الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

Filière Électronique
Spécialité Instrumentation

présenté par

AMRAOUI Abdenour

&

HEBBOUCHE Amira

Pilotage d'une Plateforme d'Assemblage Robotisée par les Systèmes Multi Agents.

Proposé par : Mr. Abdelhamid Bendjelloul & Dr.Nesrine Amirouche

Année Universitaire 2019-2020

Remerciements

Nous tenons tout d'abord à remercier Dieu le tout miséricordieux d'avoir guidé nos pas vers les portes de savoir tout en illuminant notre chemin et nous avoir donné la patience et le courage pour mener ce cursus jusqu'à son terme.

Nous remercions en particulier Mr BENDJELLOUL Abdelhamid, pour l'honneur qu'il nous a fait de bien vouloir nous encadrer, et pour les conseils donnés lors de la réalisation de ce travail.

Nous souhaitons également remercier Mm AMIROUCHE Nesrine pour le temps qu'elle nous a accordé à diriger notre travail à travers ses orientations et aide précieuse et toutes les heures consacrées à assister notre travail.

Nous adressons nos remerciements aux membres de jury pour avoir accepté de nous prêter de leur attention et évaluer ce travail.

Nous tenons à remercier nos chers parents, frères, sœurs et proches qui ont toujours été là pour nous soutenir et encourager.

Dédicace 1

Je dédie ce modeste travail :

A mes chers parents pour leur soutien, leur patience, leur encouragement et leur sacrifice durant mon parcours scolaire

A mes sœurs et mes frères qui m'ont toujours encouragé tout au long de mes années d'étude

A tous les membres de ma famille

A tous mes ami(e)s et mes proches que j'aime beaucoup pour leur soutien inconditionnel et leurs conseils sincères.

A mon binôme Amira.

Abdenour...

Dédicace 2

A mes très chers parents qui m'ont toujours soutenue et encourager à tout moment, que dieu les protèges.

A ma chère sœur Kounouz pour sa patience, et son aide.

A mes très chers frères Issam et Mohamed pour leurs disponibilité et leurs aide précieuse.

A tous mes amis.

A mon binôme Abdenour.

Amira. . .

ملخص:

الغرض من هذا المشروع هو قيادة منصة تجمع روبوتية من خلال انظمة متعددة الوكلاء وهذا من خلال انشاء نظام الإدارة والاشراف الموزع متعدد الوكلاء وتصميم معمارية لنظام متعدد الوكلاء قادرة على تنفيذ خطة الجدولة باستعمال لغة الجافا.

كلمات المفاتيح: متعدد الوكلاء، قيادة نظام انتاج، منصة جاد.

Résumé :

Le but de ce projet consiste à piloter une plateforme d'assemblage robotisée par les systèmes multi-agents, les travaux réalisés en vue d'atteindre cet objectif sont la mise en place du système de pilotage et de supervision multi-agents distribué et la conception d'une architecture SMA capable d'exécuter un plan d'ordonnancement, en utilisant le JAVA.

Résumé : Mots clés : Multi-agent, pilotage de système de production, la plateforme JADE

Abstract :

The goal of this project is to drive a robotic assembly platform by multi-agent system, the work carried out to acheive this objective is the implementation of the distributed multi-agent management and supervision system, the design of an SMA architecture capable of executing a scheduling plan, all using JAVA language.

Keywords : Multi-agent, production system managment, JADE platform **:**

Listes des acronymes et abréviations

ACC :	Agent Communication Agent
ACL :	Agent Communication Language
AID:	Agent Identifier
AMS:	Agent Management System
API :	Application Programming Interface
DF:	Directory Facilitator
ERP:	Entreprise Resource planning
RFID:	radio frequency identification
FIPA:	Fondation For Intelligent Physical Agent
GUI:	graphycal User Interface
IA:	intelligence artificielle
IOT &IIOT:	internet of things
JADE :	Java Agent Developement framework
JAR :	Java Archive
JDK :	Java developement Kit
JRE :	Java Runtime Environement
JVM :	Java Virtuel Machine
J2SE :	Java 2 Standard Edition
OPC:	OLE for Process Control
OPC UA:	Open Platform Communication Unified Architecture
RMA:	Remote Monitoring Agent
SMA:	System Multi Agent
SOA:	service oriented architecture
TILAB:	Telecom Italia Laboratories
UA :	unified architecture

