REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE SAAD DAHLAB, BLIDA

FACULTE DES SCIENCES
DEPARTEMENT INFORMATIQUE

Mémoire en vue de l'obtention du diplôme de Master II

Spécialité : Génie des Systèmes Informatiques

Thème :

Mise en œuvre des systèmes cyber-physiques à base de système multi-agents en intégrant le système de développement Android.

Présenté par :
M^lle^ BELKACEM Meriem
M^lle^ LEMOU Houda Amet Allah

Promoteur :
Mr KAMECHE Abdallah Hicham

Encadré par :
Mr GAHAM Mehdi

Devant le jury composé de :
Président : Mr N.Chikhi
Examinatrice : Mme H.Ghribi

Date soutenance: 21/06/2016

Promotion 2015/2016

Remerciement

Avant de présenter ce travail, nous tenons à remercier Allah le tout puissant pour nous avoir donné beaucoup de patience et de courage pour réaliser ce travail.

Nous tenons à présenter nos sincères remerciements et notre profonde reconnaissance à notre encadreur Monsieur Gaham Mehdi pour le sujet qu'il nous a proposé. Merci d'avoir accepté de suivre la réalisation du travail, pour les conseils et l'encouragement.

A Monsieur Mihoubi Bachir, Qu'il veuille bien trouver ici l'expression de notre reconnaissance pour sa patience, sa disponibilité, ses conseils et son aide, qui nous ont permis de réaliser ce travail dans les meilleures conditions.

Nous remercions très sincèrement Monsieur Kameche Abdallah Hicham, Pour son aide précieuse, pour ces conseils et son soutien qui nous avons permis de mette en valeur toute notre connaissance.

Notre respect s'adresse aux membres du jury qui nous feront l'honneur d'apprécier ce travail.

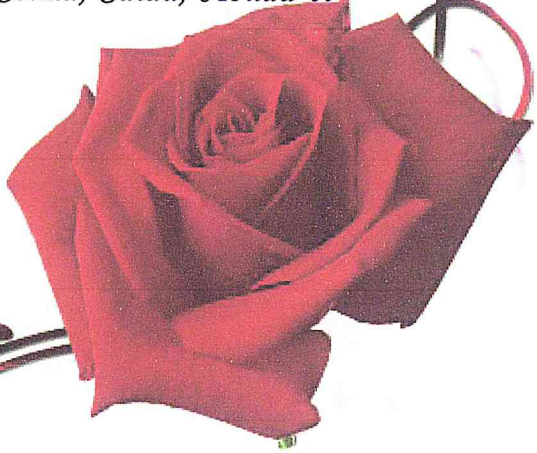
Nous tenons à présenter tout notre respect à tous les enseignants qui ont contribué à notre formation du primaire jusqu'au cycle universitaire.

Dédicaces

Je dédie au fond de cœur cette ouvrage mémoire tout d'abord à ma famille et surtout mes très chères parents pour ses encouragements, que dieu les protèges.

Je le dédie à mes sœurs : Khadidja et Fatma Zohra, et Mon frère : Ahmed.

Sans oublier à ma chère amie et binôme « Houda », à tous mes collègues de notre section, et sans oublié mes très chers ami(e)s : Oussama, Maroine, Khadidja, Zahra, Selma, Saïda, Houda et Amina.



MERIEM

Dédicaces

Je dédie ce travail à mon très cher père ***Sidali*** *: rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être, ce travail est le fruit de tes sacrifices. A ma chère mère* ***Zineb*** *: tu représentes pour moi la source de tendresse et l'exemple de dévouement qui n'a cessé de m'encourager et de prier pour moi, aucun dédicace ne saurait être assez éloquent pour exprimer ce que tu mérites pour tous ce que tu me donnes depuis ma naissance.*

A mon futur mari ***Mehdi*** *: quand je t'ai connu, j'ai trouvé l'homme de ma vie et la lumière de mon chemin, ton soutien moral et matériel, m'a permis de réussir mes études.*

A mes sœurs : Remaissa, Sara, Dhaouia, Ikram, Malak, Zenira et Imene, à mes frères : Abderahmane et Abderezzak.

A ma très chère nièce Rawan.

Sans oublier ma chère amie et binôme « Meriem », à tous mes collègues de notre section, et sans oublié mes très chers ami(e)s : Houda, Amina, Ahlem, Zahra, Saida, Serine, Assia, Samra, Djedjiga, Zineb, Widad et Soumia.

HOUDA

ملخّص

هذا العمل هو جزء من استمرارية العمل في إنتاج فريق النظم الروبوتية، ووحدة التصنيع الروبوتات التابعة لمركز التنمية التكنولوجيات المتطورة CDTA في إطار تطوير مشروع اختبار منصة متعددة الوكلاء لإنتاج أنظمة التحكم الإلكترونية المادية.

الهدف من العمل في هذه الأطروحة هو تنفيذ الأنظمة الإلكترونية المادية على أساس نظام متعدد الوكلاء ودمج أنظمة تطوير الأندرويد.

العمل الذي قمنا به هو تنفيذ بنية متعدد الوكلاء والبروتوكولات والنهج التنسيقي للسيطرة على خلية إنتاج مرنة. ودمج "الأجهزة في حلقة المحاكاة" باستخدام هاتف يعمل بنظام أندرويد.

الكلمات الرئيسية: أنظمة الإلكترونية المادية، نظام متعدد الوكلاء، منصة JADE، برمجة الأندرويد.

Résumé

Ce travail s'insère dans la continuité des travaux réalisés au sein de l'équipe Systèmes Robotisés de Production, de la division robotique et productique du Centre de Développement des Technologies Avancées CDTA dans le cadre du projet développement et expérimentation d'une plateforme multi-agents pour le pilotage des systèmes de production cyber-physiques.

L'objectif de travail dans ce mémoire, est la mise en œuvre des systèmes cyber-physiques à base de système multi-agents en intégrant le système de développement Android.

Notre travail consiste à implémenter d'une architecture multi-agent (JADE), des protocoles et approches de coordination pour le pilotage d'une cellule de production flexible. Ainsi que l'intégration d'une approche "Hardware in the Loop Simulation (HILS)" à l'aide d'un système Android.

Mots clefs : les systèmes cyber-physiques, systèmes multi-agent, plateforme JADE, programmation Android.

Abstract

This work is part of the continuity of the work at the team Production Robotic systems, robotics and Manufacturing division of CDTA in the project development and testing of a multi-agent platform for control systems cyber-physical production.

The objective of work in this thesis is the implementation of cyber-physical systems based on multi-agent system in integrating Android development system.

Our work is to implement a multi-agent architecture (JADE), protocols and coordinating approaches to controlling a flexible production cell. Moreover, the integration of an approach "Hardware in the Loop Simulation (HILS)" using an Android system.

Key words: cyber-physical systems, multi-agent systems, JADE platform, Android programming.

Sommaire

Introduction Générale 1

Chapitre I : 3

Un système cyber-physique 3

1. Introduction 4

2. Définition d'un système cyber-physique 4

3. Les caractéristiques d'un système cyber-physique 5

3.1. Caractère électronique 5

3.2. Caractère Métrologie 5

3.3. Caractère Informatique 6

3.4. Caractère Automates 6

3.5. Caractère Fusion de données 6

3.6. Caractère Mécanique 6

4. Domaines d'application d'un système cyber-physique 7

5. Le défi de la conception cyber-physique 7

6. Les systèmes Cyber-physiques et le concept Industrie 4.0 8

7. Conclusion 10

Chapitre II : 11

Un système multi-agents et la simulation des systèmes de production 11

1. Introduction 12

2. Le concept d'agent 12

2.1. Définition d'un agent 12

2.2. Les caractéristiques d'un agent 13

2.3. Les types d'agents 13

2.3.1. Agent cognitif 13

2.3.2. Agent réactif 14

2.3.3. Agent hybride 14

3. Un système multi-agents 14
3.1. Définition d'un système multi-agents 14
3.2. Les caractéristiques d'un système multi-agents 15
3.3. Domaines d'application des systèmes multi-agents 15
3.4. Interaction entre agents et protocoles 15
4. Plateformes de développement des systèmes multi-agents 16
5. La plateforme JADE 17
5.1. Définition de JADE 17
5.2. La norme FIPA pour les systèmes multi-agents 17
5.3. Les outils de débogage de JADE 19
5.3.1. Agent RMA (Remote Management Agent) 19
5.3.2. Agent Dummy 20
5.3.3. Agent Directory Facilitator 21
5.3.4. Agent Sniffer 22
5.3.5. Agent Inspector 23
6. La simulation des systèmes de production : 23
6.1. La simulation HIL (Hardware in the Loop) 23
6.2. La simulation à événements discrets et FlexSim 24
6.2.1. La simulation à événements discrets 24
6.2.2. Le logiciel FlexSim 25
7. Conclusion 27
Chapitre III : 28
Description et spécification du système 28
1. Introduction 29
2. Problématique et description du système 29
3. Couche de simulation HIL 30
3.1. La partie simulation 30

3.2. La partie matérielle 32
4. Couche de pilotage MA 33
5. Spécification du système 35
5.1. Conception orientée agent 35
5.1.1. Diagrammes de séquence 35
5.1.2. Diagramme de classe 38
6. Conclusion 46
Chapitre IV : 47
Réalisation et validation du système 47
1. Introduction 48
2. Environnement de développement 48
2.1. NetBeans 48
2.2. Intégration de JADE dans NetBeans 48
2.3. Android Studio 49
2.4. FlexSim 49
3. Le protocole de communication 51
3.1. Le protocole TCP/IP 51
3.2. Les sockets 52
4. Le système de simulation et le système de pilotage 52
4.1. L'implémentation des règles de répartition 55
5. Le système de pilotage 60
5.1. Implémentation de socket dans un agent 60
5.2. Le fonctionnement Multi-Agents 62
6. Implémentation du produit cyber-physiques 64
6.1. Android 64
7. Validation de la simulation Hardware In The Loop 65
8. Conclusion 65

Conclusion Générale..... 66

Bibliographie 68

Liste des Figures

Figure 1 : La composition d'un système cyber-physique. 4
Figure 2 : Les éléments d'un système cyber-physique. 5
Figure 3 : Les caractéristiques d'un système cyber-physique. 6
Figure 4 : De l'Industrie 1.0 à l'Industrie 4.0. 9
Figure 5 : Un Agent avec son environnement. 13
Figure 6 : Le modèle de référence pour une plate-forme multi-agents FIPA. 18
Figure 7 : L'interface de l'agent RMA. 19
Figure 8 : L'interface de l'agent Dummy. 20
Figure 9 : L'interface de l'agent DF. 21
Figure 10 : L'interface de l'agent Sniffer. 22
Figure 11 : L'interface de l'agent Inspector. 23
Figure 12 : Système de production sous FlexSim. 26
Figure 13 : Représentation générale du système. 29
Figure 14 : Schéma global qui détaille la partie simulation. 30
Figure 15 : Emplacement des machines de la cellule AIP-PRIMECA. 31
Figure 16 : la cellule AIP-PRIMECA en réalité à l'université de Valenciennes (France). . 31
Figure 17 : Schéma global qui détaille la partie matérielle. 33
Figure 18 : Schéma global qui montre les agents du système de la couche de pilotage MA. 34
Figure 19 : Diagramme de séquence « choisir machine ». 35
Figure 20 : Diagramme de séquence "enregistrement". 36
Figure 21 : Diagramme de séquence "routage". 37
Figure 22 : La classe Agent Lanceur Machine et Routage. 38
Figure 23 : La classe Agent Lanceur Produit. 38
Figure 24 : La classe Agent Produit. 40
Figure 25 : La classe Agent Machine. 42
Figure 26 : La classe Agent Routage. 43
Figure 27 : La classe Données. 44
Figure 28 : La classe Regles. 45
Figure 29 : L'accès aux propriétés de l'objet. 50
Figure 30 : Modification du code de la fonction OnEntry via l'onglet Triggers. 50
Figure 31 : Principe global de la communication entre les deux Systèmes. 51

Figure 32 : Interaction entre les deux systèmes..... 53
Figure 33 : L'utilisateur donne le nombre de produit. 54
Figure 34 : Le message reçu du FlexSim contenant la quantité des produits..... 56
Figure 35 : Le choix des règles produits. 57
Figure 36 : La file d'attente..... 58
Figure 37 : L'échange des messages en utilisant l'agent Sniffer..... 59
Figure 38: La création de l'agent dans un split Container..... 65

Liste des Tableaux

Tableau 1 : Tableau des différents temps d'opérations(en seconde). 32
Tableau 3 : Assignations des entités s à leurs agents. 54
Tableau 4 : Table des règles de répartition de machines. 55
Tableau 5 : Table des règles de répartition de produits. 55

Introduction Générale

Les systèmes cyber physiques sont une nouvelle génération de systèmes distribués intégrant la décision et le calcul avec une orientation vers l'interaction avec l'être humain. La mise en place de ces systèmes nécessite l'utilisation de nouvelles technologies tels que les plateformes multi-agent et les plateformes de calcul embarqué ainsi que la mise en place d'interfaces électroniques avec le monde environnant.

Intensivement exploré ces dernières années, le paradigme multi-agent issu des recherches sur l'intelligence artificielle distribuée, permet le pilotage intelligent des systèmes de production complexes tel que les systèmes cyber-physiques.

Dans ces systèmes, les machines ne sont plus pilotées à partir d'une centralisation mais seront configurées et pilotées par les produits qui se présentent à elles. Cette approche est basée sur une association en continue du produit physique à son image informationnelle dans le système informatique de pilotage et sur l'introduction du produit en tant qu'agent actif voir même intelligent.

Ces produits communicants en l'occurrence des ébauches de pièces ou de matière première vont dialoguer avec les machines et leur indiquer quelle transformation, traitement ou test ils doivent subir, chacun de ces produits emporte avec lui les informations concernant son état.

Les informations peuvent être récupérées à partir de capteurs sans fil, qui peuvent à leur tour être intégrés dans une boucle de simulation, d'où l'approche "Hardware-in-the-Loop-simulation (HILS)", déjà utilisée depuis de nombreuses années pour le développement de systèmes de commande contrôle électroniques, s'est développée graduellement ces dernières années dans le cadre de la conception, de l'expérimentation et de la vérification des systèmes de production automatisés.

Dans ce contexte, notre travail est la mise en œuvre des systèmes cyber-physiques à base de systèmes multi-agents en intégrant le système de développement Android, pour la connexion aux ressources de production. Le travail présenté dans ce mémoire a été élaboré au sein du laboratoire Systèmes Robotisés de Production du Centre de Développement des Technologies Avancées (CDTA).

L'intérêt étant principalement l'implémentation d'une architecture multi-agent (JADE) et des protocoles et approches de coordination en développant un ordonnancement par rapport aux règles de priorité pour les machines et aux règles d'affectation pour les produits pour le pilotage d'une cellule de production flexible, ainsi que le déploiement du système sur une architecture Android, pour la connexion aux ressources de production.

L'organisation du mémoire est comme suit :

Dans le premier chapitre nous présentons la définition, les caractéristiques, les domaines d'applications, le défi de conception et le concept industrie 4.0 liés au système cyber-physiques.

Dans le deuxième chapitre, nous présentons les éléments de base du travail accompli : le concept d'agent, le système multi-agents, la plateforme de développement multi-agent JADE, la simulation "hardware in the loop", la simulation à évènements discret et FlexSim.

Dans le troisième chapitre, nous donnons le problématique et la description du système de production, une couche de simulation Hardware in the Loop (HIL), une couche de pilotage Multi-Agents (MA) et spécification du système.

Dans le quatrième chapitre, nous exposons d'une part l'implémentation du système à l'aide d'une plateforme JADE, et d'autre part l'intégration de l'approche "Hardware in the Loop Simulation" dans le système à l'aide un téléphone Android.

Nous conclurons par une discussion des résultats. Enfin, nous présenterons des perspectives par rapport au système réalisé.

Chapitre I :

Un système cyber-physique

1. Introduction

De nombreux systèmes complexes conjuguent les domaines de l'informatique, des technologies de l'information et de la physique. Il est fréquent de faire référence aux systèmes informatiques, de technologies de l'information et de communication collectivement en utilisant le terme "cyber-systèmes" ; ces derniers interagissent directement avec le monde qui nous entoure de manière nouvelle et bénéfique. L'étude et le développement de systèmes techniques regroupant les mondes cybernétique et physique est un domaine scientifique émergent. Ces systèmes cyber-physiques sont conçus pour interagir continuellement et de manière dynamique avec leur environnement grâce à l'association d'éléments du monde physique et d'éléments du monde de l'information distribués.

Dans ce chapitre, nous présentons un système cyber-physique et ses caractéristiques et ses domaines d'applications. À la fin, nous verrons la conception d'un système cyber-physique et le concept Industrie 4.0.

2. Définition d'un système cyber-physique

Les systèmes cyber-physiques sont des systèmes embarqués intelligents, composés d'électronique et de logiciel, reliés au monde réel au travers de capteurs et d'acteurs, et connectés entre eux et à internet comme l'illustre la Figure 1. Le monde physique fusionne ainsi avec un monde virtuel, pour créer un cyberespace, qui selon sa définition est un ensemble de données numérisées, constituant un univers d'information et de communication, lié à l'internet [1].

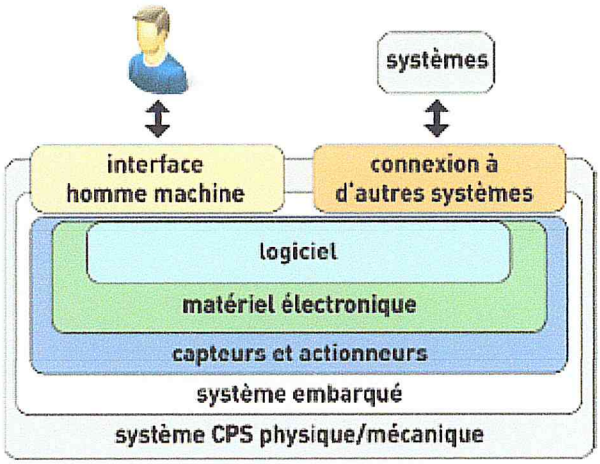


Figure 1 : La composition d'un système cyber-physique.

Un système cyber-physique se caractérise par trois comportements fondamentaux qui s'entrecroisent : le calcul informatique, la communication et le contrôle comme l'illustre la Figure 2 [2].

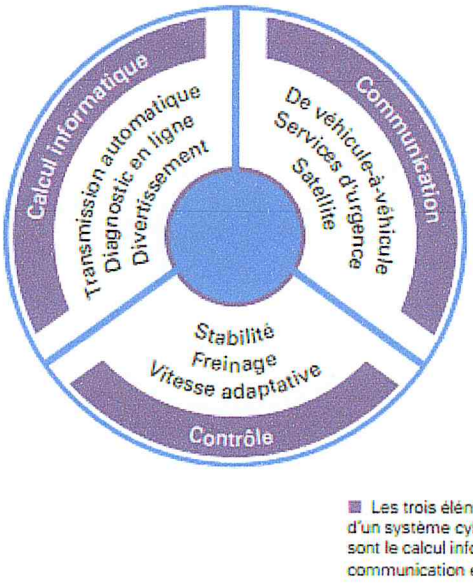


Figure 2 : Les éléments d'un système cyber-physique.

3. Les caractéristiques d'un système cyber-physique

Les caractères d'un système cyber-physique sont [3]:

3.1.Caractère électronique

Comme l'intelligence interne, les circuits de protection, la technologie standard, le temps réel mou et dur, et l'utilisation sous conditions difficiles.

3.2. Caractère Métrologie

Comme l'acquisition de nombreux types de signaux, la haute précision de signaux, et la réduction des incertitudes de mesure.

3.3. Caractère Informatique

Comme la connexion entre le monde physique et le monde numérique, l'interaction avec des capteurs/actuateurs, l'intégration dans le réseau de l'entreprise, l'interconnexion entre les systèmes et à l'extérieur, et les technologies internet.

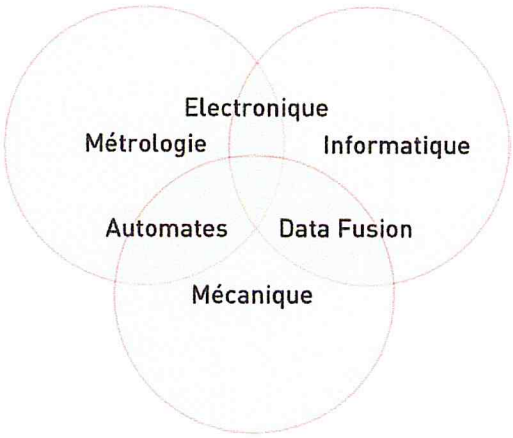


Figure 3 : Les caractéristiques d'un système cyber-physique.

3.4. Caractère Automates

Comme la connexion aux logiciels d'automates les plus courants, le transfert seule des données appréciables, le transfert de données physiques, et les systèmes embarqués cyber-physique.

3.5. Caractère Fusion de données

Comme la combinaison et le calcul de données de différentes sources, la fusion de données multi-sensorielle, et les réseaux des capteurs/actuateurs.

