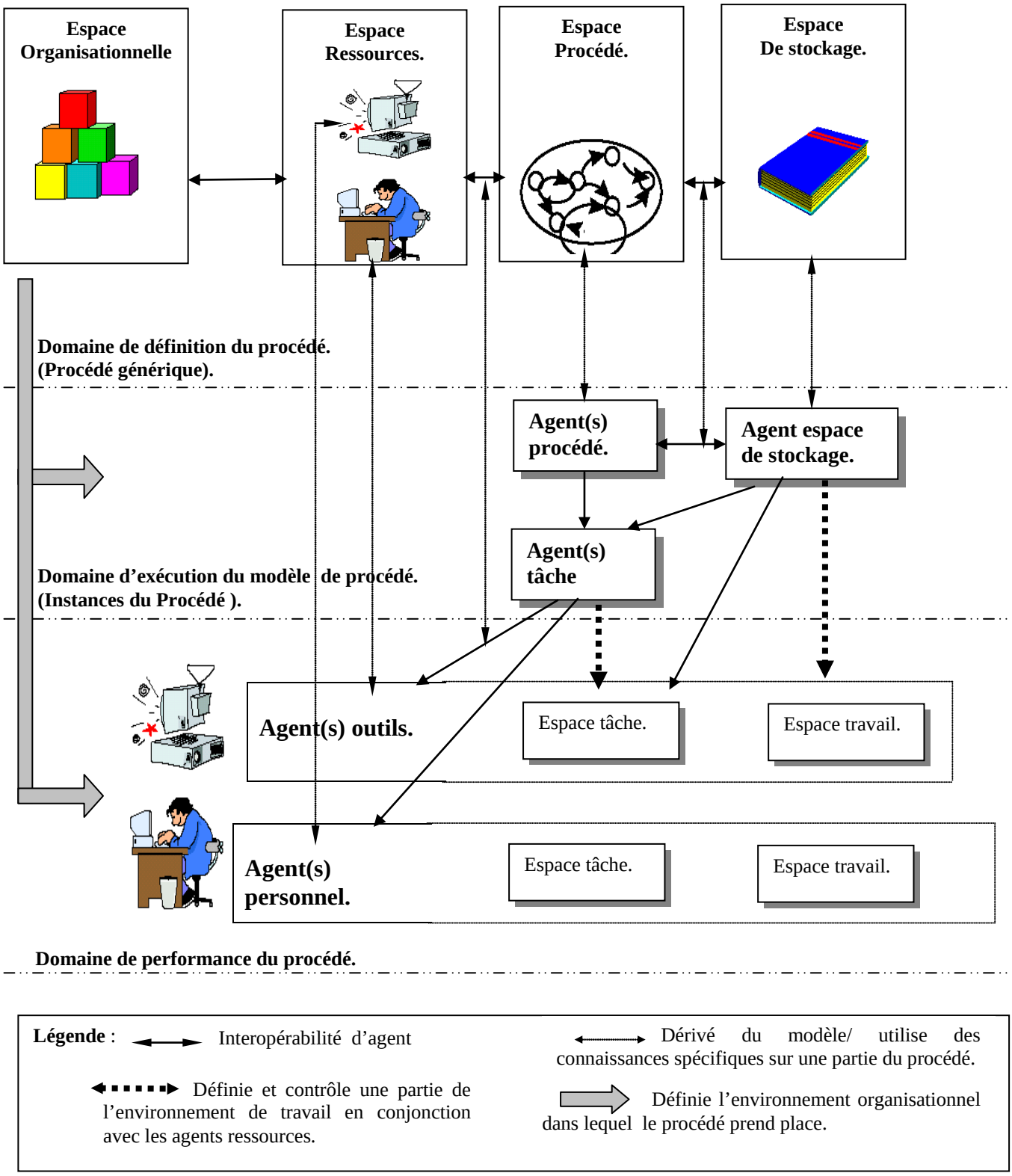


Figure 2 : la modélisation et l'exécution des procédés basés agents.