Tables des matières :

Liste des figures

Liste des tableaux

Introduction générale

Dans la société d'aujourd'hui, l'électronique intervient dans la vie quotidienne de chacun. On est souvent amené à être en relation avec des machines, de plus en plus intelligentes. Mais une chose est sure, on se voit mal vivre sans ces merveilles que l'Homme a inventées.

L'industrie fait partie des secteurs les plus en avance en matière de technologie. Produire plus et plus rapidement tout en réduisant les coûts est devenu une véritable priorité pour toute entreprise industrielle qui se veut prospère et compétitive, sur un marché où la concurrence est toujours plus forte. L'électronique offre de nombreuses solutions pour permettre aux industriels d'optimiser, d'automatiser et de mieux rentabiliser leurs chaînes de production.

Les systèmes de pilotage jouent un rôle fondamental dans la maitrise des activités de production et l'amélioration des performances au sein des systèmes quand l'environnement est dynamique et fortement perturbé. En effet, la globalisation des machines économiques, la concurrence féroce, la complexification des produits, leur durée de vie toujours plus courte, ainsi que des clients exigeants, sont le résultat de la rencontre du monde cyber numérique et des systèmes de production modernes. Caractérisé par une distribution de la décision, pour plus de flexibilité et d'agilité.

Dans ce contexte notre travail consiste à la mise en place d'un système de pilotage et de supervision multi agent distribué, la conception d'une architecture SMA capable d'exécuter un plan d'ordonnancement, le travail présenté dans ce mémoire a été élaboré au sein du Centre de Développement des Technologies Avancée.

L'organisation du mémoire est comme suit :

Dans le premier chapitre nous présentons l'état de l'art : les définitions et les concepts liés aux l'industrie 4.0.

Dans le deuxième chapitre, nous présentons la plateforme au niveau du labo SRP ainsi le fonctionnement général de la cellule et enfin nous exposons notre problématiques.

Dans le troisième chapitre, nous donnons la description de la solution et quelques détails techniques concernent les logiciels utilisés.

Dans le quatrième chapitre, nous exposons notre implémentation, d'une part Le simulateur de la plateforme conçu en JAVA à l'aide de l'éditeur Eclipse IDE, et d'autre part La création d'un système Multi Agents à l'aide de la plateforme JADE sous JAVA. Ainsi que l'exécution d'un plan d'ordonnancement par ce dernier sur le simulateur de la plateforme.

Nous conclurons par une discussion des résultats.

Chapitre 1 Généralités sur l'industrie4.0

1.1 Introduction

Ce chapitre présente les caractéristiques de l'industrie 4.0 qui font partie de notre travail qui s'articule autour de l'utilisation des systèmes Multi Agents pour le pilotage de la production. Qui fait partie de l'utilisation de l'intelligence artificielle en milieux de production qui est l'une des choses qui caractérisent l'industrie 4.0.

1.2 Historique des révolutions industrielles

1.2.1 La 1ère révolution industrielle

Elle remonte à l'exploitation du charbon et la mise au point de la machine à vapeur par James Watt en 1769. Cela va transformer radicalement le mode de fabrication. En effet**,** l'artisanat va être remplacé par la production mécanique, les usines vont se substituer aux manufactures et ateliers artisanaux**.** Dans les usines, la révolution correspond à l'utilisation de la machine à vapeur comme moteur pour actionner les machines permettant des cadences accrues**. [15]**