3.6. Caractère Mécanique

Comme le boîtier robuste contre les perturbations, la résistance contre les chocs, l'intégration dans la machine, et le fonctionnement continu.

4. Domaines d'application d'un système cyber-physique

Une génération de précurseurs des systèmes cyber-physique peut être trouvée dans des domaines aussi variés que [4] :

- La fabrication (industriel) par exemple un système cyber-physique peut améliorer les processus de partage de l'information en temps réel entre les machines industrielles, la fabrication de la chaîne d'approvisionnement, les fournisseurs, les systèmes d'affaires, et les clients.
- Les soins de santé par exemple les dispositifs médicaux, les réseaux de gestion de la santé, ou bien un système cyber-physique peut améliorer les traitements pour les patients handicapés et les personnes âgées.
- Les énergies renouvelables par exemple les réseaux intelligents, ou bien un système cyber-physique peut contrôler et assurer une meilleure fiabilité, et améliorer l'efficacité énergétique.
- Les constructions intelligentes par exemple un système cyber-physique interagissent pour réduire la consommation d'énergie, et d'améliorer le confort des habitants.
- Le transport par exemple les réseaux véhicules et les autoroutes intelligents, prévenir les accidents ou la congestion, et améliorer la sécurité.
- L'agriculture par exemple la surveillance et le contrôle de la chaleur, de l'humidité, l'arrosage et la santé des plantes, ou bien un système cyber-physique peut recueillir des informations fondamentales sur le climat, le sol.
- Les réseaux informatiques par exemple amélioration les performances et la gestion des ressources, surveiller les ressources disponibles, les réseaux sociaux populaires et sites e-commerce de l'information de navigation.

5. Le défi de la conception cyber-physique

Sont qualifiés de cyber-physiques les systèmes embarqués complexes conçus pour interagir avec leur environnement de manière continue et dynamique via l'association d'éléments informatique, de communication et physique. On les trouve dans les réseaux intelligents (smart grids), les réseaux de circulation de véhicules, les bâtiments intelligents, les robots coopératifs, les systèmes automobiles et avioniques.

La conception d'un système cyber-physique est soumise à des contraintes (coût, consommation électrique, fiabilité, performances). Mais elle doit avant tout assurer

l'interaction intelligente, dynamique et prévisible du système avec le monde réel. Les ingénieurs sont chargés de mettre en avant le comportement du système selon son environnement, en évitant l'utilisation d'outils et de techniques disparates, pour favoriser la conception "au niveau système", qualifiée de "holistique". Cette approche permet de modéliser les interactions entre les mondes informatique et physique pour prévenir d'éventuels dysfonctionnements. Elle aurait sans doute permis d'éviter la panne d'électricité de 2003 qui a affecté 55 millions de personnes pendant 3 jours, au nord-est des Etats-Unis. Une des méthodes qui a fait ses preuves repose sur la modélisation destinée à concevoir, analyser, vérifier et valider des systèmes dynamiques, les modèles utilisés provenant des spécifications système et de l'analyse de l'environnement. Les outils de conception de systèmes ayant les niveaux appropriés d'abstraction permettent aux modèles d'être automatiquement combinés, simulés et déployés, et de s'adapter aux exigences de suivi et de vérification de type Hardware In the Loop (matériel dans la boucle).

Autre méthode éprouvée : la conception basée sur une plate-forme dans les secteurs automobile et aérospatial. Elle permet de construire une structure étendue à des systèmes volumineux et complexes avec de longues durées de vie. Elle peut être utilisée comme couche d'abstraction pour penser aux contraintes de l'application sans se soucier des ajustements au niveau de l'implémentation. Elle permet de définir les éléments ayant des interconnexions claires, donnant lieu à des conceptions indépendantes, composables et modularités. Il est ainsi possible de réutiliser ou remanier les éléments d'une plate-forme pour des structures de test, du suivi, de la vérification et de la documentation de spécifications.

Ces deux approches sont complémentaires, souvent utilisées en parallèle. Il convient surtout de disposer d'une méthode de conception disciplinée, d'outils de développement holistique ainsi que du matériel "sur étagère" prêt à l'emploi pour relever les défis des systèmes cyber-physiques. [5]

6. Les systèmes Cyber-physiques et le concept Industrie 4.0

Le concept d'Industrie 4.0 correspond à une nouvelle façon d'organiser les moyens de production : l'objectif est la mise en place d'usines dites "intelligentes" ("smart factories") capables d'une plus grande adaptabilité dans la production et d'une allocation plus efficace

des ressources, ouvrant ainsi la voie à une nouvelle révolution industrielle [6]. Ses bases technologiques sont l'Internet des objets et les systèmes cyber-physiques.

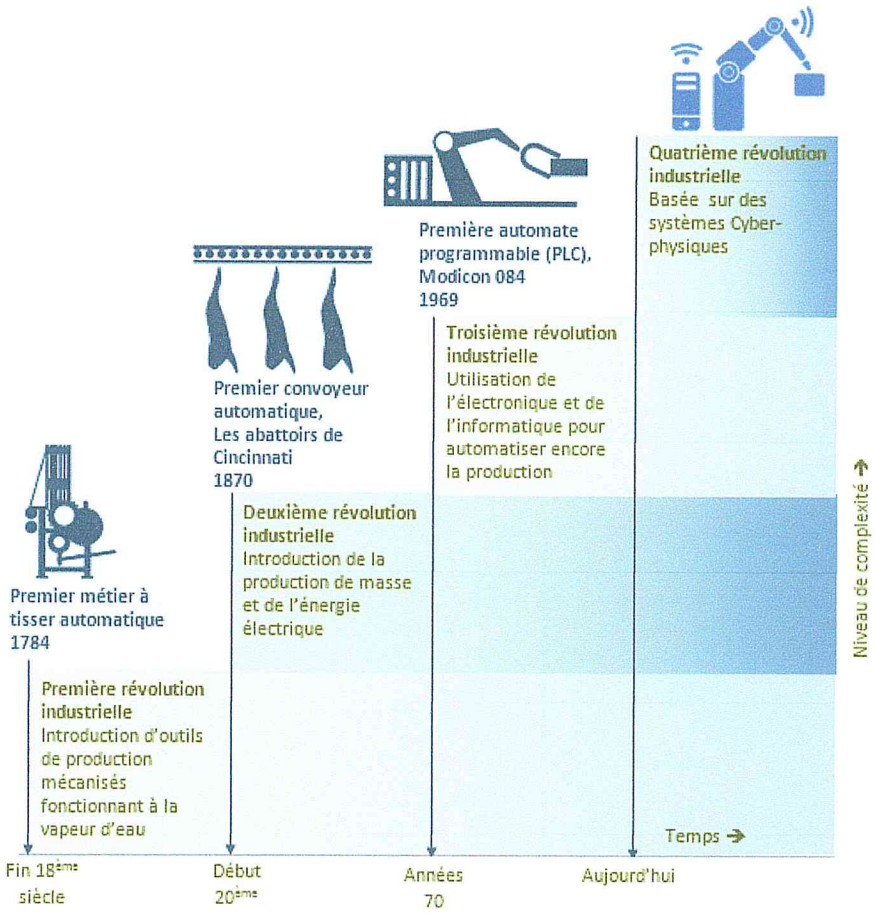


Figure 4 : De l'Industrie 1.0 à l'Industrie 4.0.

L'Industrie 4.0 est l'un des projets clés de la stratégie concernant les hautes technologies du gouvernement allemand, qui encourage la révolution numérique des industries.

Aux États-Unis le projet de "Coalition pour le leadership dans la fabrication intelligente" travaille aussi sur le futur de la fabrication industrielle. Indépendamment, General Electric travaille depuis quelques années sur un projet intitulé l'Internet Industriel qui cherche à associer les avancées de deux révolutions :

- La multiplicité de machines, de dispositifs et de réseaux qui résulte de la révolution industrielle.
- Les évolutions plus récentes des systèmes d'information et de communication apportées par la révolution de l'Internet.

L'Industrie 4.0 correspond en quelque sorte à la numérisation de l'usine. À travers le recours à l'internet des objets et aux systèmes cyber-physiques, c'est-à-dire aux réseaux virtuels servant à contrôler des objets physiques, l'usine intelligente se caractérise par une communication continue et instantanée entre les différents outils et postes de travail intégrés dans les chaînes de production et d'approvisionnement. L'utilisation de capteurs communicants apporte à l'outil de production une capacité d'autodiagnostic et permet ainsi son contrôle à distance tout comme sa meilleure intégration dans le système productif global [7].

7. Conclusion

Dans ce chapitre, nous avons présenté les notions fondamentales d'un système cyber-physique. Les systèmes cyber-physiques sont basés sur de nouvelles technologies électroniques et infotroniques des systèmes physiques et sur une distribution de la décision sur les entités de ce système.

Le prochain chapitre sera consacré à la présentation d'un système multi-agents et la simulation des systèmes de production.

Chapitre II :

Un système multi-agents et la simulation des systèmes de production

1. Introduction

La technologie des agents est un nouveau paradigme de conception et de programmation d'applications informatiques distribuées et adaptatives. Durant ces dernières années, beaucoup d'applications importantes, entre autres celles dédiées à internet, au calcul distribué et au développement de la nouvelle génération des systèmes de recherches gravitent autour de l'analyse, la conception et l'implémentation des systèmes multi-agents.

Dans ce chapitre, nous proposons de présenter les éléments de bases du travail accompli : la plateforme de développement multi-agent JADE, la simulation hardware in the loop, et la simulation à évènements discrets et FlexSim.

2. Le concept d'agent

Cette section donne d'abord une définition du concept d'agent, ainsi que ses caractéristiques et ses types.

2.1. Définition d'un agent

Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents [8].

- ✓ **Remarque :**

Un agent ne peut exister sans environnement. L'environnement est une structure dans laquelle l'agent évolue. Un agent va agir sur son environnement et l'environnement va agir sur l'agent.

2.2. Les caractéristiques d'un agent

Un agent est une entité qui possède les caractéristiques suivantes [9]:

- **Autonome :** Il prend des décisions motivées par son état interne sans intervention extérieure.
- **Réactif :** Un agent est situé dans un environnement. Il est capable de percevoir cet environnement et de réagir aux changements qui interviennent par ses actions.
- **Social :** Il est capable d'interagir avec d'autres agents.
- **Proactif :** Il ne fait pas que réagir à son environnement mais il est capable de lui-même de produire des actions motivées par des buts.

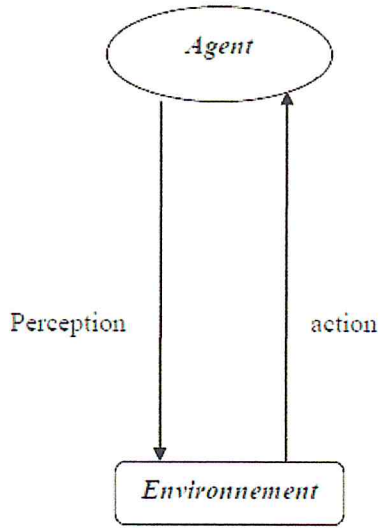


Figure 5 : Un Agent avec son environnement.

Généralement on ne trouve pas tous ses caractéristiques en un seul agent.

2.3. Les types d'agents

Les types d'agents sont [10] :

2.3.1. Agent cognitif

Il est intelligent par lui-même c'est-à-dire qu'il effectue un certain raisonnement pour choisir ses actions. Un tel raisonnement peut se faire soit en se basant sur les buts de l'agent, soit sur une certaine fonction d'utilité.

2.3.2. Agent réactif

Comme son nom l'indique, un agent réactif ne fait que réagir aux changements qui surviennent dans l'environnement. Autrement dit, un tel agent se contente simplement d'acquérir des perceptions et de réagir à celles-ci en appliquant certaines règles prédéfinies. Étant donné qu'il n'y a pratiquement pas de raisonnement, ces agents peuvent agir et réagir très rapidement.

2.3.3. Agent hybride

Chaque agent hybride est caractérisé par la notion de couches et chaque couche représente soit les agents cognitifs, soit les agents réactifs. Donc l'agent hybride combine entre les deux comportements (comportement réactif et comportement cognitif).

3. Un système multi-agents

Cette section donne d'abord une définition du concept multi-agents, ainsi que ses caractéristiques, et ses domaines d'applications.

3.1. Définition d'un système multi-agents

Un système multi-agents est un ensemble organisé d'agents. Il est constitué d'une ou plusieurs organisations qui structurent les règles de cohabitation et de travail collectif entre agents. Un système multi-agents est défini comme un système dans lequel plusieurs entités (ou agents) autonomes et intelligentes interagissent ensemble dans le but de réaliser un ensemble de buts ou de tâches [8].

Un système multi-agents est composé des éléments suivants :

- Un environnement **E**, c'est-à-dire un espace disposant généralement d'une métrique.
- Un ensemble d'objets **O** situé dans **E**. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.
- Un ensemble **A** d'agents, qui représentent les entités actives du système.
- Un ensemble de relations **R** qui unissent des objets (et donc des agents) entre eux.
- Un ensemble d'opérations **Op** permettant aux agents de **A** de percevoir, produire, consommer, transformer et manipuler des objets de **O**.

- Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers [8].

3.2. Les caractéristiques d'un système multi-agents

Un système multi-agents est caractérisé par [11]:

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

3.3. Domaines d'application des systèmes multi-agents

Plusieurs domaines existent pour les systèmes multi-agents, nous citons [11]:

- La résolution distribuée de problèmes algorithmiques qui a comme problématique la distribution des algorithmes d'intelligence artificielle (recherche opérationnelle, planification, ordonnancement...).
- Applications industrielles : contrôle en temps réel, production, réseaux de télécommunications, systèmes de transport, systèmes de distribution, etc.
- Gestion de processus de business, support à la décision.
- Commerce électronique.
- Modélisation, Simulation et Analyse d'entités distribuées.
- Systèmes d'information coopératifs : découverte des sources, recherche de l'information, filtrage des informations, fusion des informations et personnalisation
- Applications Logiciels Distribuées.
- Interaction homme-machine.
- Mondes virtuelles.

3.4. Interaction entre agents et protocoles

Une des principales propriétés de l'agent dans un système multi-agents est celle d'interagir avec les autres. Ces interactions sont généralement définies comme étant toute forme d'action exécutée au sein de la société qui a pour effet de modifier le comportement d'un

autre agent. Elles donnent à l'agent la possibilité de participer à la satisfaction d'un but global.

Trois types d'interaction existent entre agents [12] :

- **L'interaction par la communication :** La communication, dans les Systèmes multi-agents est souvent l'un des moyens utilisés pour échanger des informations entre agents. Il faut savoir combiner entre la communication et les actions afin de coordonner et de contrôler les échanges entre plusieurs agents pour avoir un comportement collectif cohérent du système.
- **L'interaction par la coopération d'agent :** Des agents totalement coopératifs peuvent changer leurs buts pour répondre aux besoins des autres agents afin d'assurer une meilleure coordination entre eux.
- **L'interaction par la négociation entre ressources :** La négociation est avant tout un mécanisme de résolution de conflit. Cette résolution peut alors s'effectuer dans un contexte favorable ou défavorable. Dans un contexte favorable, toutes les parties sont prêtes à faire les concessions nécessaires pour arriver à un accord. Dans un contexte défavorable l'objectif principal pour au moins l'un des participants, est d'obtenir un accord qui le favorise même aux dépends des autres. De ce fait, la négociation peut être décrite comme une coopération dont l'objectif commun est l'obtention d'un accord.

4. Plateformes de développement des systèmes multi-agents

Une plateforme multi-agents est un outil permettant de faciliter la construction et l'exploitation d'un système multi-agents. Elle peut prendre différentes formes, allant d'outils d'ordre méthodologique, à des outils de développement, ou des supports d'exécution. Ce qui importe, c'est qu'à un moment ou un autre, la plateforme simplifie la tâche du développeur. Il existe actuellement de nombreuses plateformes multi-agents qu'on peut classer en plusieurs catégories [13] :

- Les plateformes pour agents mobiles (Voyager, Odissey, Aglet, etc.) qui fournissent la mobilité à des agents.
- Les plateformes pour agents cognitifs (Agent Builder, etc.) dans lesquels on trouve les plateformes se référant avec la norme FIPA (JADE, FIPA-OS, etc.) ou à la norme KQML (JAT, JAT-Lite, etc.).

- Les plateformes pour agents collaboratifs (Zeus, JAFMAS, KAoS, JAFIMA, etc.).
- Les plateformes pour la simulation multi-agents (Cormas, Swarm, etc.).

5. La plateforme JADE

Cette section donne d'abord une définition de JADE, la norme FIPA pour les systèmes multi-agents, ainsi que les outils de débogage de JADE.

5.1. Définition de JADE

JADE (Java Agent Development Framework) est une plate-forme multi-agents développée en Java par CSELT (Groupe de recherche de Gruppo Telecom, Italie) qui a comme but la construction des systèmes multi-agents et la réalisation d'applications conformes à la norme FIPA (Foundation for Intelligent Physical Agents). JADE comprend deux composantes de base : une plateforme agents compatible FIPA et un paquet logiciel pour le développement des agents Java [14].

5.2. La norme FIPA pour les systèmes multi-agents

FIPA (Foundation for Intelligent Physical Agents) en français : fondation pour l'intelligence physique des agents, est une association internationale sans but lucratif, travaillant pour produire des spécifications pour des technologies d'agent générique. [15]

Les premiers documents de spécification de la norme FIPA (FIPA 1997), appelés spécifications FIPA97, établissent les règles normatives qui permettent à une société d'agents d'inter-opérer. Tout d'abord, les documents FIPA décrivent le modèle de référence d'une plate-forme multi-agents comme l'illustre la Figure 6 où ils identifient les rôles de quelques agents clés nécessaires pour la gestion de la plate-forme, et spécifient le contenu du langage de gestion des agents et l'ontologie du langage.

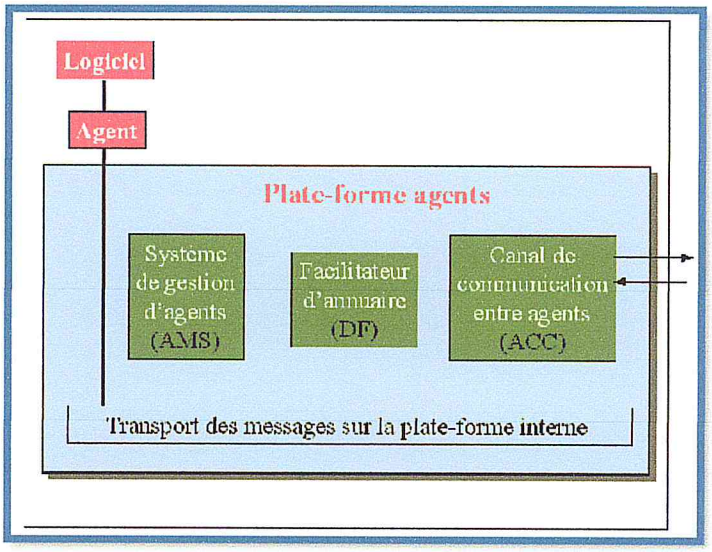


Figure 6 : Le modèle de référence pour une plate-forme multi-agents FIPA.

Dans la Figure 6, on voit qu'il y a trois rôles principaux dans une plate-forme multi-agents FIPA :

- **Le Système de Gestion d'Agents** (Agent Management System - **AMS**) est l'agent qui exerce le contrôle de supervision sur l'accès et l'usage de la plate-forme ; il est responsable de l'authentification des agents résidents et du contrôle d'enregistrements.
- **Le Canal de Communication entre Agents** (Agent Communication Channel - **ACC**) est l'agent qui fournit la route pour les interactions de base entre les agents dans et hors de la plate-forme ; c'est la méthode de communication implicite qui offre un service fiable et précis pour le routage des messages ; il doit aussi être compatible avec le protocole IIOP pour l'interopérabilité entre les différentes plates-formes multi-agents.
- **Le Facilitateur d'Annuaire** (Directory Facilitator - **DF**) est l'agent qui fournit un service de pages jaunes à la plate-forme multi-agents. Il faut remarquer qu'il n'y a aucune restriction sur la technologie utilisée pour l'implémentation de la plate-forme : émail, basé sur CORBA, applications multithreads Java, etc.

Le langage de communication de la plateforme JADE est FIPA-ACL (Agent Communication Language) en français : Langage de Communication d'Agents, est le langage standard des messages et impose le codage, la sémantique et la pragmatique des messages. La norme n'impose pas de mécanisme spécifique pour le transport interne de messages.

Plutôt, puisque les agents différents pourraient s'exécuter sur des plates-formes différentes et utiliser technologies différentes d'interconnexion, FIPA spécifie que les messages transportés entre les plates-formes devraient être codés sous forme textuelle [16].

5.3. Les outils de débogage de JADE

Pour supporter la tâche difficile du débogage des applications multi-agents, des outils ont été développés dans la plateforme JADE. Chaque outil est empaqueté comme un agent, obéissant aux mêmes règles, aux mêmes possibilités de communication et aux mêmes cycles de vie d'un agent générique (agentification de service) [17].

5.3.1. Agent RMA (Remote Management Agent)

Le RMA permet de contrôler le cycle de vie de la plateforme et tous les agents la composant. L'architecture répartie de JADE permet le contrôle à distance d'une autre plateforme. Plusieurs RMA peuvent être lancés sur la même plateforme du moment qu'ils ont des noms distincts.

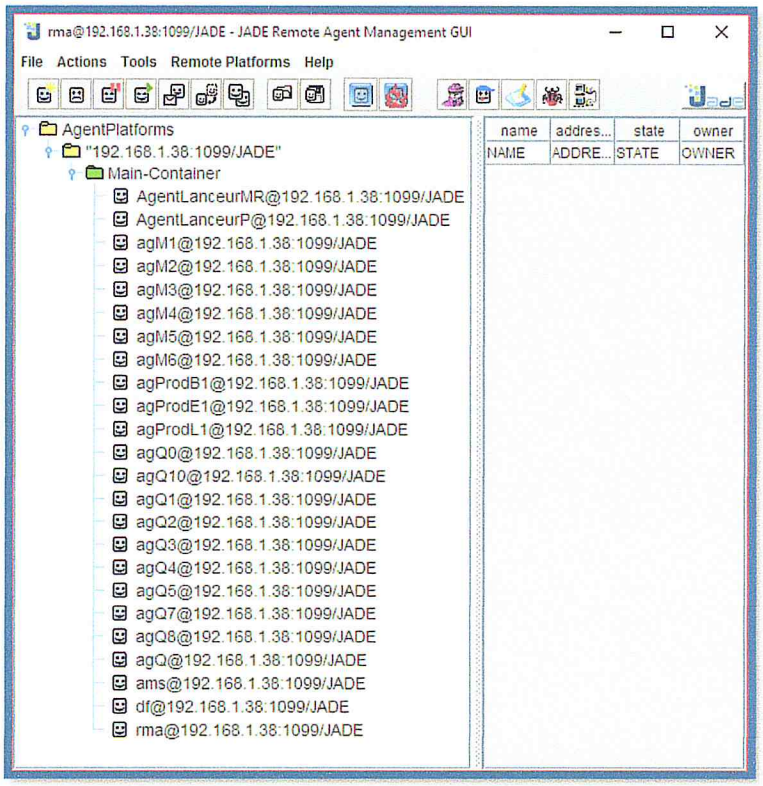


Figure 7 : L'interface de l'agent RMA.

5.3.2. Agent Dummy

L'outil Agent Dummy permet aux utilisateurs d'interagir avec les agents JADE d'une façon particulière. L'interface permet la composition et l'envoi de messages ACL et maintient une liste de messages ACL envoyés et reçus. Cette liste peut être examinée par l'utilisateur et chaque message peut être vu en détail ou même édité. Plus encore, le message peut être sauvegardé sur le disque et renvoyé plus tard.

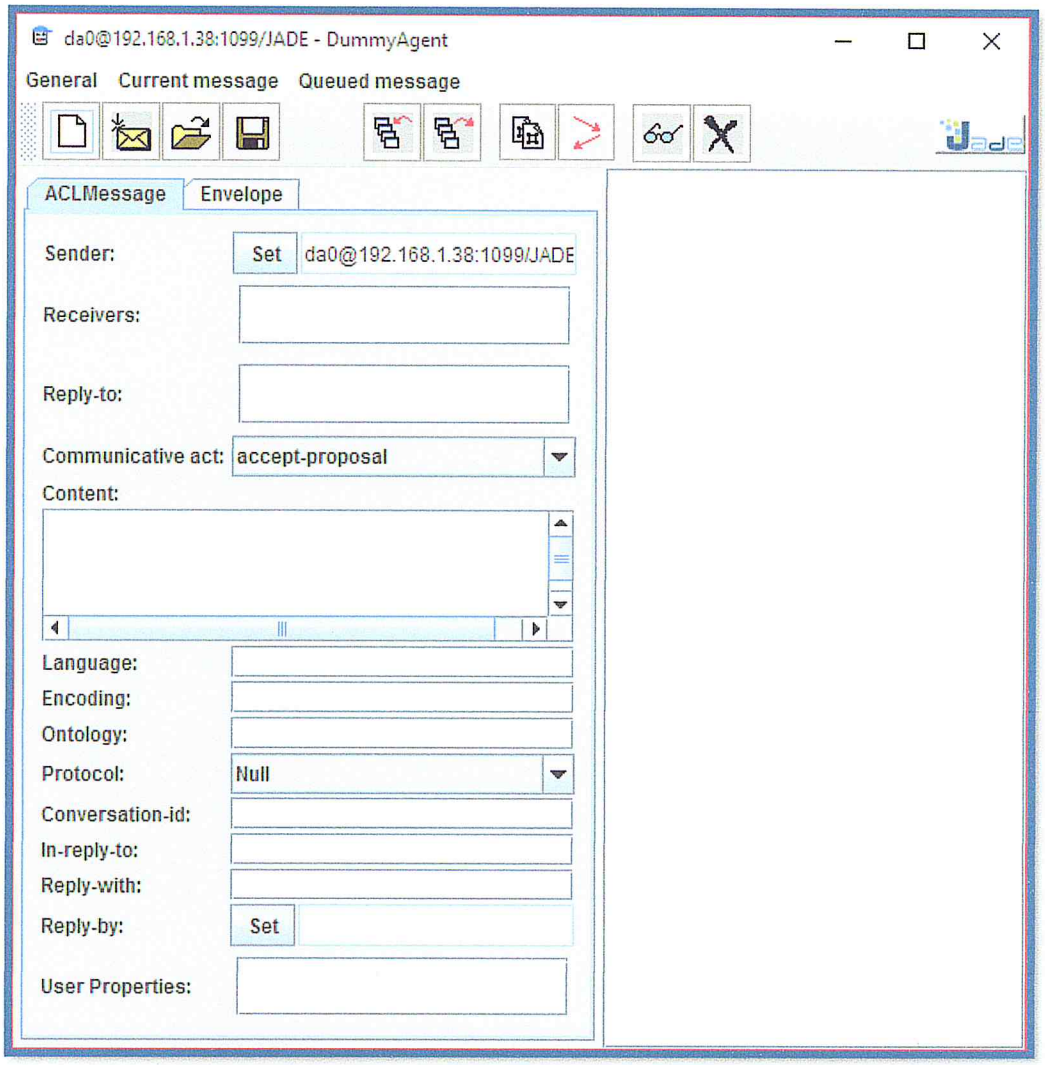


Figure 8 : L'interface de l'agent Dummy.

5.3.3. Agent Directory Facilitator

L'interface du DF peut être lancée à partir du menu du RMA. Cette action est en fait implantée par l'envoi d'un message ACL au DF lui demandant de charger son interface graphique. L'interface peut être juste vue sur l'hôte où la plateforme est exécutée. En utilisant cette interface, l'utilisateur peut interagir avec le DF.

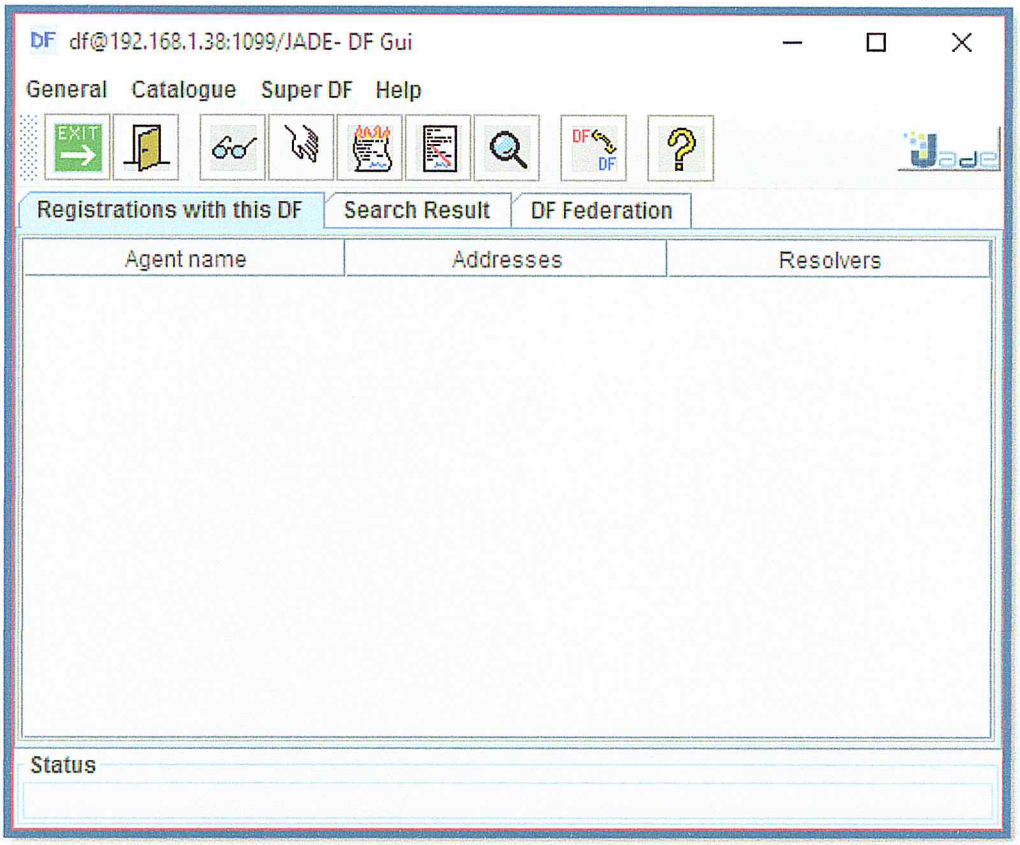


Figure 9 : L'interface de l'agent DF.

5.3.4. Agent Sniffer

Quand un utilisateur décide d'épier un agent ou un groupe d'agents, il utilise un agent Sniffer. Chaque message partant ou allant vers ce groupe est capté et affiché sur l'interface de Sniffer. L'utilisateur peut voir et enregistrer tous les messages, pour éventuellement les analyser plus tard. L'agent peut être lancé du menu RMA ou de la ligne de commande suivante : Java jade.Boot sniffer.jade.tools.sniffer.sniffer.

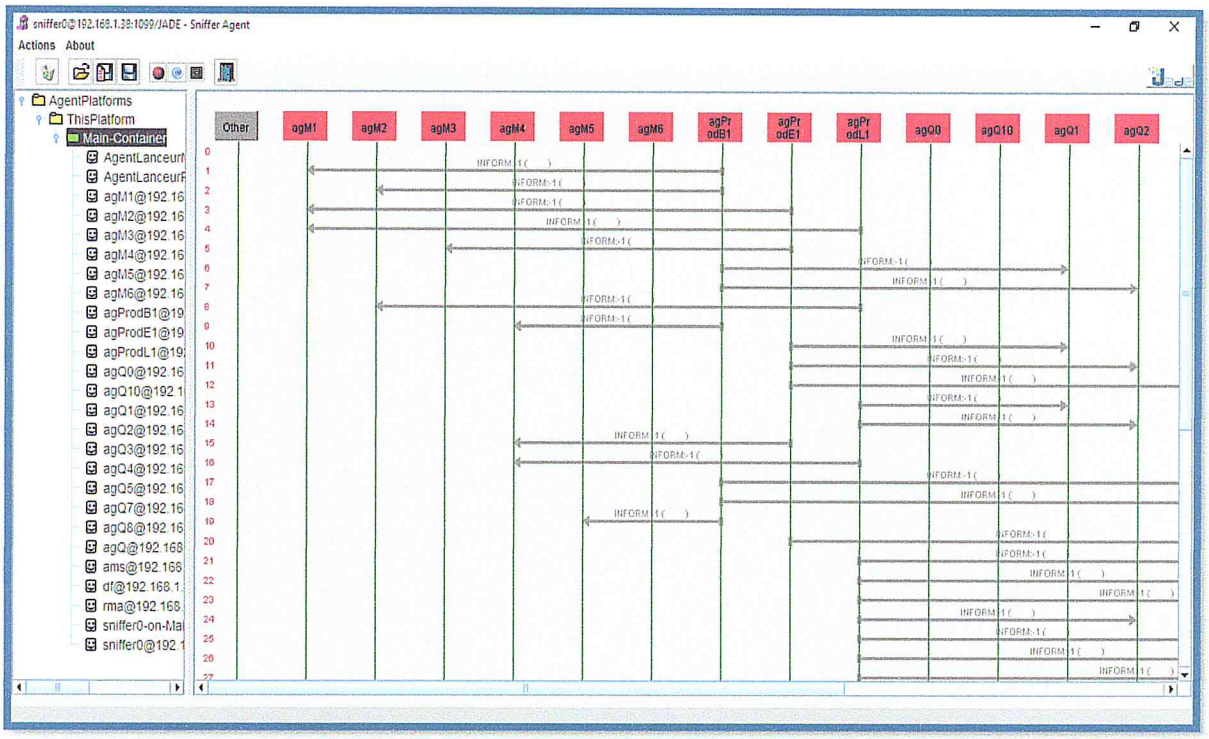


Figure 10 : L'interface de l'agent Sniffer.

5.3.5. Agent Inspector

Cet agent permet de gérer et de contrôler le cycle de vie d'un agent s'exécutant et la file de ses messages envoyés et reçus.

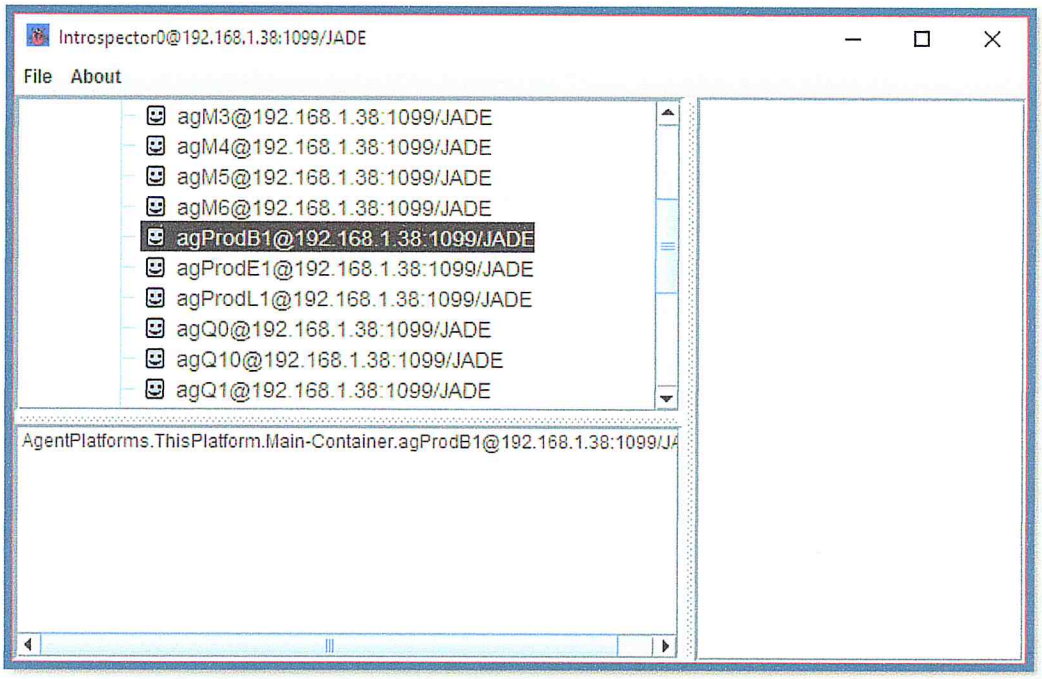


Figure 11 : L'interface de l'agent Inspector.

6. La simulation des systèmes de production :

La simulation est largement répandue dans le monde industriel, et dans le monde de la recherche académique.

6.1. La simulation HIL (Hardware in the Loop)

La simulation Hardware-in-the-loop (traduisible par « matériel dans la boucle ») est une méthode de simulation caractérisée par l'association de véritables composants, connectés à une partie temps-réel simulée.

Habituellement, les systèmes de contrôles matériel et logiciel sont identiques à ceux retenus pour la production en série.

Le processus à contrôler (actionneurs, système physique, et ses capteurs) peut être composé soit d'éléments simulés, soit d'éléments réels (cf. illustration plus bas). En général, un mixe des deux est réalisé.

Fréquemment, les actionneurs sont réels, et le système physique ainsi que les capteurs sont simulés. Cela s'explique par le fait que les actionneurs et système de contrôle font, la plus part du temps, partie du même sous-ensemble, ou bien, que les actionneurs sont difficilement "modélisables" précisément et "simulable" en temps-réel.

Pour des raisons évidentes, l'utilisation de véritables capteurs dans un système simulé peut demander de considérables efforts de réalisations. L'entrée physique du capteur (déplacement, vitesse, température, pression etc.) doit être alors générée artificiellement. Ce qui est effectivement peu commode [18].

Les avantages offerts par la simulation Hardware in the loop sont multiples :

- La conception et les tests du système de contrôle peuvent être faits sans le système réel (le travail peut s'effectuer en "labo").
- Les conditions d'essais sur système de contrôle matériel peuvent être poussées à l'extrême (exemple : haute/basse température, fortes accélérations et choques mécaniques, compatibilité électromagnétique).
- Il est possible d'effectuer des essais sur les effets de défauts et pannes des actionneurs, capteurs et ordinateurs sur l'ensemble du système.
- Opérations et essais en conditions extrêmes et dangereuses.
- Expériences reproductibles et répétables à souhait.
- Opérations facilitées avec différentes interfaces hommes-machines (conception de cockpit et formation des opérateurs).
- Économie de temps et d'argents dans le processus de développement.

6.2. La simulation à événements discrets et FlexSim

Le survol des travaux de recherche existants montre que la simulation à événements discrets est d'un usage plus large que la simulation continue.

6.2.1. La simulation à événements discrets

Dans une simulation à événements discrets, les variables d'état que l'on désire connaître à tout l'espace d'état du système. Il s'ensuit que l'espace d'état est dénombrable ou fini. Chaque changement d'état où événement se produit d'une manière discrète dans le temps à des instants que l'on appelle des dates d'événement. En général les événements sont contingents, c'est-à-dire que leur occurrence dépend de l'évolution du système et donc d'autres

événements antérieurs [19], Nous allons préciser les notions qui concourent à la réalisation des changements dynamiques du système :

- Une activité est un intervalle de temps pendant lequel l'état de la ressource ne change pas.
- Un événement est un instant précis de changement d'état de ressource.
- Un processus est une succession d'un nombre fini d'états d'une ressource.

Pour construire un modèle de simulation à événements discrets, trois approches existent : approche par événement, approche par activité et approche par processus.

Approche par évènement : c'est l'approche de base, dans cette approche, un système est modélisé en définissant les changements qui ont lieu aux instants d'occurrence des événements. Le concepteur doit donc déterminer les événements qui peuvent changer l'état du système et de développer ensuite la logique associée avec chaque type d'événement [20].

Approche par cycle d'activités : au lieu de répertorier des événements, dans cette approche on répertorie les types d'activités. La logique de changement d'état se fait alors en précisant les conditions nécessaires au début et à la fin d'une activité. Le déroulement de la simulation se fait à l'aide d'une horloge. A chaque pas, on teste, pour chaque activité, si les conditions nécessaires à son début ou à sa fin sont remplies [19].

Approche par processus : cette approche combine la simplicité de la description de l'approche par cycle d'activités et l'efficacité de l'approche par événements. Elle présente des séquences d'événements ou des activités similaires pour un type d'objet, défini sous forme de processus, la description du fonctionnement du système complet par macro représentations et les conflits et la synchronisation entre processus sont gérés par des règles d'interruption et de reprise [21].

FlexSim est l'un des logiciels de simulation à évènement discrets, dans la section suivante, nous allons introduire cet outil qui permet le développement, la modélisation, la simulation, la visualisation et la surveillance des activités et systèmes à évènements discrets.

6.2.2. Le logiciel FlexSim

FlexSim est un logiciel de simulation à événements discrets. Son avantage principal réside dans le fait qu'il utilise un environnement OpenGL qui permet d'avoir une représentation graphique des différents éléments en interaction. Il utilise une structure orientée objet qui comporte 4 classes :

- Ressource Fixe : déterminent l'écoulement du flux

- Ressource Mobile : reçoivent des tâches des Ressources Fixes
- Nœud : utilisés pour concevoir le parcours de travail des Ressources Mobiles
- Objet Visuel : permettent d'afficher et de recueillir des données

Les paramètres des objets et leurs logiques de fonctionnement peuvent être modifiés en utilisant le script FlexScript ou le C++ [22].

FlexSim est un des meilleurs logiciels de simulation sur le marché grâce à sa flexibilité, sa puissance, sa facilité d'utilisation et la qualité de son animation 3D comme l'illustre la Figure 12 Cette dernière est affichée en temps réel grâce au moteur graphique avancé de FlexSim.

La relation entre FlexSim et les bases de données (tel que Oracle ou Access) est possible, ainsi avec les structures de données (texte, Excel, ou les fichiers Word), et pratiquement tout dispositif matériel qui peut être connecté à un ordinateur [23].