1.2.2 La 2ème révolution industrielle

La deuxième révolution industrielle est basée sur l'utilisation du pétrole et de l'électricité à la fin du 19ème siècle. Cela va permettre de moderniser les moyens de production. Dorénavant, les machines de production ne sont plus "à vapeur" mais "électrique". Cette époque correspond à la mise en place du taylorisme[1] et du travail à la

[1] Le taylorisme est une méthode de travail qui préconise l'organisation scientifique du travail (OST) grâce à une analyse détaillée des modes et techniques de production. [15]

chaîne rendant productif les ouvriers non qualifiés. Nous parlons alors de production en masse de produits identiques. [15]

1.2.3 La 3ème révolution industrielle

Une 3ème révolution a eu lieu au milieu du 20ème siècle avec l'avènement de l'électronique, des télécommunications ou encore de l'informatique. Ces différentes disciplines vont permettre la mise en place d'automatisations importantes qui soulageront les ouvriers des tâches les plus difficiles. C'est le début de la robotique, de la flexibilité des outils de production et de la production en grandes séries [15].

Figure 1.1. Les quatre révolutions industrielles

1.2.4 L'industrie 4.0

L'Industrie 4.0 est aussi fréquemment nommée « Usine Connectée ». Il s'agit de la quatrième révolution industrielle, après celle de la mécanisation, celle de la production de masse et celle de l'automatisation, le concept fut mis en évidence pour la première fois lors la foire de Hanovre (salon de technologie industrielle) de 2011. [1]

Grâce à l'arrivée de la numérisation, l'industrie devient un système global interconnecté dans lequel les machines, les systèmes ERP (Entreprise Ressource Planning) et les produits communiquent en permanence. [3]

L'Industrie 4.0 est à base de la numérisation et de la mise en réseau. Suite à la pénétration croissante des technologies de l'information et de communication dans la fabrication industrielle, le monde réel fusionne avec le monde virtuel. Les machines, les Hommes et les procédés sont mis en réseau et toutes les informations essentielles sont traitées en temps réel. Cette évolution va durablement changer l'image de l'industrie et permettre des innovations à tous les niveaux. [3]

1.3 L'objectif de l'industrie 4.0

Les objectifs de l'industrie 4.0 sont :

Objectif commerciale : Il permet de répondre pleinement aux exigences de ses clients et parties intéressées

Objectif réglementaire : il permet de respecter en permanence les exigences réglementaires, et légales et autres, applicables à ses produits, aux aspects environnementaux et aux risques associés à son activité.

Objectif approvisionnement : il permet de développer des relations mutuellement bénéfiques avec les fournisseurs.

Objectif écologique : il permet de développer et mettre en place des procédés de conception, de réalisation et de maîtrises sûres, écologiques et efficients.

Objectif sociale : il permet de développer une culture permettant de prévenir les accidents et les maladies professionnelles à travers la maîtrise des risques liés au bruit, à la manutention et à l'utilisation des produits chimiques. [3]

1.4 Défis, avantages et risques :

1.4.1 Les défis :

Avec l'arrivée de l'industrie 4.0 les défis sont nombreux les principaux sont :

- Disposer des nouvelles compétences requises.
- Assurer la sécurité des données.
- Répondre aux besoins en investissements. [2]

1.4.2 Les avantages :

- Avantages sur le marché grâce à d'avantage de rapidité.
- Augmentation de la capacité d'adaptation et de la flexibilité.
- Avantages en termes de coûts.
- Ouverture à de nouveaux champs ou modèles d'activité.
- Cohésion entre production de masse et fabrication personnalisée. [2]

1.4.3 Les risques

- Risque d'obsolescence rapide des nouveaux systèmes.
- Perte de contrôle : les décisions sont déléguées à des systèmes autonomes.
- Les systèmes et structures complexes sont plus difficiles à maîtriser.
- Il se pourrait qu'il y ait plus d'emplois perdus que d'emplois créés.
- Les employeurs profiteront des réseaux connectés et des possibilités de travail décentralisé. Cette constante accessibilité peut engendrer des difficultés de séparation entre travail et vie personnelle. [2]