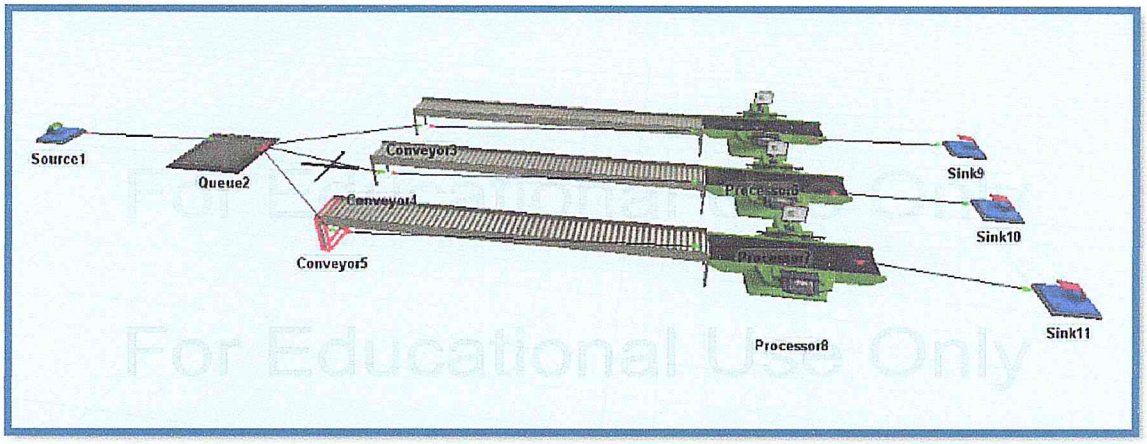


Figure 12 : Système de production sous FlexSim.

Les applications de simulation FlexSim sont utilisées pour la création des modèles de simulation à évènement discret, chaque application a été spécialement conçue pour le marché approprié afin de rendre le processus de modélisation rapide et efficace [23], parmi ces applications nous citons :

- FlexSim GP pour la simulation à usage général ;
- FlexSim Fabmodeler pour la simulation du processus de fabrication des semi-conducteurs ;
- FlexSim Port pour la simulation des terminaux de conteneurs maritimes.

Le développement des modèles sous FlexSim passe par cinq étapes, nous allons les citer brièvement :

- **Créer le modèle** : se fait en sélectionnant les objets de la bibliothèque et en les plaçant dans la fenêtre de disposition. Cette dernière est une vue spatiale 3D.
- **Connecter les objets du modèle** : FlexSim connecte les objets automatiquement dans l'ordre de disposition, chaque objet aura par défaut un port d'entrée, un port de sortie et un port central qui sert d'un nœud de référence pour d'autres objets. L'utilisateur pourra ajouter les ports dont il a besoin.
- **Détailler les objets** : le concepteur ajoutera la logique et les données aux objets tel que les temps de cycle, les capacités, les vitesses, la logique de routage etc. FlexScript ou le C++ peuvent être saisi directement par l'utilisateur ou en entrées logiques complexes.
- **Exécuter le modèle** : Une fois qu'un modèle est créé et la logique est affectée aux objets, le concepteur peut commencer la simulation en exécutant différents scénarios.
- **Examiner la sortie** : les résultats peuvent être vus dynamiquement lors de la simulation du modèle (en 2D, 3D et RV), les résultats peuvent également être exportés à des logiciels externes à travers DDE, DLL, ODBC, SQL ou Windows Sockets connexions.

Apres l'analyse des résultats, le concepteur pourra effectuer des modifications au modèle, une fois validé, les changements vont être appliqués au système réel.

Avec la possibilité de se connecter aux systèmes externes, tels que des bases de données en temps réel et des systèmes de gestion d'entrepôt, des informations en temps réel peuvent être introduites dans un modèle FlexSim et utilisé pour surveiller (Ou même contrôler) le système réel.

7.Conclusion

Dans ce chapitre, nous avons présenté une vue générale sur le domaine des systèmes multi-agents. Le concept de base de ce domaine est la notion d'agent, qui représente une entité autonome capable de percevoir, de se représenter et d'agir sur son environnement.
Le prochain chapitre sera consacré à la partie conceptuelle de ce travail.

Chapitre III :

Description et spécification du système

1. Introduction

Dans le but de décrire et de spécifier le système réalisé, ce chapitre est organisé en deux parties. Dans la première partie nous décrivons le système et son fonctionnement. Dans la deuxième nous présentons la spécification du système en utilisant les langages de modélisation : AUML (Annexe A) et UML.

2. Problématique et description du système

Notre travail a pour objectif la mise en œuvre des systèmes cyber-physique à base de système multi-agents en intégrant le système de développement Android. Le système est constitué de deux couches comme l'illustre la Figure 13 : une couche de simulation Hardware in the Loop (HIL) et une couche de pilotage multi-agents (MA). L'intérêt étant principalement l'implémentation d'une architecture multi-agent (JADE), des protocoles et approches de coordination en développant un ordonnancement par rapport aux règles de priorité pour les machines et règles d'affectation pour les produits pour le pilotage d'une cellule de production flexible, ainsi que le déploiement du système sur une architecture Android.

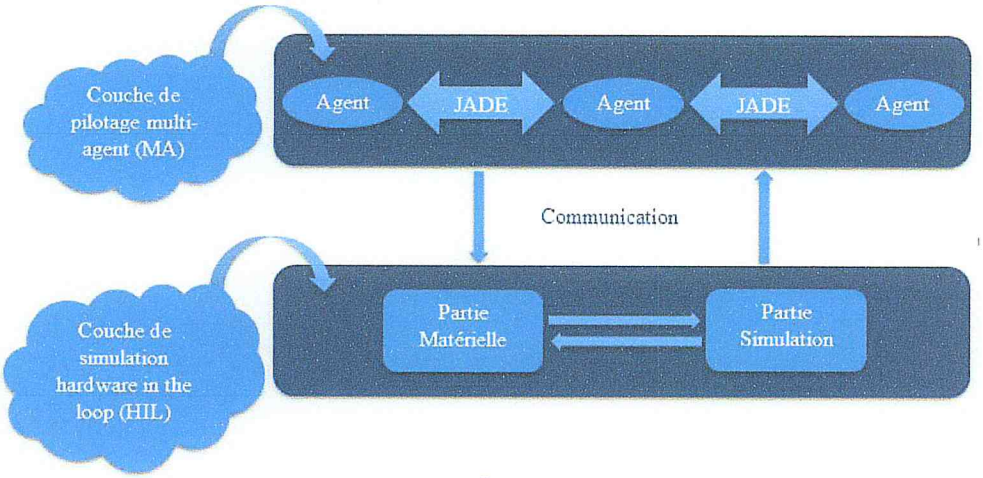


Figure 13 : Représentation générale du système.

3. Couche de simulation HIL

La couche de simulation HIL est composée de deux parties : la partie simulation et la partie matérielle.

3.1. La partie simulation

La cellule de production utilisée dans ce travail est l'AIP PRIMECA modélisée sous le logiciel de simulation à évènement discret FlexSim, qui a été présenté dans le chapitre précédant. Cette cellule a été utilisée dont l'objectif est de valider l'approche HILS piloté par les systèmes multi-agents comme l'illustre la Figure 14. Dans ce qui suit, nous allons présenter la cellule AIP PRIMECA.

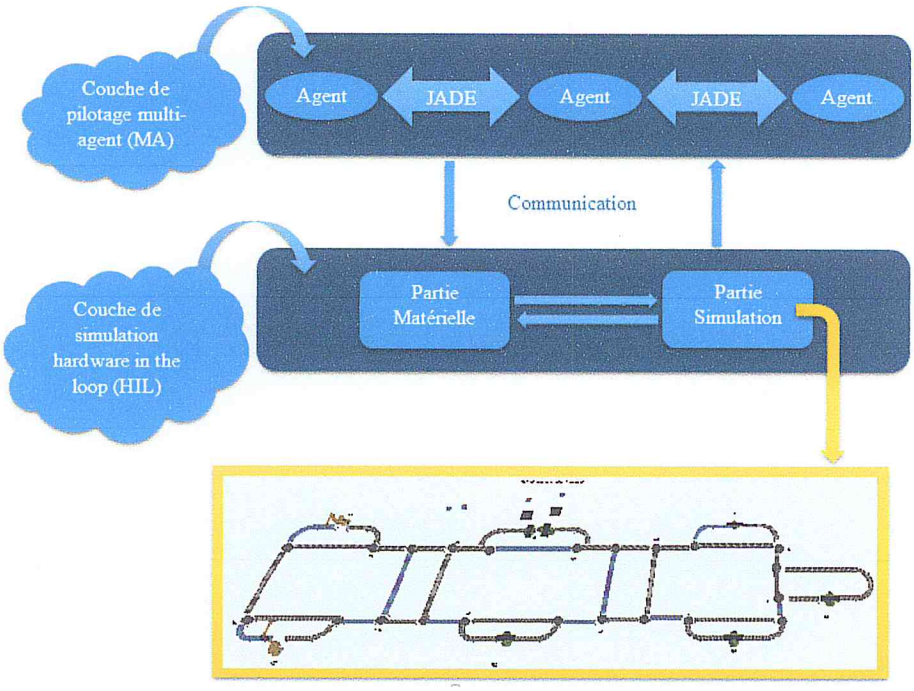


Figure 14 : Schéma global qui détaille la partie simulation.

- **Présentation de la cellule de production :**

La cellule AIP PRIMECA est une cellule flexible de production qui est implanté à l'université de Valenciennes (France). Elle est composée de sept machines :

- Un poste de chargement / déchargement (R1)
- Trois postes d'assemblage (R2, R3 et R4)
- Une unité d'inspection automatisée (R5)

- Une unité de réparation (R6), seul poste manuel
- Une station optionnelle (R7), utilisée dans certains scénarios avec perturbation [24].

Les deux dernières machines (R6 et R7) ne sont pas utilisées dans notre étude. La Figure 15 montre la cellule AIP-PRIMECA en localisant ses machines et la Figure 16 montre la cellule AIP-PRIMECA en réalité à l'université de Valenciennes (France).

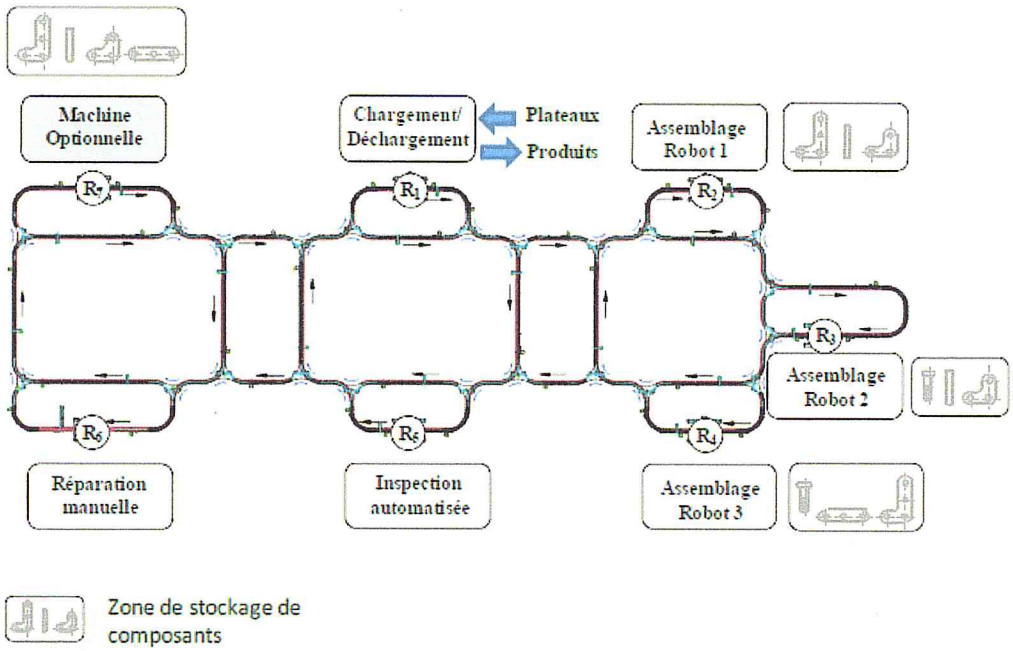


Figure 15 : Emplacement des machines de la cellule AIP-PRIMECA.

Figure 16 : la cellule AIP-PRIMECA en réalité à l'université de Valenciennes (France).

Il y a huit opérations qui peuvent être réalisées sur les machines de la cellule : ("Chargement", "Montage_Axe", "Montage_R", "Montage_I", "Montage_L", "Montage_Vis", "Inspection" et "Déchargement"). Par exemple "Montage_Axe" signifie qu'un Axe est pris du stock de la machine concernée et monté sur le plateau correspondant au produit en cours d'assemblage [24].

Dans ce système, sept types de produits peuvent être assemblés : "B", "E", "L", "T", "A", "I" et "P", ils sont basés sur les opérations effectuées par l'ensemble des machines. À chaque type de produit, une séquence de production est appliquée (chargement, déchargement, assemblages et l'opération d'inspection). Le Tableau 1 montre qu'un produit peut exécuter une opération donnée dans deux (02) machines différentes, par exemple l'opération "Montage_Axe" peut être effectuée soit par la machine R2 ou R3, et une machine peut effectuer plusieurs opérations d'assemblage, tel que R4 [25].

Produit	Chargement	Montage Axe	Montage R	Montage L	Montage I	Montage Vis	Inspection	Déchargement
(B)	10	60	40	0	20	20	5	10
(E)	10	60	40	20	0	0	5	10
(L)	10	60	0	0	40	40	5	10
(T)	10	40	20	20	0	0	5	10
(A)	10	60	20	20	20	20	5	10
(I)	10	40	0	0	20	20	5	10
(P)	10	40	20	20	0	0	5	10
Machine	R1	R2 R3	R2 R3	R2 R4	R4	R3 R4	R5	R1

Tableau 1 : Tableau des différents temps d'opérations(en seconde).

Dans le paragraphe suivant, nous décrivons le matériel utilisé dans la boucle de simulation.

3.2. La partie matérielle

Afin de valider la simulation HIL, nous allons intégrer des composantes matérielles au système de production comme l'illustre la Figure 17 et ce, pour rendre un produit intelligent, en utilisant un téléphone Android. Nous décrivons dans les points suivants le matériel utilisé :

- **Téléphone Android** : le téléphone que nous allons utiliser est de marque Samsung Glaxy Core Prime, doté de la version Android KitKat 4.4.4, le numéro du modèle est SM-G360H.

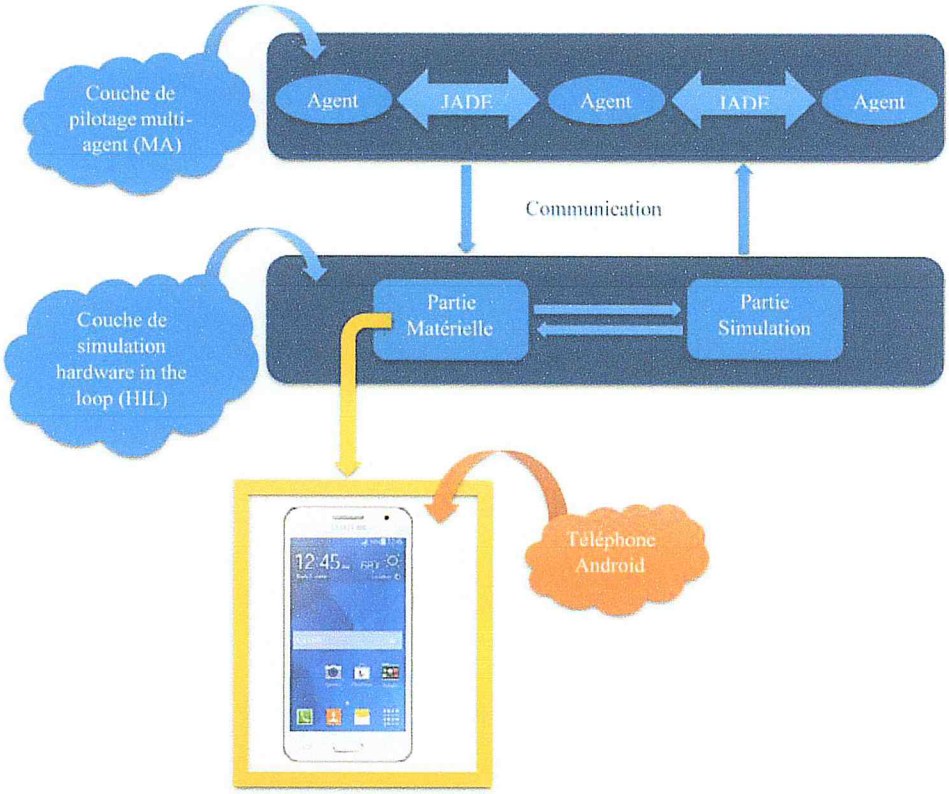


Figure 17 : Schéma global qui détaille la partie matérielle.

4. Couche de pilotage MA

Le système de pilotage multi-agents intègre les agents décideur (agent produit, agent machine et agent routage) comme l'illustre la Figure 18, ces agents sont associés respectivement à des entités dans le système de production (produit, machine et queue (point de routage)), chaque agent est consulté lors de la prise de décision (le produit choisit une machine pour une opération donnée, séquencement des produits pour chaque opération dans une machine et cheminement d'un produit…).

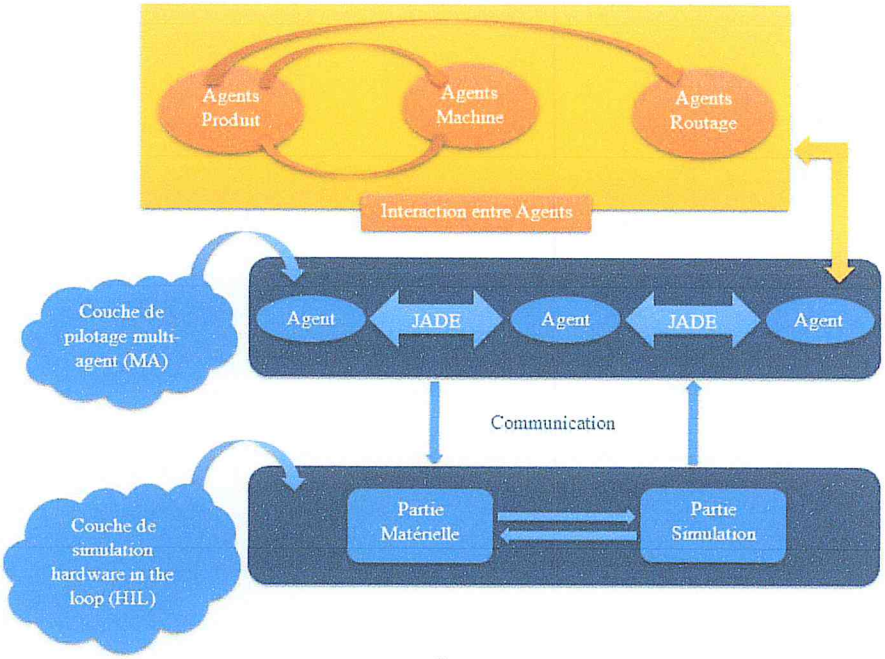


Figure 18 : Schéma global qui montre les agents du système de la couche de pilotage MA.

Le système de pilotage multi-agents comporte les agents suivants :

Agent Lanceur Machine et Routage : Il se charge de la création des agents machines et routage et l'agent lanceur produit.

Agent Lanceur Produit : Il se charge de la création dynamique des agents produits lors de la création des produits sur FlexSim.

Agent Produit : Chaque agent produit est associé à un produit sur FlexSim, il représente les compétences décisionnelles du produit dans le système. L'agent décide son activité (choix de la machine) selon la règle d'affectation et les informations envoyées par FlexSim.

Agent Machine : Il représente les compétences décisionnelles de la machine dans le système. L'agent machine choisit un des produits selon l'enregistrement et la file d'attente et selon les informations envoyées par FlexSim et la règle de priorité adoptée.

Agent Routage : Il indique aux agents produits les chemins à emprunter selon les informations envoyées par les agents machines à l'agent Produit, et l'état de la file d'attente.

5. Spécification du système

Dans cette section, nous présentons la modélisation des différentes composantes de notre système. Pour bien expliquer le fonctionnement du système nous allons donner une spécification orientée agent utilisant les diagrammes AUML pour la partie multi-agents.
La modélisation est réalisée avec les deux diagrammes suivants avec le logiciel Edraw Max :

- Diagramme de séquence.
- Diagramme de classe.

5.1. Conception orientée agent

Nous commençons par l'élaboration du diagramme de séquence, ensuite nous allons associer les diagrammes de classes.

5.1.1. Diagrammes de séquence

Le diagramme de séquence AUML permet de représenter les interactions, les échanges de messages et le protocole entre les agents.

1- Diagramme de séquence "choisir machine"

Ce cas d'utilisation est déclenché à la fin de chaque opération.