1.5 Les aspects technologiques de l'industrie 4.0 :

1.5.1 L'intelligence Artificielle (IA) :

L'intelligence artificielle est l'ensemble des techniques permettant à des machines

Informatisées d'accomplir des tâches et résoudre des problèmes algorithmiques ou de logiques. Le nouvel essor de l'intelligence artificielle est dû aux nouvelles capacités d'apprentissage des machines. Il y a encore, les ingénieurs devaient entièrement programmer à la main des algorithmes. Les machines étaient alors limitées ce qu'on leur avait codé.

Aujourd'hui grâce à des machines de plus en plus performantes et la possibilité de traiter une masse importante de traiter de données (big data) il n'y a plus de limites. [4]

A présent les machines peuvent apprendre, le but de l'apprentissage machine est d'entrainer des algorithmes à prédire d'après une base d'information. Anticiper sera alors tout l'enjeu de l'intelligence artificielle. Dans l'industrie, les programmes d'intelligence artificielle apprennent des données contenues dans les systèmes d'information. Ils en déduisent les schémas récurrents, et les comparent à des bases de données plus vastes, ils pourront prendre des décisions et avertiront en cas d'anomalie. L'intelligence est en mesure d'optimiser une chaine de production sur mesure. Par exemple, elle adapte et synchronise les machines selon le besoin. [2]

1.5.2 Le protocole de communication (opc ua : Open Plateform Communication Unified Architecture) :

Est un protocole de communication standard des fabricants développé par l'OPC Foundation pour les applications d'automatisation industrielle. Il est basé sur le principe **client-serveur** et permet une communication transparente, des capteurs actionneurs aux systèmes ERP ou au cloud.

Le protocole est indépendant de la plateforme et intègre par défaut des mécanismes de sécurité, il est considéré comme le protocole de communication idéal pour la mise en œuvre de l'industrie 4.0.

Avec OPC UA toutes les données du processus de production sont transférées via un seul et unique protocole, aussi bien à l'intérieur d'une machine qu'entre machines, ou encore entre une machine et une base de données dans le cloud. [1]

1.5.3 Les systèmes cyber physique de production (CPPS : Cyber Physic Production System) :

Industrie 4.0 décrit une nouvelle structure émergente, dans laquelle les systèmes de fabrication et de logistique sous la forme de systèmes de production cyber-physiques (CPPS) utilisent intensivement les réseaux d'information et de communication pour un échange d'informations largement automatisé et dans lequel les processus de production et d'affaires sont mis en correspondance. La partie cyber fait référence à leur puissance numérique, et la partie physique à leur emprise sur le monde réel. [4]

Dans l'usine de l'avenir, les systèmes cyber-physiques (CPS) sont indispensables. Ils fournissent de manière autonome des informations les concernant et échangent des informations avec d'autres appareils intelligents du réseau. Ils sont identifiées sans équivoque et toutes leurs activités sont enregistrées et tracées. Les CPS sont capables de convertir des valeurs physiques en données numériques. Cela permet à différents logiciels d'interagir – même s'ils sont distants

Le système complet est autonome avec une commande numérique qui se nourrit en permanence des informations des capteurs pour commander les actionneurs. C'est un grand volet de l'industrie 4.0. Cette technologie va permettre d'automatiser et de rendre agile la production. [4]

1.5.4 L'informatique en nuage/l'infonuagique : (Le Cloud computing)

Le cloud computing ou informatique en nuage est une infrastructure dans laquelle la puissance de calcul et le stockage sont gérés par des serveurs distants auxquels les usagers se connectent via une liaison Internet sécurisée. L'ordinateur de bureau ou portable, le téléphone mobile, la tablette tactile et autres objets connectés deviennent des points d'accès pour exécuter des applications ou consulter des données qui sont hébergées sur les serveurs. Le Cloud se caractérise également par sa souplesse qui permet aux fournisseurs d'adapter automatiquement la capacité de stockage et la puissance de calcul aux besoins des utilisateurs. [2]