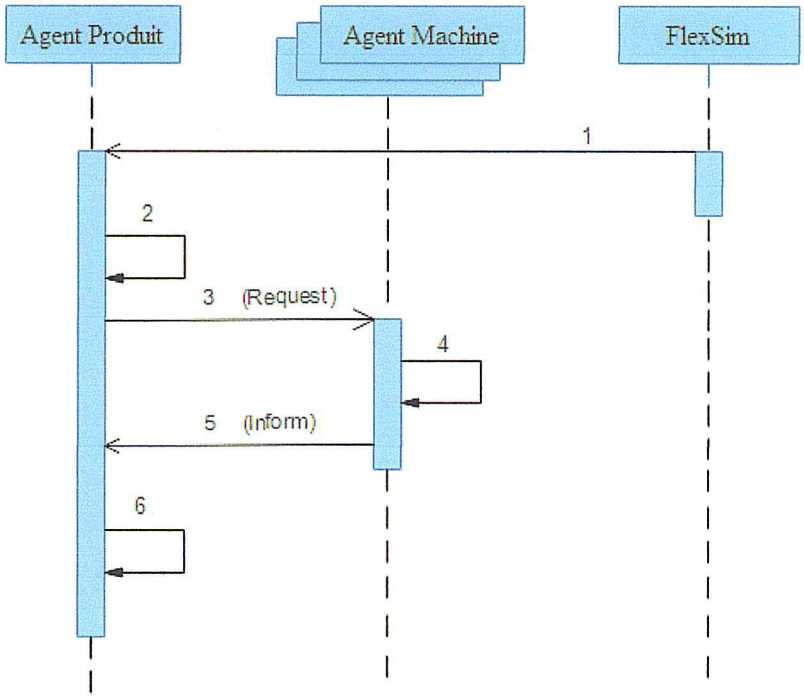


Figure 19 : Diagramme de séquence « choisir machine ».

1- FlexSim envoie un message à l'agent produit demandant l'opération suivante et la machine dans laquelle elle s'exécutera.
2- L'agent produit récupère les noms des machines de l'opération suivante.
3- L'agent produit demande le temps total des machines.
4- L'agent machine calcule le temps total de toutes les opérations enregistrées.
5- L'agent machine envoie un message à l'agent produit contenant le nom de la machine et le temps total.
6- L'agent produit compare les temps des machines et fait son choix machine.

Le choix machine se fait seulement dans les opérations suivantes : "Montage_Axe", "Montage_R", "Montage_I", "Montage_L" et "Montage_Vis".

2- Diagramme de séquence "enregistrement"

Ce cas d'utilisation est déclenché à chaque fin d'un choix machine.

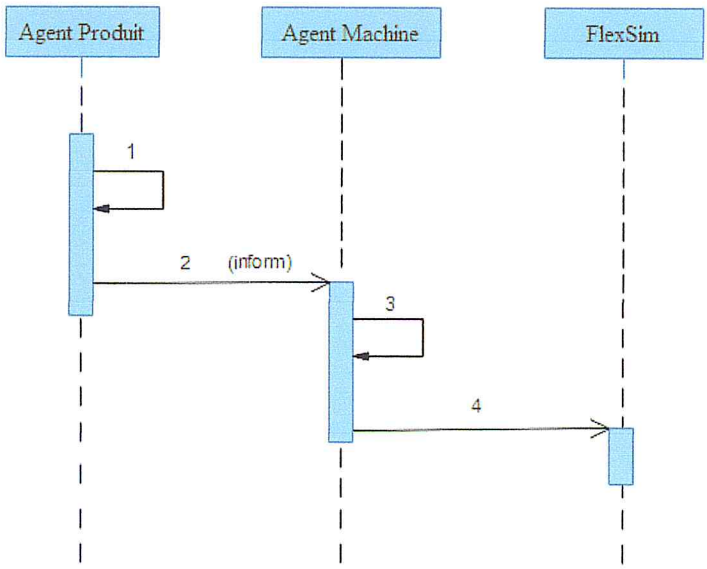


Figure 20 : Diagramme de séquence "enregistrement".

1- L'agent produit relève toutes les opérations possibles de la machine voulu.
2- L'agent produit envoie un message à l'agent machine pour s'enregistrer, contenant son identifiant ainsi que le temps total des opérations.
3- L'agent machine fait l'enregistrement de l'opération et du temps, au même temps le séquencement (FIFO).
4- L'agent machine envoie un message à FlexSim contenant l'identifiant du produit ainsi que le temps des opérations.

Remarque :

Les opérations : "Chargement", "Inspection" et "Déchargement" n'ont pas de choix machine, donc l'opération d'enregistrement se fait directement.

3- Diagramme de séquence "routage"

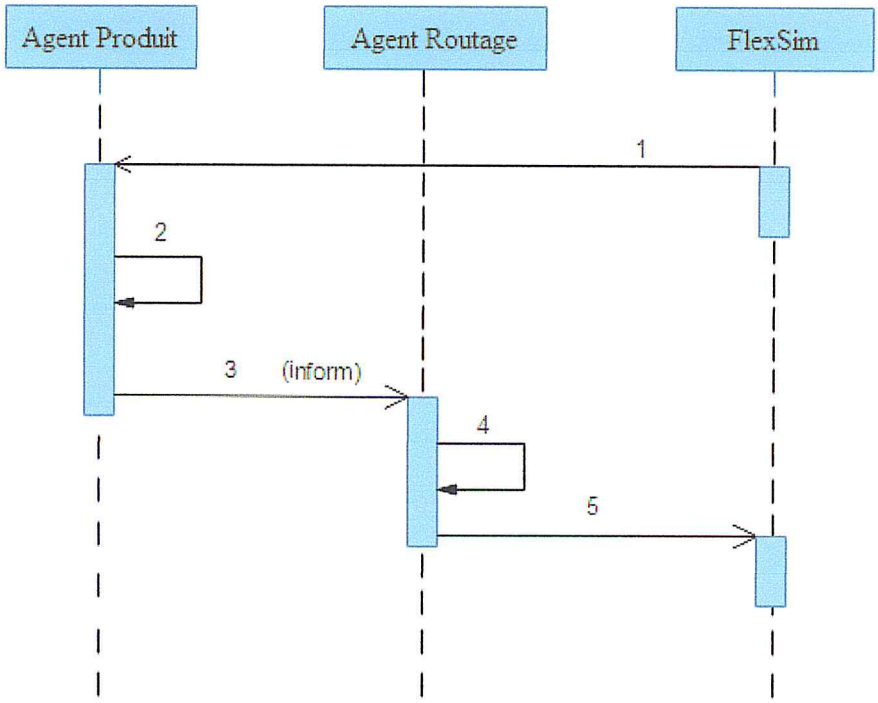


Figure 21 : Diagramme de séquence "routage".

1- FlexSim envoie un message à l'agent Produit contenant l'emplacement du point de routage et l'identifiant du produit.
2- L'agent produit relève le nom de la machine dans laquelle l'opération sera exécutée.
3- L'agent produit envoie à agent Routage l'identifiant du produit, le nom de la machine ainsi que le nom du routage.
4- L'agent routage fait un traitement selon le message reçu de l'agent produit.
5- L'agent routage envoie le cheminement à FlexSim.

5.1.2. Diagramme de classe

Ces classes montrent les interactions et les messages envoyés et reçus de chaque agent avec les autres agents du système.

1- La classe AUML de l'agent LanceurMR (Agent lanceur des agents machine et routage)

Figure 22 : La classe Agent Lanceur Machine et Routage.

Description de l'agent

- **Les attributs :**
  - t1 : une instance de l'interface AgentController qui permet l'accès à un agent Jade.

2- La classe AUML de l'agent LanceurP (Agent lanceur de produit)

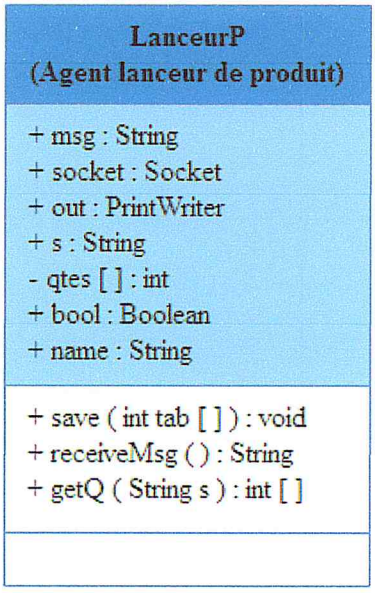


Figure 23 : La classe Agent Lanceur Produit.

Description de l'agent

- **Les attributs** :
  - msg : message reçu de la part du serveur (FlexSim).
  - socket : une instance de la classe Socket du paquetage java.net qui nous permet de mettre en œuvre une connexion bidirectionnelle entre notre programme « client » et le serveur situé sur FlexSim.
  - out : une instance de la classe PrintWriter qui permet d'écrire dans un flux pour échanger des messages entre le client et le serveur.
  - s : message reçu de la part des agents routages et machine.
  - qtes : vecteur de synchronisation entre le thread de réception du serveur et le thread de l'agent (setup) pour passer le message reçu.
  - bool : variable permettant l'attente entre le thread de l'agent (setup) et le thread de réception.
  - name : variable contenant le nom de chaque Agent Produit.
- **Les opérations :**
  - save (int tab []) : méthode permettant de sauvegarder le message reçu dans un tampon sous la forme de vecteur.
  - receiveMsg () : méthode permettant de récupérer le message envoyé du serveur au client.
  - getQ (String S) : méthode permettant de décomposer le message reçu du serveur, le convertir en entier et le mettre dans un vecteur.

3- La classe AUML de l'agent AgProd (Agent produit)

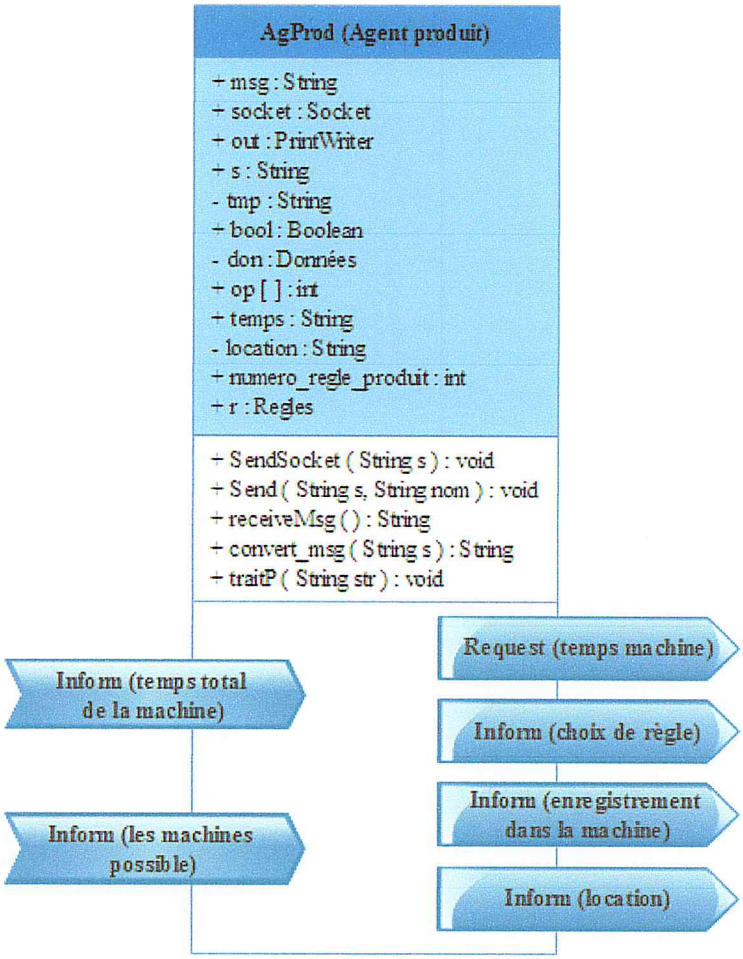


Figure 24 : La classe Agent Produit.

Description de l'agent

- **Les attributs :**
  - msg : message reçu de la part du serveur (FlexSim).
  - socket : une instance de la classe Socket du paquetage java.net qui nous permet de mettre en œuvre une connexion bidirectionnelle entre notre programme « client » et le serveur situé sur FlexSim.
  - out : une instance de la classe PrintWriter qui permet d'écrire dans un flux pour échanger des messages entre le client et le serveur.
  - s : message reçu de la part des agents routages et machine.
  - tmp : variable de synchronisation entre le thread de réception du serveur et le thread de l'agent (setup) pour faire passer le message reçu du serveur à l'agent.

- bool : variable permettant l'attente entre le thread de l'agent (setup) et le thread de réception.
- don : une instance de la classe "Données" qui nous permet de récupérer des informations sur les produits.
- op : vecteur contenant les états des opérations, vecteur de flag (à 1 si l'opération est réalisée, à 0 sinon).
- temps : le temps d'exécution d'une opération dans une machine.
- location : variable contenant le nom de la machine pour le routage.
- numero_regle_produit : choisir une des règles produit.
- r : une instance de la classe "Regles" qui nous permet de récupérer la règle produit choisis pour faire les calculs temps d'exécution dans une machine.

- **Les opérations :**
  - SendSocket (String s) : méthode d'envoi de message de l'agent produit (client) vers FlexSim (serveur).
  - Send (String s, String nom) : méthode permettant l'envoi de message entre les agents.
  - receiveMsg () : méthode qui permet de récupérer le message envoyé du serveur au client.
  - convert_msg (String s) : méthode de décomposition du message reçu du serveur, le convertit en entier et le met dans un vecteur.
  - traitP (String str) : méthode de traitement du choix de la machine selon la règle de la machine la moins chargée et du routage selon la machine choisie.

4- La classe AUML de l'agent AgM (Agent machine)

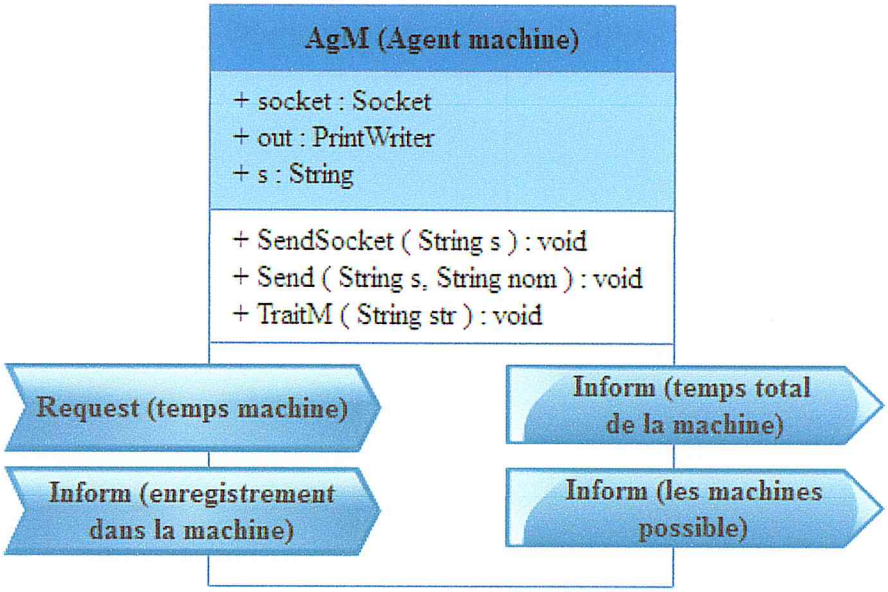


Figure 25 : La classe Agent Machine.

Description de l'agent

- **Les attributs** :
  - socket : une instance de la classe Socket du paquetage java.net qui nous permet de mettre en œuvre une connexion bidirectionnelle entre notre programme « client » et le serveur situé sur FlexSim.
  - out : une instance de la classe PrintWriter qui permet d'écrire dans un flux pour échanger des messages entre le client et le serveur.
  - s : message reçu de la part de l'agent Produit.
- **Les opérations** :
  - SendSocket (String s) : méthode d'envoi de message de l'agent machine (client) vers FlexSim (serveur).
  - Send (String s, String nom) : méthode permettant l'envoi de message entre les agents.
  - TraitM (String str) : méthode de traitement des messages reçus des agents produit pour le processus d'enregistrement.

5- La classe AUML de l'agent AgQ (Agent routage)

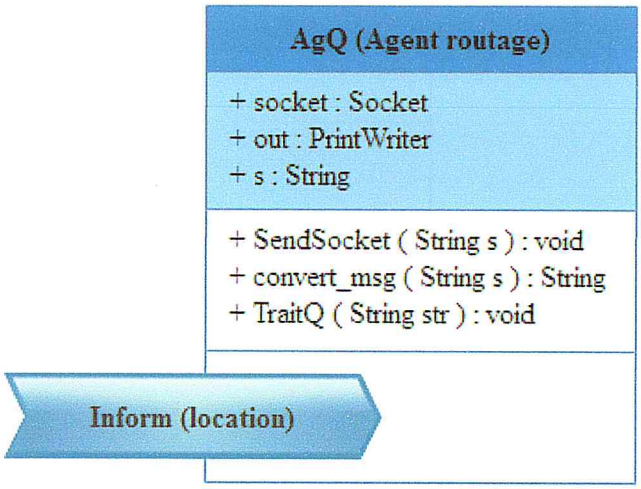


Figure 26 : La classe Agent Routage.

Description de l'agent

- **Les attributs :**
  - socket : une instance de la classe Socket du paquetage java.net qui nous permet de mettre en œuvre une connexion bidirectionnelle entre notre programme « client » et le serveur situé sur FlexSim.
  - out : une instance de la classe PrintWriter qui permet d'écrire dans un flux pour échanger des messages entre le client et le serveur.
  - s : message reçu de la part de l'agent produit.
- **Les méthodes** :
  - SendSocket (String s) : méthode d'envoi de message de l'agent machine (client) vers FlexSim (serveur).
  - convert_msg (String s) : méthode de décomposition du message reçu du serveur, le convertit en entier et le met dans un vecteur.
  - TraitQ : méthode de traitement pour le cheminement du produit courant selon le choix machine, elle est invoquée par le message reçu de l'agent produit.

6- La classe UML Données (information sur le produit)

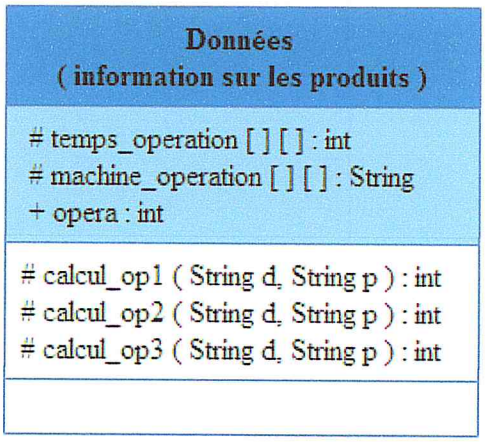


Figure 27 : La classe Données.

Description de la classe

- **Les attributs :**
  - temps_operation : matrice contenant tous les temps opératoires pour chaque type de produit et ses opérations.
  - machine_operation : matrice contenant le choix machine possible pour chaque type d'opération.
  - opera : temps d'exécution d'une machine.
- **Les méthodes** :
  - calcul_op1 : méthode permettant à l'agent AgProd qui doit choisir entre la machine 2 et la machine 3 de faire les calculs de temps d'exécution pour chaque machine selon la règle choisis.
  - calcul_op2 : méthode permettant à l'agent AgProd qui doit choisir entre la machine 3 et la machine 4 de faire les calculs de temps d'exécution pour chaque machine selon la règle choisis.
  - calcul_op3 : méthode permettant à l'agent AgProd qui doit choisir entre la machine 2 et la machine 4 de faire les calculs de temps d'exécution pour chaque machine selon la règle choisis.

7- La classe UML Regles (Règles des produits)

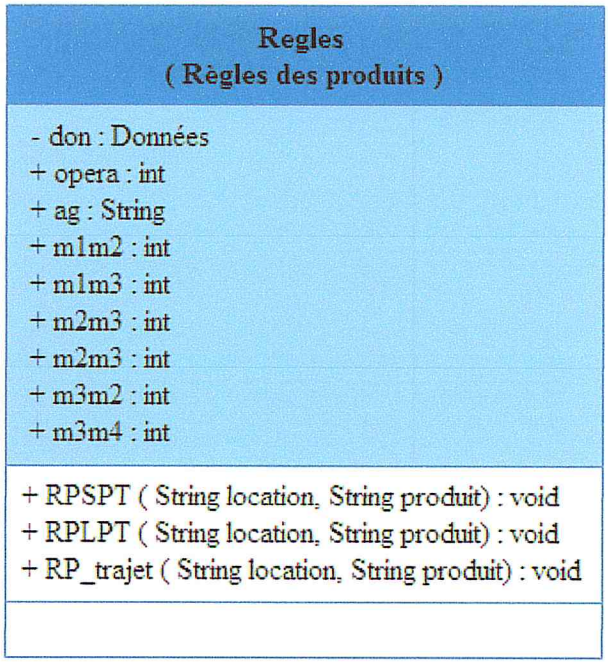


Figure 28 : La classe Regles.

Description de la classe

- **Les attributs :**
  - don : une instance de la classe "Données" qui nous permet de récupérer des informations sur les produits.
  - opera : temps d'exécution d'une machine.
  - ag : variable qui permet à l'agent AgProd de choisir une machine selon les calculs fais pour chaque règle produit.
  - m1m2, m1m3, m2m3, m2m4, m3m2, m3m4 : le temps de transport entre deux machines différents.
- **Les méthodes** :
  - RPSPT : une règle où la machine a réalisé un temps d'exécution plus court.
  - RPLPT : une règle où la machine a réalisé un temps d'exécution plus long.
  - RP_trajet : une règle pour réaliser la plus courte distance (location, destination).

6. Conclusion

Nous avons présenté dans ce chapitre la description et la spécification AUML et UML du système développé. La notation AUML a permis de mettre en évidences les différents aspects liés à la conception multi-agents du système.

Nous présentons dans le prochain chapitre l'implémentation et la validation du système.

Chapitre IV :

Réalisation et validation du système

1. Introduction

Nous consacrons ce chapitre au passage de la théorie à la pratique. Dans les sections qui vont suivre nous présentons l'implémentation du système.

Dans un premier temps nous abordons nos choix technologique qui nous ont permis d'élaborer notre système. Ensuite, Nous exposons le système d'une vue globale ainsi que les moyens de communication utilisés. Aussi, Nous précisons la réalisation de chaque couche notamment leur validation et enfin, nous passerons à la réalisation de la partie matérielle et nous présentons les résultats finaux.

2. Environnement de développement

Dans la réalisation de ce projet, nous avons employé un bon nombre de techniques (langages, environnements...etc.). Dans la section suivante, nous allons présenter ces différentes techniques.

2.1. NetBeans

NetBeans est un EDI (Environnement de Développement Intégré), un environnement libre et facile à utiliser. Il supporte plusieurs langages de programmation à savoir JAVA, C++, PHP, et bien d'autres. Il fournit plusieurs outils tels qu'un éditeur de texte doté d'un pré-compilateur avancé, un gestionnaire de projets. Ainsi que des outils de débogage et de test des programmes. C'est un outil qui facilite énormément la phase de développement et de tests.

2.2. Intégration de JADE dans NetBeans

Nous avons utilisé NetBeans car il permet aussi d'intégrer des bibliothèques Java en toute facilité. Ces bibliothèques sont des éléments d'outil de développement Java constituées de procédures et d'objets prédéfinis, utilisables de façon modulaire, et qui permettent le développement des applications Java par la réutilisation de codes existant. Dans notre cas, nous avons utilisé une bibliothèque qui regroupe l'ensemble des classes de JADE.

Nous avons utilisé la version JADE 4.4.0 sur NetBeans.

2.3. Android Studio

"Eclipse" était l'outil de développement le plus utilisé pour développer des applications Android jusqu'à mai 2013. Cet outil était interfacé avec le plug-in « ADT » (Android Development Tool) pour permettre un tel développement. Ce plug-in était disponible aussi pour les outils de développement « Netbeans » et « Intellij ». Google a annoncé, le 16 mai 2013, lors de la conférence « Google I/O », un nouvel outil de développement pour des applications Android, « Android Studio ». Android Studio est basé sur la version open-source (Community Edition) de « Intellij IDEA » développée par « JetBrains ». Avec le lancement d'Android Studio, Google a annoncé qu'elle cessait de mettre à jour le plug-in ADT utilisé dans Eclipse. Elle recommandait l'utilisation d'Android Studio.

De ce fait, nous avons utilisé Android Studio pour la programmation Android.

2.4. FlexSim

Nous avons déjà mentionné dans les chapitres précédents que nous avons utilisé FlexSim pour la simulation de notre système, notre choix a été basé sur le fait que FlexSim offre une vue 3D ainsi qu'une programmation facile de chaque objet du système. Tous les objets de la simulation possèdent des variables paramétrées opérationnelles pouvant être modifiées, et l'accès au menu d'édition se fait en cliquant sur l'objet. La fenêtre de propriétés "properties" possède plusieurs onglets reliés à différentes fonctions comme l'illustre la Figure 29.

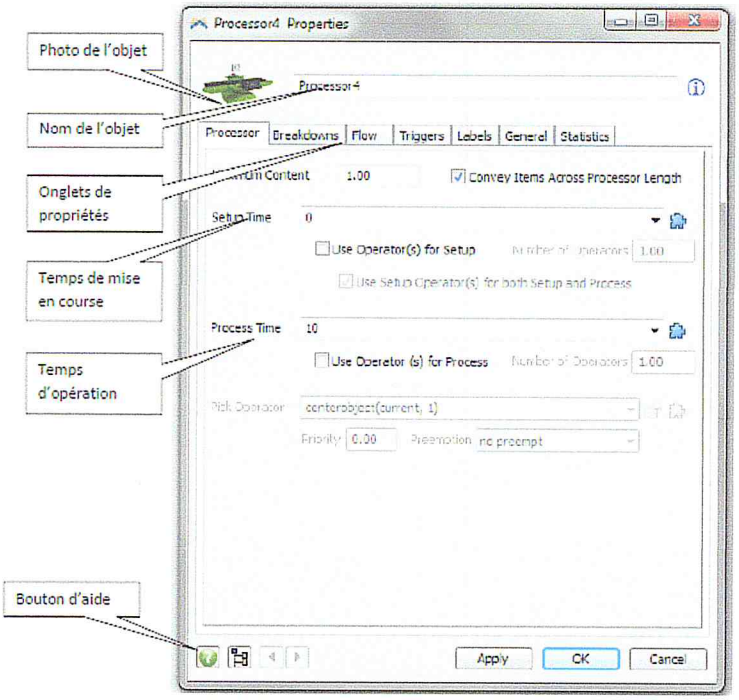


Figure 29 : L'accès aux propriétés de l'objet.

La Figure 30 montre que pour améliorer le comportement de l'objet lors de la simulation, nous pouvons ajouter du code FlexScript aux fonctions optionnelles (telle que en entrée, ou à la réception d'un message) à partir de l'onglet Triggers.

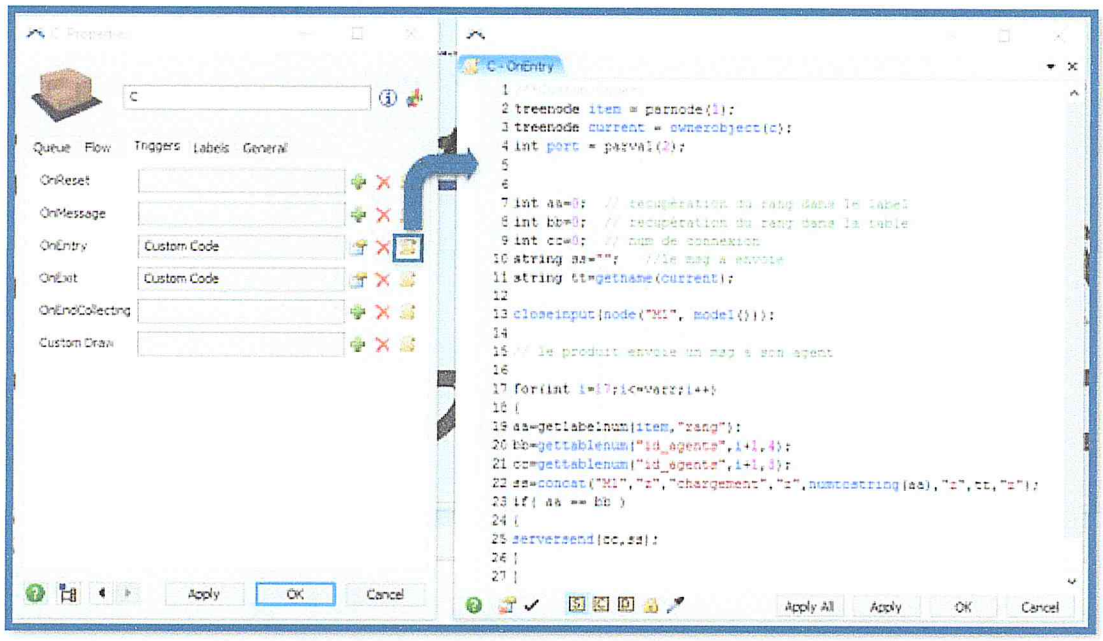


Figure 30 : Modification du code de la fonction OnEntry via l'onglet Triggers.

Plusieurs autres fonctions peuvent être modifiées afin d'adapter la simulation aux besoins du modélisateur.

3. Le protocole de communication

La communication entre le système de simulation et le système de pilotage présenté dans le chapitre précédent, se fait via un système de socket avec le protocole TCP/IP comme l'illustre la Figure 31.

Dans les domaines de la télécommunication et des réseaux informatiques, il existe de nombreux protocoles qui nous permettent la communication entre deux systèmes différents en établissant des règles et des normes assez spécifiques, nous avons choisi d'utiliser le protocole de communication le plus répandu et le plus utilisé dans le monde à savoir le protocole TCP/IP.

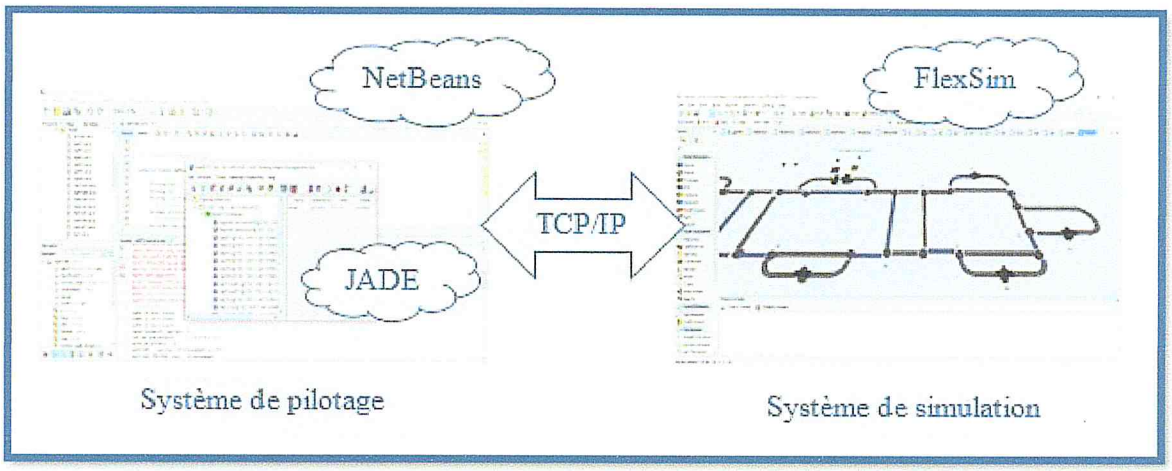


Figure 31 : Principe global de la communication entre les deux Systèmes.

3.1. Le protocole TCP/IP

Surnommé le moteur d'internet et des réseaux dans le monde entier, le protocole TCPI/IP est le protocole réseau le plus choisi au monde et cela grâce à sa simplicité et sa puissance.

- **Transmission Control Protocol (TCP) :** fiable, il se charge de la communication entre des applications, vérifie que le destinataire est prêt à recevoir les données et vérifie l'intégralité des informations envoyées à la réception.
- **Internet Protocol(IP)** : la couche IP offre la possibilité de transférer des données d'un hôte à un autre, nous citons comme exemple l'envoi d'un courrier électronique, chaque

message est un paquet de données qui s'accompagne de différentes informations : l'adresse de l'expéditeur, l'adresse du destinataire et les données supplémentaires.

Ces paquets transitent par des dizaines d'ordinateurs ou routeurs (un dispositif réseau qui route et oriente des paquets) afin de parvenir à leur destinations, pour cela, une adresse IP unique est attribuée à chaque ordinateur connecté au réseau.

3.2. Les sockets

Un socket est un point de terminaison d'une communication bidirectionnelle, c'est-à-dire entre un client et un serveur en cours d'exécution sur un réseau donné. Les deux sont liés par un même numéro de port TCP de sorte que la couche puisse identifier la demande de partage de données. Un serveur fonctionne sur une machine bien définie et est lié à un numéro de port spécifique. Le serveur se met simplement à l'écoute d'un client, qui demande une connexion [26].

Comme nous travaillons sur deux systèmes et que nous employons le protocole de communication TCP/IP, nous avons intégré des sockets à nos programmes client-serveur.

Dans notre cas le système de simulation est le serveur, les sockets intégrés au système sont programmé en FlexScript et les agents du système de pilotage sont des clients, de même les sockets intégrés au système sont programmé en JAVA.

4. Le système de simulation et le système de pilotage

Dans cette section nous illustrons le lancement et l'interaction entre le système de simulation (la couche de simulation HIL) et le système de pilotage (la couche de pilotage MA).

Chaque entité du système de simulation produit, machine et queue (point de routage) est associées à son propre agent, le nombre d'agents machine et d'agents routage est respectivement fixe pour les entités machines et queues (point de routage). Pour le nombre d'agent produit, il dépend de la quantité du produit pour chaque type B, E, L, T, A, I et P décrit dans le chapitre précédent.

La Figure 32 montre le lancement des deux systèmes et l'interaction de chaque agent avec son entité selon la décision du produit (le produit choisit une machine pour une opération donné, etc.).

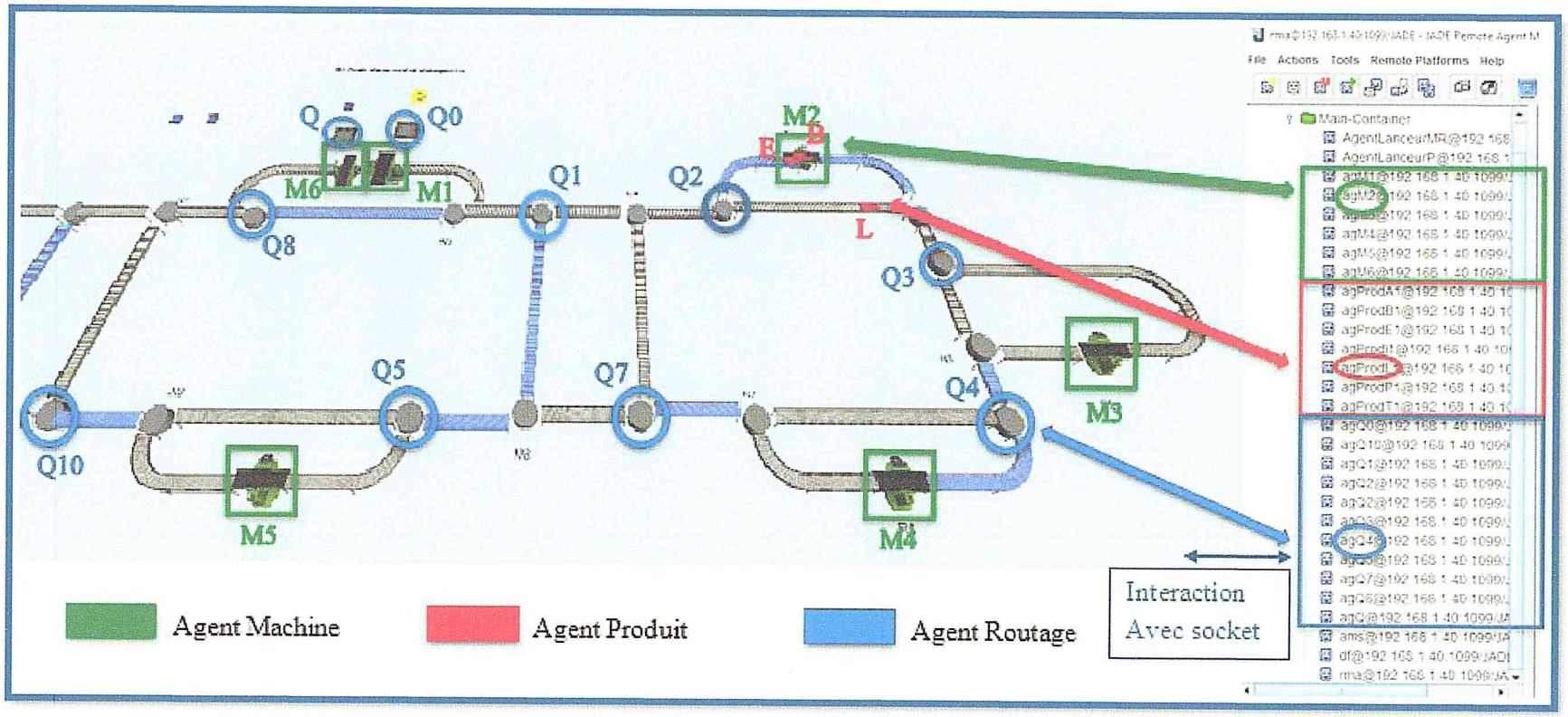


Figure 32 : Interaction entre les deux systèmes.

Dans le cas des produits, leur nombre est donné avant le lancement des deux systèmes par l'utilisateur sur le logiciel de simulation FlexSim comme l'illustre la Figure 33.

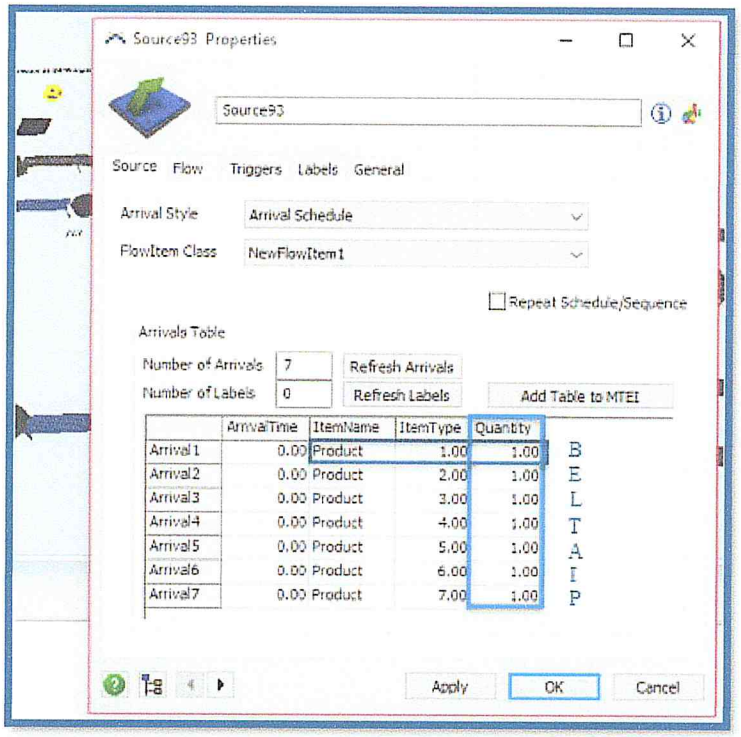


Figure 33 : L'utilisateur donne le nombre de produit.

De ce fait, prenons l'exemple de B=2, E=3, L=0, T=0, A=0, I=0 et P=0 nous aurons des agents lancés sur le système de pilotage qui portent respectivement les noms suivant : agProdB1, agProdB2, agProdE1, agProdE2 et agProdE3, à noter que les agents associés aux produits dont la quantité est nulle ne seront pas lancés.

Le Tableau 3 détaille toutes les entités qui existent sur le système de simulation et leurs agents coopérants.

Machine Physique	Agent
M1	agM1
M2	agM2
M3	agM3
M4	agM4
M5	agM5
M6	agM6

Produit Physique	Agent
B	agProdB1
E	agProdE1
L	agProdL1
T	agProdT1
A	agProdA1
I	agProdI1
P	agProdP1

Routage Physique	Agent
Q0	agQ0
Q1	agQ1
Q2	agQ2
Q3	agQ3
Q4	agQ4
Q7	agQ7
Q5	agQ5
Q10	agQ10
Q8	agQ8
Q	agQ

Tableau 2 : Assignations des entités s à leurs agents.

4.1. L'implémentation des règles de répartition

Quatre règles de répartition des priorités sont implémentées pour chaque machine (SPT, LPT, FIFO et LIFO) et cinq règles de répartition des priorités sont également implémentées pour chaque produit (RP1, SPT, RPLPT, RP2 et RP-trajet).

N° règle répartition	Nom règle répartition	Description
1	SPT	Un produit qui possède un temps d'exécution plus court.
2	LPT	Un produit qui possède un temps d'exécution plus long.
3	FIFO	Premier entré premier sorti.
4	LIFO	Dernier entré premier sortie.

Tableau 3 : Table des règles de répartition de machines.

N° règle répartition	Nom règle répartition	Description
1	RP1	La machine moins occupé avec notamment la distance plus courte.
2	RPSPT	Où la machine a réalisé un temps d'exécution plus court.
3	RPLPT	Où la machine a réalisé un temps d'exécution plus long.
4	RP2	La machine moins occupé.
5	RP-trajet	Réaliser la plus courte distance (location, destination)

Tableau 4 : Table des règles de répartition de produits.

Dans notre travail, nous avons implémenté une règle de priorité FIFO pour les machines pour l'ordonnancement des produits, et les règles d'affection RPSTP, RPLPT et RP-trajet pour les produits pour leur choix de machines.

Pour bien montrer l'interaction entre les deux systèmes, nous présentons le scénario détaillé des trois produits B, E, L :

- Donner le nombre de produit pour chaque type ici : B=1, E=1, L=1, T=0, A=0, I=0 et P=0, sur le système de simulation. la Figure 34 montre la réception des quantités de chaque produit de la part de FlexSim.