1.5.5 L'internet industrielle des objets (IIOT : Internet of things) :

L'Internet industriel des objets (IIoT) se focalise sur les seuls besoins des professionnels et des entreprises. L'industrie 4.0 et l'Internet des objets (IoT) sont des concepts qui nécessitent un haut niveau de mise en réseau et de communication entre les appareils et les services. De grandes quantités de données doivent être échangées, entre les capteurs et les dispositifs informatiques. Les protocoles et les normes de contrôle sur PC correspondants sont parfaitement adaptés à cette tâche. L'automate SOA (Service-Oriented Architecture) est un autre facteur fondamental contribuant à la faisabilité de l'IoT et de l'Industrie 4.0. L'accès au PLC (l'automate programmable) via le service Web n'est pas nouveau.

Les fabricants d'objets IIOT se concentrent sur une valeur ajoutée qui fait sens dans le milieu industriel, comme la collecte et l'analyse des données en temps réel, les gains d'efficacité et l'amélioration de la traçabilité. [2]

1.5.6 Grosses données (big data) :

Les Big data ou méga données désignent l'ensemble des données numériques produites par l'utilisation des nouvelles technologies à des fins personnelles ou professionnelles. Cela recoupe les données d'entreprise (courriels, documents, bases de données, historiques de processeurs métiers...) aussi bien que des données issues de capteurs, des contenus publiés sur le web (images, vidéos, sons, textes), des transactions de commerce électronique, des échanges sur les réseaux sociaux, des données transmises par les objets connectés (étiquettes électroniques, compteurs intelligents, Smartphones...), ou encore des données géo localisées, etc. [2]

1.6 L'ordonnancement :

L'ordonnancement est défini comme étant une allocation, dans le temps, des ressources (machines) disponibles aux différents jobs (taches) à réaliser, dans le but d'optimiser un ou plusieurs objectifs. [5]

L'ordonnancement en gestion de production correspond aux concepts et techniques qui règlent les problèmes de priorités successives. [16]

L'Ordonnancement consiste à organiser dans le temps la réalisation d'une suite de tâches, en prenant en compte les contraintes de production :

1. Temporelles : délais requis, retards et priorités.
2. Techniques : contraintes d'enchaînement et technologie des machines.
3. Capacitaires : disponibilité des ressources.

De manière générale, on entend par ordonnancement de la Production, la détermination de l'ordre de passage d'un certain nombre de travaux à l'exécution. Cette détermination concerne la planification de l'utilisation des ressources disponibles en Homme et en machine afin de mieux contrôler les coûts et de maîtriser les délais de fabrication des productions décidées. [6]

1.7 Les ateliers de production :

On distingue trois types d'ateliers à savoir :

1.7.1 Le model open-shop *:*

Dans le type open-shop ou (les ateliers à cheminement libre) l'ordre de passage de n job (taches) sur les m machine n'est pas connu à l'avance. Chaque job j peut avoir son propre ordre de passage sur toutes les machines. Les opérations d'un même job peuvent donc être exécutées dans un ordre quelconque. Le fait qu'il n'y ait pas d'ordre prédéterminé rend la résolution de problème d'ordonnancement de ce type très complexe. [21]

1.7.2 Le model job-shop :

C'est le modèle des ateliers à cheminement multiples, chaque job a un ordre à suivre et chacun d'entre eux peut s'exécuter plusieurs fois sur la même machine, ce qui n'est pas le cas du flow-shop. Il s'agit dans ce cas de déterminer les dates de passage sur différentes ressources d'ordre de fabrication ayant des trajets différents dans l'atelier. [21]

1.7.3 Le model flow-shop :

Correspond à des ateliers à cheminement unique, les ordres de fabrications passent par les machines dans le même ordre, avec des durées opératoires qui peuvent être différentes. Il s'agit d'un ensemble de machines disposées en série, chaque job va s'exécuter sur les m machines et tous les jobs vont suivre le même ordre de passage sur ces machines. [21]

1.8 Conclusion :

Nous avons présenté dans ce chapitre les concepts de l'industrie 4.0, qui correspond en quelques sortes à la numérisation de l'usine. A travers le recours à l'internet des objets et aux systèmes cyber-physique.

Dans le chapitre suivant nous présentons le système sur lequel se base notre travail est une plateforme Industrie 4.0, située au niveau du laboratoire SRP « Systèmes Robotisés de Production » du CDTA.

Chapitre 2 Instrumentation et fonctionnement général de la plateforme

2.1 Introduction :

Dans ce chapitre nous présentons tout d'abord la description de la plateforme d'assemblage robotisée et piloté par les multi agents en expliquant son fonctionnement d'une manière générale, avec les instruments installés au laboratoire du CDTA, ensuite on expose la problématique de notre travail.

2.2 Description du système d'assemblage robotisé :

Le système sur lequel se base notre travail est une plateforme Industrie 4.0, située au niveau du laboratoire SRP « Systèmes Robotisés de Production » de la Division Robotique et Productique au Centre de Développement des Technologies Avancées (CDTA). La plateforme est développée par le Labo SRP/ CDTA en collaboration avec SIEMENS.

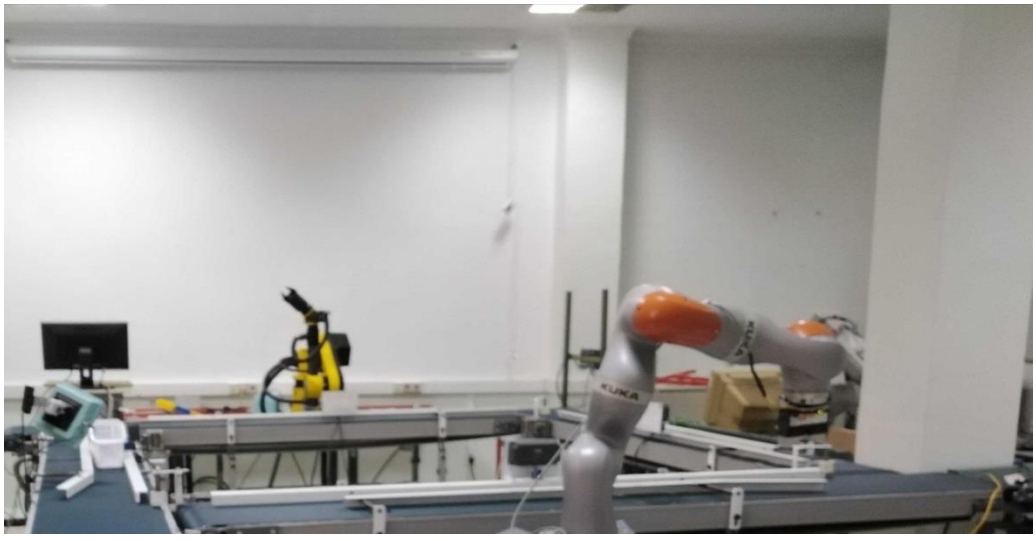


Figure 2.1. La plateforme d'assemblage.

La plateforme se compose de quatre postes de travail et un système de convoyage des pièces entre les postes, constitué de quatre tapis roulants commandés par des moteurs triphasés asynchrones, comme la montre la photo ci-dessous :