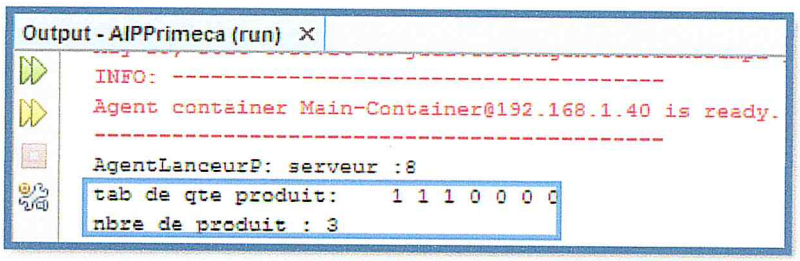


Figure 34 : Le message reçu du FlexSim contenant la quantité des produits.

- Lancement du système de simulation (serveur) et du système de pilotage dont les agents (clients) sont : LanceurMR, LanceurP, agM1, agM2, agM3, agM4, agM5, agM6, agQ0, agQ1, agQ2, agQ3, agQ4, agQ7, agQ5, agQ10, agQ8, agQ, agProdB1, agProdE1 et agProdL1.
- Le système de simulation reste en attente des connexions de tous les agents du système de pilotage.
- Une fois les connexions sont acceptées, B, E et L sont créés sur le système de simulation.
- A la création du B, E et L, les identifiants sont assignés ; avec ce dernier nous faisons la correspondance avec agProdB1, agProdE1 et agProdL1.
- Avant que les produits commencent l'exécution on doit d'abord lui associé une règle d'affectation, dans notre cas c'est la règle RPSPT (Short Processing Time) pour le produit B, RP-trajet (le plus court temps de transport entre les machines) pour le produit E et RPLPT (Long Processing Time) pour le produit L, comme l'illustre dans la Figure 35.

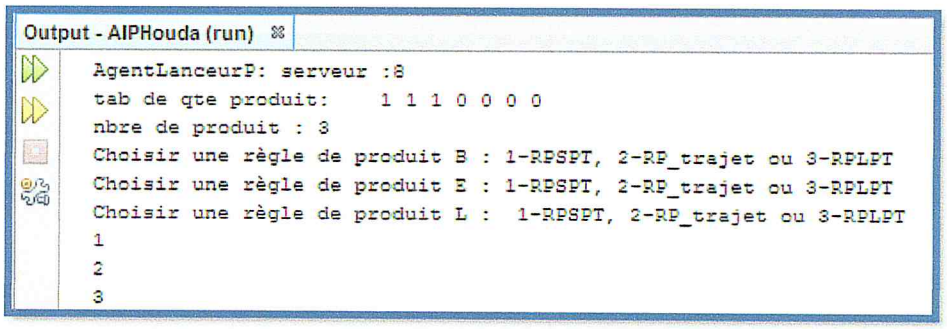


Figure 35 : Le choix des règles produits.

- A l'entrée de Q0, B envoie un message à agProdB1 pour l'informer de l'opération à exécuter "Chargement", la même chose pour les produits E et L.
- agProdB1, agProdE1 et agProdL1 reçoivent le message et font le traitement correspondant, duquel ils récupèrent le temps d'exécution de l'opération, puis ils envoient à agM1 l'information récupérée.
- agM1 envoie à son tour le message à M1, en suivant un séquencement en appliquant la règle de priorité FIFO.
- A la sortie de Q0, B envoie un message à agProdB1 pour l'informer du routage à faire, la même chose pour les produits E et L.
- agProdB1, agProdE1 et agPordL1 reçoivent le message et font le traitement correspondant, duquel ils récupèrent le nom de la machine ou ils exécuteront l'opération courante, puis ils envoient à agQ0 l'information récupérée.
- agQ0 reçoit à son tour le message, selon le nom de la machine et de son emplacement dans le système de simulation, agQ0 envoie un message à Q0 pour lui indiquer le cheminement correspondant.
- A l'entrée de M1, celle-ci récupère le temps d'exécution de l'opération "Chargement" de B, puis elle exécute le traitement dans 10s, la même chose pour les produits E et L.
- A la sortie de M1, B envoie un message à agProdB1 pour l'informer des opérations suivantes "Montage_Axe, Montage_R", E envoie un message à agProdE1 pour l'informer des opérations suivantes "Montage_Axe, Montage_R, Montage_L", et L envoie un message à agProdL1 pour l'informer d'opération suivante "Montage_Axe".

- agProdB1, agProdE1et agProdL1 reçoivent le message et font le traitement correspondant, puis ils récupèrent les noms des machines possible où ils peuvent exécuter leurs opérations. agProdB1, agProdE1 et agProdL1 font un choix de machine entre M2 et M3 en appliquant leur règle d'affectation. Après agProdB1, agProdE1 et agProdL1 envoient un message contenant le temps d'exécution respectivement 100s, 120s, 60s des opérations à l'agent machine choisi respectivement M2, M2, M3, qui à leurs tour envoient l'information reçue à la machine choisie.
- Si la file d'attente (le convoyeur avant l'entrée d'une machine) est pleine c'est-à-dire elle est déjà chargée par un autre produit, le produit tournera dans la cellule en suivant le cheminement (donné à chaque point de routage par les agents routage) jusqu'à ce que la file d'attente soit libre comme l'illustre la Figure 36.

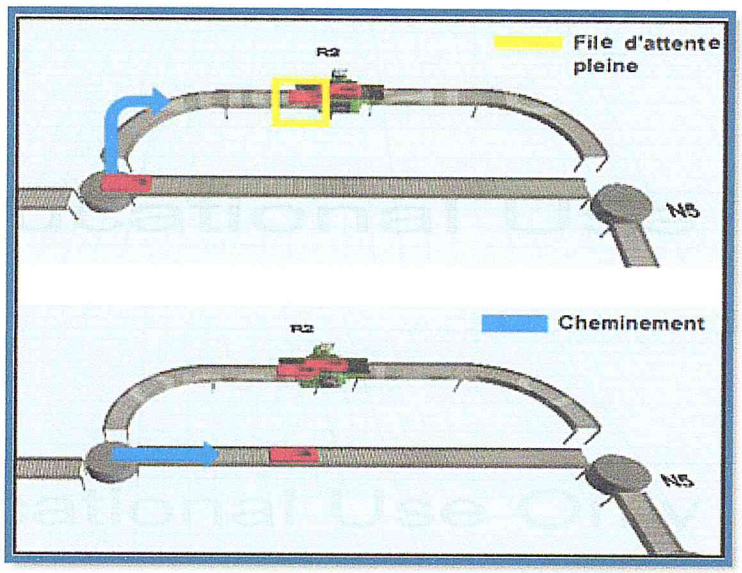


Figure 36 : La file d'attente.

- De même les produits ils vont choisir la machine M4 "Montage_I et Montage_Vis" respectivement 40s, 0s, 80s.
- A la fin les produits ils vont choisir la machine M5 "Inspection" 5s, puis la machine M6 "Déchargement" 10s.

Pour bien expliquer les échanges des messages entre les agents on va utiliser l'agent Sniffer parce que chaque message partant ou allant vers les différents agents est capté et affiché sur son interface. L'utilisateur peut voir et enregistrer tous les messages, pour éventuellement les analyser.

La Figure 37 montre la communication entre les agents durant un exemple d'exécution de notre plateforme. Cette figure est extraite à partir de l'outil "agent sniffer" de la plateforme JADE.

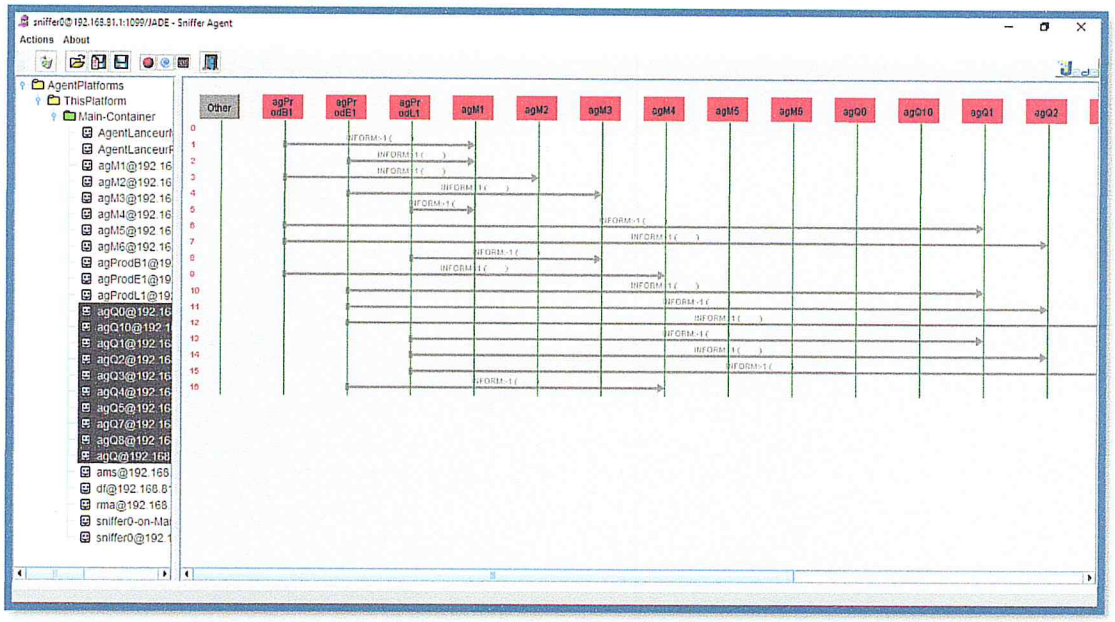


Figure 37 : L'échange des messages en utilisant l'agent Sniffer.

- La flèche 1 et 2 : agProdB1 et agProdE1 envoie un message à agM1 de type INFORM qui contient le nom de l'opération, le temps d'exécution de l'opération et l'identifiant du produit, pour l'opération "Chargement".
- La flèche 3 et 4 : agProdB1 envoie un message à agM2 de type INFORM après le choix de la règle produit pour faire les opérations "Montage_Axe, Montage_R" de même agProdE1 envoie un message à agM2 de type INFORM pour faire l'opération "Montage_Axe".
- La flèche 5 : agProdL1 envoie un message à agM1 de type INFORM qui contient le nom de l'opération, le temps d'exécution de l'opération et l'identifiant du produit, pour l'opération "Chargement".

- Les flèches 6 et 7 : agProdB1 envoie un message à agQ1 et agQ2 de type INFORM qui contient la localisation courante du produit, le nom de la machine dans laquelle la prochaine opération s'exécutera, respectivement pour le routage "routage1" et "routage2".
- Les flèches 8 : agProdL1 envoie un message à agM3 de type INFORM après le choix de la règle produit pour faire l'opération "Montage_Axe".
- Les flèches 9 : agProdB1 envoie un message à agM4 de type INFORM pour faire l'opération "Montage_I, Montage_Vis".
- De même les autre agents produit envoient des messages aux agents machine pour l'exécution des autre opérations et des messages aux agents routage pour l'information de leur cheminement jusqu'à ce que tous les produits arrivent à l'opération "Déchargement".

5. Le système de pilotage

Le système de pilotage est modélisé sur NetBeans, Dans cette section nous montrons la réalisation de ce dernier.

5.1. Implémentation de socket dans un agent

La classe java.net.socket peut être invoquée dans JADE (le même principe d'implémentation de socket), le numéro de port utilisé pour la connectivité entre le client et le serveur doit être supérieur à 1023, car les ports compris entre 0-1023 sont réservé.

Le fragment de code suivant montre comment la connexion est établie pour l'agent agProdB :

```
try {
socket = new Socket(InetAddress.getLocalHost(), 2017);
InputStreamReader(socket.getInputStream()));
out = new PrintWriter(socket.getOutputStream());
 }
catch (UnknownHostException e) {
 System.out.println("Impossible de se connecter à l'adresse " + socket.getLocalAddress());
} catch (IOException e) {
 System.out.println("Aucun serveur à l'écoute du port " + socket.getLocalPort());
}
```

Le fragment de code suivant montre la procédure d'envoi pour l'agent agProdB, cette procédure est invoquée à chaque fois que l'agent envoie un message au serveur :

```
public void SendSocket(String s) {
out.println(s);
out.flush();
 }
```

L'implémentation de la réception est réalisée dans un Thread à part, chaque agent à sa propre réception. Ce concept de Thread est utilisé pour séparer entre la réception du serveur et la réception entre agents pour éviter le problème de blocage des agents. Afin de mieux expliquer nous donnons le fragment de code qui suit :

```
//reception
Thread th1 = new Thread(new Runnable() {
public void run() {
while (true) {
try {
msg = receiveMsg();
if (msg != null) {
bool = true;
.....
}
} catch (IOException ex) {
System.err.println("la reception se fait pas " + getLocalName());
}}} });
```

5.2. Le fonctionnement Multi-Agents

Comme les agents sont développés sous la plateforme JADE, les agents de notre système sont des classes qui étendent la classe jade.core.Agent.

Sous forme de classe Java, nos agents contiennent plusieurs méthodes les caractérisant, entre autres la méthode *Setup ()* obligatoire pour l'initialisation de l'agent.

Afin d'implémenter les comportements de nos agents, nous avons défini des objets de la classe Behaviour jade.core.behaviours.Behaviour. Chaque objet de ce type dispose d'une méthode *action ()* qui constitue le traitement à effectuer par celui-ci. De plus, ces comportements sont classés en deux catégories : OneShotBehaviours et CyclicBehaviours.

- **CyclicBehaviours** : Comme son nom l'indique, il exécute sa tâche d'une manière répétitive [27].
- **OneShotBehaviours** : Il a la particularité d'exécuter sa tâche une et une seule fois puis il se termine [27].

Afin de mieux comprendre ces notions, nous donnons le fragment de code qui suit :
Voici la procédure d'envoi qui implémente le OneShotBehaviour dans lequel les agents échangent des messages entre eux.

```
public void Send(final String s, final String nom) {
 addBehaviour(new OneShotBehaviour() {
public void action() {
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
msg.addReceiver(new AID(nom, AID.ISLOCALNAME));
msg.setContent(s);
send(msg);
} }) ; }
```

Voici le fragment de code de réception qui implémente le CyclicBehaviour dans lequel les agents reçoivent les messages.

```
addBehaviour(new CyclicBehaviour(this) {
public void action() {
final MessageTemplate msgTemplate =
MessageTemplate.MatchPerformative(ACLMessage.INFORM);
final ACLMessage messageRecu = receive(msgTemplate);
// on vérifie si le message est n'est pas null et on récupère le contenu
if (messageRecu != null) {
try {
s = messageRecu.getContent();
 ....
} catch (Exception e)
 } } } });
```

Le Thread de l'agent *setup ()* est plus rapide que le Thread de réception du serveur, par conséquent nous avons utilisé un autre CyclicBehaviour et une variable booléenne qui déclenche ce Behaviour lors de la réception du message reçu du serveur. Malgré ça, le

message ne se partage pas d'où l'utilité d'une variable *volatile,* elle précise que la variable peut être changée de manière asynchrone.

Afin de mieux comprendre nous donnons le fragment de code qui suit :

```
addBehaviour(new CyclicBehaviour(this) {
public void action() {
if (tmp != null) {   // variable volatile
if (bool == true) {   // variable boolean
.....
bool = false;
}}}});
```

6. Implémentation du produit cyber-physiques

La première étape de notre travail dans cette partie est de se familiariser avec l'environnement Android.

6.1. Android

Android est un système d'exploitation mobile basé sur le noyau Linux et développé actuellement par Google. Le système a d'abord été conçu pour les smartphones et tablettes tactiles, puis s'est diversifié dans les objets connectés et ordinateurs comme les télévisions (Android TV), les voitures (Android Auto), les ordinateurs (Android-x86) et les smartwatch (Android Wear). Le système a été lancé en juin 2007 à la suite du rachat par Googleen 2005 de la startup du même nom. En 2015, Android est le système d'exploitation le plus utilisé dans le monde avec plus de 80 % de parts de marché dans les smartphones.

Notre travail est de faire une interface d'un agent produit qui connecte à la plateforme JADE, c'est-à-dire il prend le comportement de l'agent produit.

7. Validation de la simulation Hardware In The Loop

L'agent produit E exécute le même traitement que d'autres agents Produit, alors le scénario présenté dans la première validation est complété par une validation de l'approche Hardware In The Loop.

L'utilisateur saisie les configurations (l'adresse où se trouve la plateforme JADE, le numéro de port, le nom du produit, ainsi que le numéro de port de connexion au serveur) dans l'écran de démarrage de l'application comme l'illustre la Figure 38. Une fois qu'il appuie sur le bouton de "Charger Produit", un split container est créé avec l'agent produit à l'intérieur de la plateforme JADE.

Figure 38: La création de l'agent dans un split Container.

8. Conclusion

Nous avons présentés dans ce chapitre la partie implémentation et mise en œuvre de notre système complet qui se constitue de système de pilotage, ainsi que l'intégration de l'approche "Hardware in the Loop Simulation".

Pour ce faire, nous avons adopté une validation en deux temps. En premier lieu, nous avons validé le fonctionnement global du système par la mise en évidence des différentes interactions et décision qui interviennent dans ce dernier. Dans un deuxième temps, nous avons validé l'approche "Hardware in the Loop Simulation" en intégrant un composant embarqué un téléphone.

Conclusion Générale

Le travail présenté dans ce mémoire tourne autour de la mise en œuvre des systèmes cyber-physiques à base de systèmes multi-agents en intégrant le système de développement Android, pour la connexion aux ressources de production.

Deux travaux complémentaires ont été réalisés en vue d'atteindre cet objectif :

Le premier, est d'implémenter une architecture multi-agent (JADE) et des protocoles et approches de coordination pour le pilotage d'une cellule de production flexible en développant un ordonnancement par rapport aux règles de priorité pour les machines et aux règles d'affectation pour les produits.

Le deuxième, est d'intégrer une partie matérielle entre ces deux systèmes pour rendre le produit physique intelligent à l'aide un téléphone Android pour la connexion aux ressources de production.

Bien que le travail réalisé reste à améliorer car il vient d'être entamé au sein du laboratoire de l'équipe SRP, il nous a permis d'enrichir nos connaissances théoriques et pratiques dans le domaine d'électronique, de programmation Android. A l'issue du travail présenté dans ce mémoire, différentes pistes forment les directions futures à cette recherche. Notamment :

- Intégrer un automate industriel programmable pour le routage des produits afin de mieux valider l'approche "Hardware In The Loop Simulation", et ceci en utilisant le serveur OPC qui sert comme interface entre les systèmes multi-agents, FlexSim et l'automate.
- Améliorer l'interface de l'application mobile du produit physique et offrir plus de fonctionnalités pour l'utilisateur. Par exemple : l'utilisation de plusieurs smart phone dont chaque téléphone représente un agent de la plateforme JADE.
- Augmenter l'intelligence du produit physique en ajoutant un lecteur RFID, ceci en utilisant la carte Arduino et le protocole industriel MODBUS qui permet la communication entre appareils en mode maître-esclave, la carte Arduino jouera le rôle de maître et le lecteur RFID le rôle d'esclave pour la lecture des étiquettes "TAG".

Cette technologie nous permet d'accroitre les possibilités en termes de traçabilité avec l'identification, donc optimiser le pilotage du flux physique.

Bibliographie

[1] O. R, Comprendre l'industrie 4.0 et ses défis, Lycée Heinrich-Nessel de Haguenau: In Industrie 4.0, 2015.

[2] *L'observatoire des tendances 2014 de ni,* 2014.

[3] «ADDI-DATA,» France , 2015. [En ligne]. Available: http://www.addi-data.fr/les-systemes-cyber-physiques/.

[4] Z. Alessandro, «developerWorks,» 20 April 2015. [En ligne]. Available: https://www.ibm.com/developerworks/library/ba-cyber-physical-systems-and-smart-cities-iot/.

[5] K. Richard, «IT Industrie et Technologies,» 03 Juin 2014. [En ligne]. Available: http://www.industrie-techno.com/national-instruments-le-defi-de-la-conception-cyber-physique.30379.

[6] «Challenges,» 05 Mars 2013. [En ligne]. Available: http://www.challenges.fr/high-tech/20130305.CHA6937/demain-des-usines-pilotees-par-internet.html.

[7] «L'usine Digitale,» 7 Mars 2013. [En ligne]. Available: http://www.usine-digitale.fr/article/l-usine-du-futur-du-plan-allemand-industrie-4-0-s-esquisse-au-cebit.N192937.

[8] F. J, Les Systèmes multi-agents : vers une intelligence collective, Paris: InterEditions, 1995.

[9] R. J. Nicholas, S. Katia et W. Michael, «A roadmap of agent research and development,» *Autonomous agents and multi-agent,* vol. 1, n° %11, p. 7–38, 1998.

[10] H. G et M. T, Intelligence artificielle distribuée systèmes multiagents, Université evry val d'essone, France: Technical report, 2008.

[11] M. F. A, Agents et systèmes multi-agents, Université Politechnica de Bucarest.

[12] M. H, Ingénierie des protocoles d'interaction : des Systèmes Distribués, Paris 9, 2001.