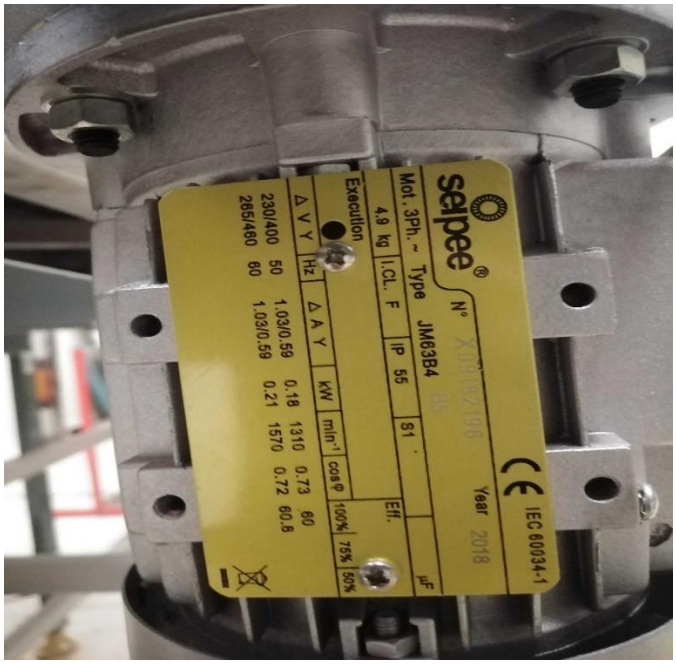


Figure 2.2. Moteur asynchrone.

- Les moteurs des convoyeurs sont commandés par des variateurs de vitesse SIEMENS SIMATIC V20 comme illustré ci-dessous :

Figure 2.3. Variateur de vitesse.

- Chaque poste est doté d'une antenne RFID pour la lecture et écriture des données sur le TAG produit posé sur la palette :

Figure 2.4. Antenne RFID.

- Détecteur photoélectrique SIEMENS SIMATIC PXO 3RG7120 monté juste en dessous de l'antenne RFID (en bleu).

Figure 2.5. Détecteur photoélectrique.

- Les robots utilisés pour la plateforme d'assemblage robotisée :

Le premier poste possède un robot AGVM6, comme la montre la figure ci-dessous :

Figure 2.6. Robot AGVM6.

Ensuite le deuxième poste possède le robot **GT_6A**, comme la figure ci-dessous illustre :

Figure 2.7. Robot GT_6A.

Pour le 3eme poste le robot **MOTOMAN** est installé :

Figure 2.8. Robot MOTOMAN.

Enfin le quatrième poste possède un robot **KUKA IWAA**, comme la figure ci-dessous la montre :

Figure 2.9. Robot KUKA IWAA.

2.3 La commande et supervision :

Le système de commande et supervision de cette plateforme se compose de deux parties :

2.3.1 La partie centralisée :

Elle se compose d'un seul automate programmable le S7-1500 et une IHM de supervision développée en WIN CC.

Figure 2.10. L'interface homme machine.

2.3.2 La partie distribuée :

Dans ce travail nous allons nous concentrer sur la partie distribuée de la plateforme expérimentale. Le système de commande distribué se compose de quatre automates programmables ET200SP de la marque SIEMENS, un automate pour chaque poste. Le programme va s'introduire dans les quatre automates qui sont relié entre eux par le réseau Profinet, chaque automate est relié à un robot qui doit opérer une tache spécifique. Chaque agent communique avec son PLC par le protocole OPC UA.

Figure 2.11. PLC ET200SP.

2.4 Le fonctionnement général de la plateforme expérimentale :

Un modèle de type Flow-Shop synchrone avec contraintes de blocage est étudié ici, dans ce modèle de plateforme, les palettes de produits sont chargées sur le convoyeur au niveau du poste 1 responsable du chargement et déchargement.

Il y a deux variateurs de vitesse, chacun commande deux moteurs asynchrones, donc deux convoyeurs sont commandés par un variateur et les deux autres sont commandés par le second variateur.

Le convoyeur est chargé de transposer la palette de poste en poste.

Au niveau de chaque poste, la palette est détectée grâce à un capteur photoélectrique, ainsi le TAG RFID est lu par le lecteur RFID afin de récupérer les informations nécessaires à l'exécution de l'opération. La préparation du poste robotisé peut parfois requérir un certain temps, l'opération est effectuée une fois ce temps est écoulé.