[13] B. B, J. Z, K. F et M. N, Dynamique de l'environnement, 2005.

[14] O. Boissier, Systèmes multi-agents Vue d'ensemble, 2010.

[15] S. J.P, «e-forme JADE,» 2005. [En ligne]. Available: https://archives.limsi.fr/Individu/jps/enseignement/examsma/2005/1.plateformes_1/jade.htm.

[16] «Technologies de l'information et de la communication et appropriation des savoirs,» Politechnica University of Bucharest, 2002. [En ligne]. Available: http://turing.cs.pub.ro/auf2/html/chapters/chapter6/chapter_6_5_1.html.

[17] A. FERGUEN, «Plate-forme JADE,» 2005. [En ligne]. Available: https://archives.limsi.fr/Individu/jps/enseignement/examsma/2005/1.plateformes_3/index-Ferguen.html.

[18] W. Damien, «AB Mécatronique,» 2016. [En ligne]. Available: http://www.abmecatronique.com/la-simulation-hardware-in-the-loop-hil_540611/.

[19] X. Ye, Modélisation et simulation des systèmes de production : une approche orientée-objets, 1994.

[20] B. Brahim, modélisation & simulation sur ordinateur., 2004.

[21] C. Naoufel, *La simulation à événements discrets.*.

[22] C. Schaller, La Simulation Comme Outil d'Aide à la Décision des Choix Stratégiques, Suisse: Ecole polytechnique fédérale de lausanne.

[23] N. William, «FlexSim simulation environment,» chez *Simulation Conference, 2003 proceedings of the 2003 Winter.*, 2003.

[24] P. Cyrille, ORCA : Architecture hybride pour le contrôle de la myopie dans le cadre du pilotage des Systèmes Flexibles de Production, Valenciennes, 2013.

[25] M. Bachir, «A Rule-Based Harmony Search Simulation-Optimization Approach for Intelligent Manufacturing Control .,» 2015.

[26] «OpenClassrooms,» 31 Octobre 2013. [En ligne]. Available: https://openclassrooms.com/courses/introduction-aux-sockets-1.

[27] M. Youssfi, Écrivain, *Systèmes Multi Agents.* [Performance]. Signaux Systèmes Distribués et Intelligence Artificielle (SSDIA) ENSET Mohammedia, Université Hassan II Casablanca, 2013.

[28] E. h. Hind, Approche méthodologique pour l'intégration des systèmes controlés par le produit dans un environnement de juste-à-temps: Application à l'entreprise Trane., 2008.

ANNEXE A

Les faiblesses d'UML pour la représentation des systèmes multi-agents ont conduit une équipe de chercheurs travaillant dans différentes entreprises ou universités (Siemens, University Paderborn, Intelligent Automation, Fujitsu…) à concevoir AUML. L'objectif est de mettre au point des sémantiques communes, des méta-modèles et une syntaxe générique pour les méthodologies agents. AUML est un des fruits de la coopération entre FIPA (Foundation of Intelligent Physical Agents) et l'OMG (Object Management Group).
Par rapport à UML, Agent-UML propose des extensions pour la représentation des agents que nous détaillerons dans cette annexe.

1. UML

UML, c'est l'acronyme anglais pour « Unified Modeling Language ». On le traduit par « Langage de modélisation unifié ». La notation UML est un langage visuel constitué d'un ensemble de schémas, appelés des diagrammes, qui donnent chacun une vision différente du projet à traiter. UML nous fournit donc des diagrammes pour représenter le logiciel à développer : son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel, etc.
UML prend en charge plusieurs types de modèles :

- **les cas d'utilisation** : la spécification des actions que le système ou la classe peut réaliser en interaction avec les acteurs extérieurs. Ils sont souvent utilisés pour décrire comment un utilisateur communique avec son logiciel.
- **les modèles statiques** : ils décrivent la sémantique statique des données et des messages d'une manière conceptuelle et d'une manière plus proche de l'implémentation (il s'agit des diagrammes de classes et de packages).

- **les modèles dynamiques** : ils incluent les diagrammes d'interaction (i.e., les diagrammes de séquence et les diagrammes de collaboration), les schémas d'état et les diagrammes d'activité.
- **les modèles d'implémentation** : ils décrivent la répartition des composants sur différentes plateformes (i.e., les modèles de composants et les diagrammes de déploiement).

2. Déficiences d'UML

L'idée générale d'AUML est de combler les déficiences d'UML pour la modélisation des systèmes à agents.

Parmi ces déficiences, on trouve :

- Des relations entre classes statiques (agrégation, généralisation, et association) mais qui semblent tout de même adéquats. Il est possible d'utiliser des associations de classes et des stéréotypes pour étendre UML avec des relations spécifiques pour les agents.
- Les accointances sont des relations importantes entre agents. Il s'agit d'une relation dynamique entre des instances et UML n'est pas très adapté pour les représenter.
- Un certain nombre de concepts de haut niveau (comme les engagements, les contrats, etc.) peuvent être relativement bien représentés avec UML mais d'autres (comme les croyances et les intentions) ne le peuvent pas.
- Il est difficile de représenter l'état interne des agents. Il faudrait un modèle proposant des concepts de haut niveau « cognitif » (BDI, BC, GAP, …).
- UML n'est pas efficace pour représenter des connaissances fonctionnelles (buts, planification, process, etc.). Pourtant beaucoup de méthodologies agents utilisent les buts et la décomposition de buts en sous buts.

Il n'est pas évident que les approches de modélisation à états finis soient adaptées pour les agents. Les agents ont des espaces d'états vastes qu'il n'est pas évident de partitionner en un plus petit nombre de macro-états de plus haut niveau. Les agents peuvent apprendre et s'adapter à différentes choses et des paramètres comme les croyances interagissent pour influencer le comportement de façons subtiles. Ces systèmes sont dynamiques, non linéaires et ont un comportement émergeant.

3. AUML

AUML est basé sur la méthode UML (Unified Modeling Language) qui est une méthode de génie logiciel utilisée pour les développements en langages orientés-objets. Elle est déjà largement utilisée par la communauté des concepteurs-objet et son succès continue de croître.

Comme nous l'avons déjà vu, par rapport aux objets, les agents ont des activités autonomes et des buts.

C'est cette différence qui entraîne l'insuffisance d'UML pour modéliser les agents et les systèmes multi-agents.

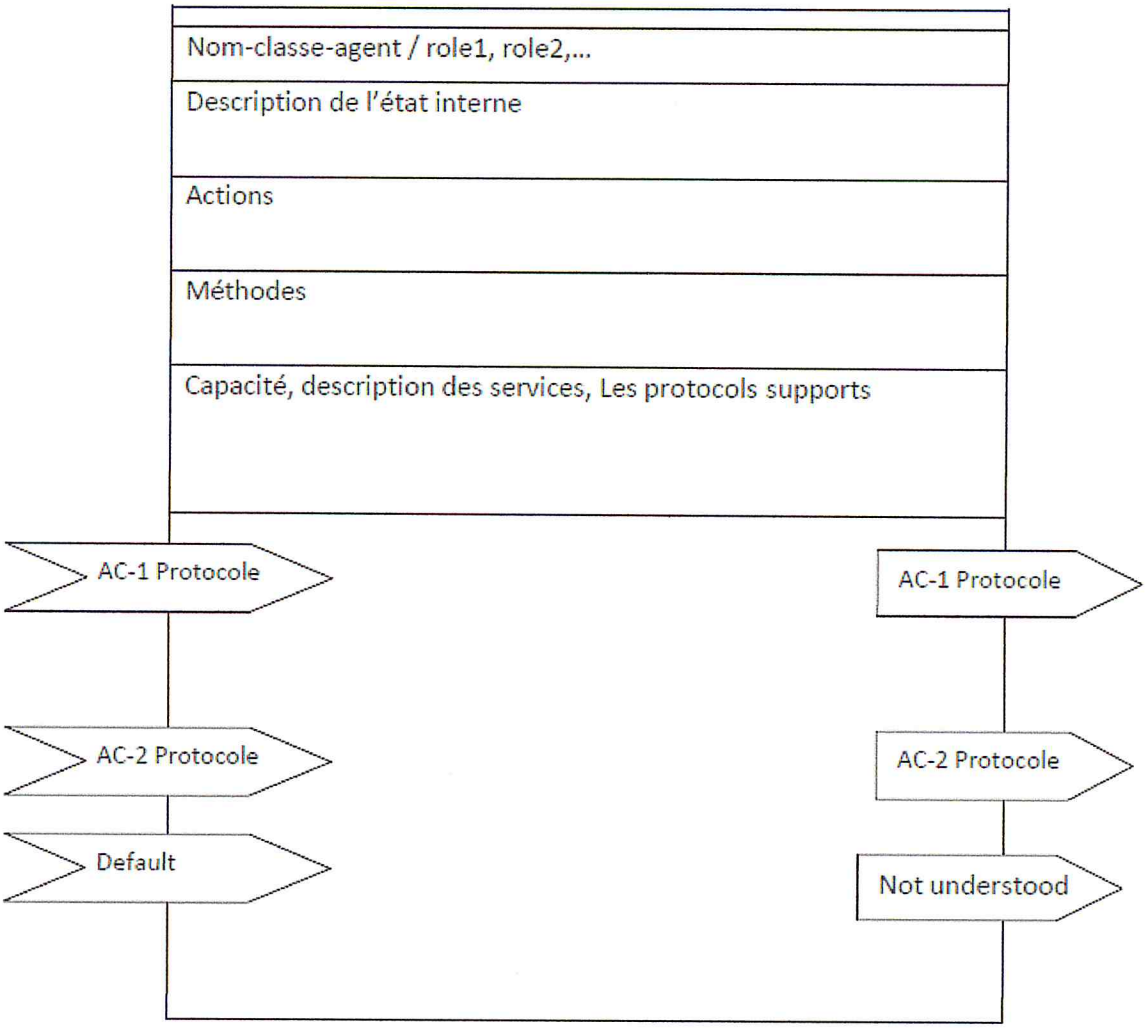
Aussi AUML remplace-t-il la notion de méthode par celle de service. Ses principales extensions sont :

- **Diagramme de classes** d'agent qui est une reformulation du diagramme de classes d'objets,
- **Diagramme de séquence** qui permet une meilleure modélisation des interactions entre agents,
- **Diagramme de collaboration** qui complémente le diagramme de séquences en proposant une autre lecture et vision des interactions entre agents.

4. Présentation générale des extensions d'AUML

4.1. Le diagramme de classes d'agent

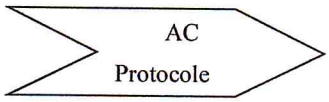
Un diagramme de classes agent doit supporter tous les concepts liés à l'agent, ce diagramme est illustré dans la figure suivante :



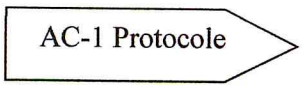
Les propriétés de ce diagramme sont :

- **Nom de la classe agent/ rôle1, rôle2,…:** Un agent d'une classe donnée peut avoir plusieurs rôles (un détaillant peut être acheteur ou vendeur).
- **Description des états :** Définition de variables d'instance qui reflètent l'état de l'agent,
- **Actions(Plans) :** deux types d'actions peuvent être spécifiés : action proactive exécutée par l'agent lui-même si une précondition devient vraie, et réactive résultant d'un message reçu d'un autre agent. En d'autres termes les actions sont les plans qu'a un agent.
- **Méthodes :** Elles sont définies comme dans UML, avec éventuellement des préconditions, post-conditions ou invariants.

- **Envoi et réception de messages :** La principale interface entre les agents et leur environnement est la réception et l'envoi de messages, ces messages peuvent être des classes ou des objets. Les messages sont représentés par :



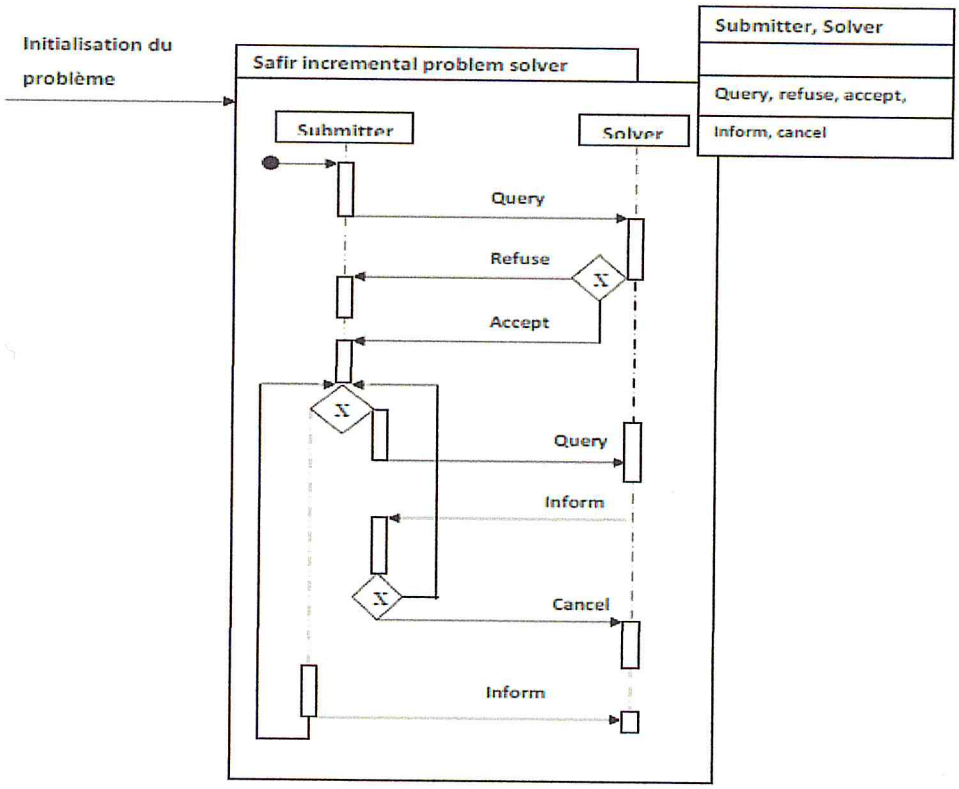
Pour les messages entrants



Pour les messages sortants

4.2. Protocoles d'interaction (AIP : Agent Interaction Protocol)

Exemple de représentation :



Dans le protocole de la figure ci-dessus, aucune précision n'est donnée sur le traitement ou la construction des messages :

- La construction de la requête (par Submitter) peut être un processus complexe décrit par un diagramme d'activités ;

- Le traitement de cette requête (par Solver) peut être décrit par un autre diagramme d'activités ou de séquence.

Le schéma ci-dessus est constitué de trois couches. Ce découpage en couches réifie les processus inter-agents et ceux internes à chaque agent : le protocole est la couche supérieure représentée sous forme d'un package ou d'un Template,

Diagramme de séquences :

Nommage des acteurs : nom agent/rôle : Class

Les messages échangés ne sont plus des noms de méthodes mais des actes de communications,

Utilisation des **threads d'interactions** concurrents : AND, OR, XOR

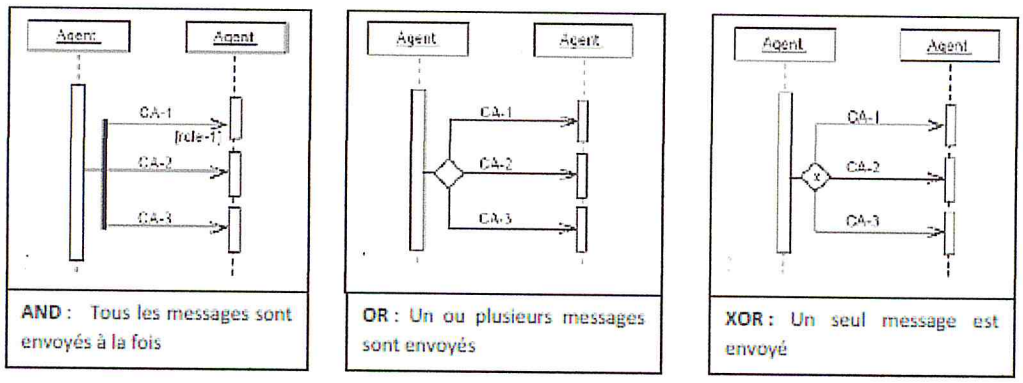
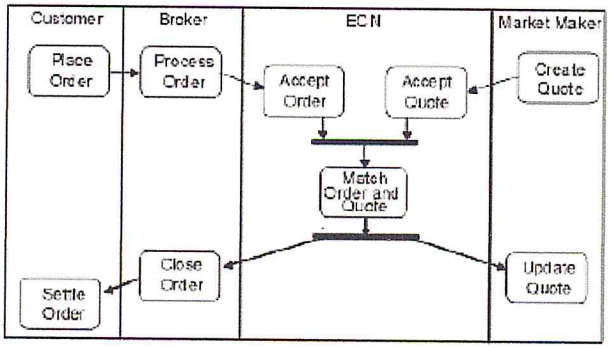


Diagramme d'activités :

Décrit les opérations entre agents et les événements qui les déclenchent.

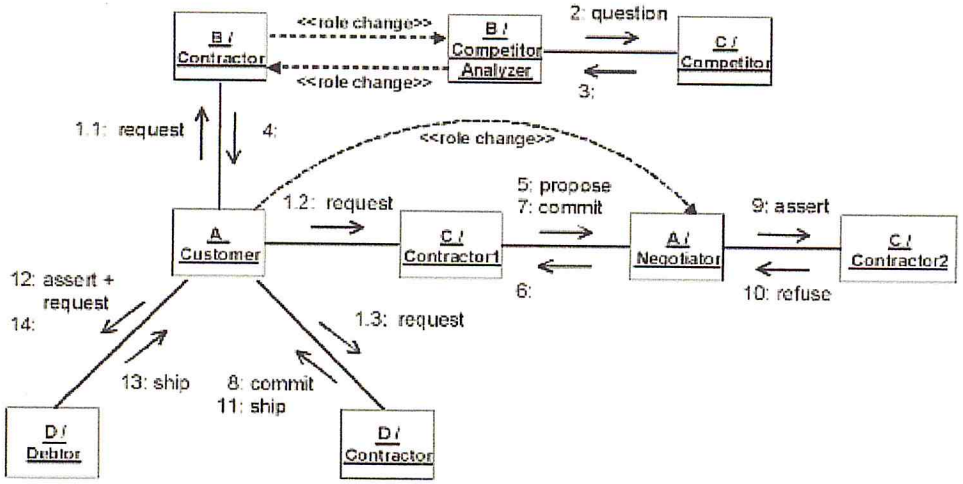


ECN : Electronic Commerce Network

Diagramme de collaboration :

Se constitue comme dans les modèles objets, il représente les interactions entre les agents.

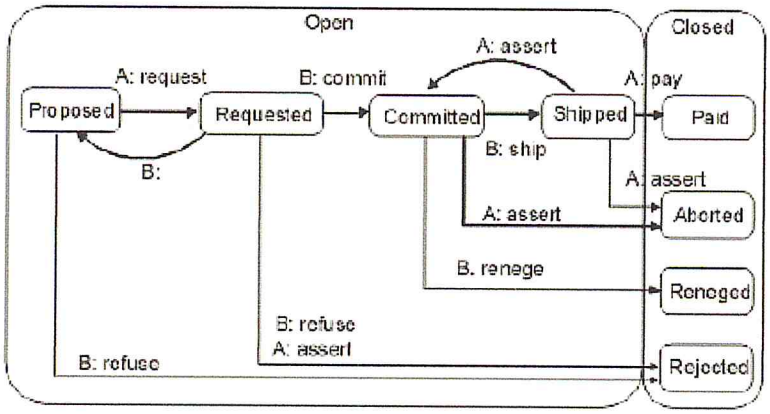
Voici un exemple :



On remarque que les agents sont placés n'importe où dans le diagramme, les messages échangés sont numérotés et par rapport au diagramme de séquence, les protocoles d'interactions sont moins clairs.

Diagramme d'états- transitions :

Décrit les changements d'état lors des échanges entre agents,



Les traitements internes de l'agent constituent la troisième couche :

Diagramme d'activité pour chaque agent,

Diagramme d'état transition pour chaque agent.

5. Limitations d'AUML

AUML présente les limitations ci-après :

- Les diagrammes sont désordonnés et peuvent être mal interprétés ;
- L'expression de toutes les informations nécessaires sur les protocoles peut les rendre illisibles ;
- Les cas de redondance sont difficiles à identifier et corriger ;
- La description des actions temporelles (telles que le timeout, deadline, ...) est difficile à exprimer
- La notion d'historique des échanges n'existe pas ;
- La terminaison des interactions n'est pas toujours spécifiée.

Aujourd'hui AUML n'est pas encore un langage de modélisation fini et adopté par la communauté car aucune spécification finale n'a été publiée officiellement mais plusieurs publications sur des extensions d'UML ont étés publiées par les membres du groupe.
Actuellement, leurs travaux se concentrent sur les interactions, plus spécifiquement sur les protocoles d'interaction, mais aussi sur d'autres aspects connexes comme les langages d'interaction, la coordination, les rôles des agents. D'autres travaux portent également sur les architectures, et les ontologies.