A la fin de chaque opération les TAG de la palette sont mis à jour, et un signal est donné au convoyeur pour transporter la palette au poste suivant, et ainsi de suite.

Ce model se base sur les contraintes suivantes :

- La machine est prête quand le temps de préparation (Setup Time) est écoulé.

- Quand la machine N+1 est disponible le produit sort de la machine N et ne commence l'opération que lorsque la machine N+1 est prête.
- Chaque produit n'occupe qu'une seule position dans la séquence.
- Chaque position n'est affectée qu'à un seul produit.
- Le produit en position k doit terminer son traitement sur la machine M avant de pouvoir débuter sur la machine suivante M+1.
- Le produit en position K+1 commence son traitement sur la machine M après la fin de traitement du produit en position K sur cette même machine.
- Le produit n'entre pas directement dans les machines il attend un temps Ts que la machine soit prête (Setup Time), le setup time d'une machine est initialisé à chaque fois que le produit finit son opération en cours de la machine.

2.5 La problématique :

L'objectif de l'intelligence artificielle est de construire des programmes informatiques capables d'exécuter des tâches complexes. Mais ce n'est pas toujours le cas pour certains types de problèmes, essentiellement pour les problèmes de groupe et les problèmes de travail coopératif qui nécessite l'intervention d'un ensemble d'entités intelligentes qui doivent communiquer et coopérer, pour atteindre un objectif commun. [12]

On va s'intéresser dans ce travail à l'un des axes de l'intelligence artificielle qui est les systèmes Multi-Agents (SMA) qui est actuellement un champ de recherche très actif, il est à la connexion de plusieurs domaines.

Notre problématique dans ce travail consiste à la mise en place du système de pilotage et de supervision Multi-Agents distribué, et comment un ensemble d'agents peut agir de manière simultanée.

La performance d'un système de pilotage se retrouve fortement tributaire de son aptitude à adapter rapidement les décisions à apprendre. Cette exigence a contribué à l'émergence du concept de pilotage réactif dont la finalité est d'assurer une exécution coordonnée et adaptative des opérations, en répondant aux changements tout en satisfaisant les demandes des clients de manière opportune et rentable. [14]

Le développement de notre application sera assuré moyennant la plateforme Multi-Agents Jade.

Le pilotage dans un système de production, a pour objectif principal « la bonne exécution du programme prévisionnel de production par le système physique ». [15]

Cette fonction assure :

- l'exécution des ordres de fabrication (ordonnancement et lancement).
- la coordination des opérations et des tâches (pour la réalisation d'un processus).
- la synchronisation des opérations (contraintes de précédences).
- la sélection en temps réel du mode de production (opter pour une gamme alternative en cas de problème sur une ressource donnée).
- le suivi des séquences opératoires (suivi des gammes).
- le contrôle de la production (contrôle de la qualité, contrôle des cycles).
- la correction des écarts par rapport au plan de production (modifications, quantités, délais).

Le pilotage est décomposé en deux phases réparties sur cinq niveaux : la gestion prévisionnelle qui concerne les décisions prises hors-ligne et le pilotage en temps réel du système de production qui englobe les décisions prises en ligne. [17]

Trois fonctions de gestion prévisionnelle sont distinguées :

- "La planification" correspond à un horizon temporel à long terme et aboutit à la définition du plan directeur de production à partir des objectifs commerciaux (commandes fermes et prévisionnelles), financiers (coûts fixes et coûts de fabrication) et de production (données techniques et moyens de production).
- A partir du plan directeur de production, la fonction"programmation" établit le programme prévisionnel de production et définit les besoins nets en fonction des quantités et délais de production (besoins bruts), des approvisionnements et des quantités déjà en stock.
- " L'ordonnancement" élabore le planning de production détaillé sur la base de la charge à produire et de l'état des moyens de production à disposition. Ce planning est établi pour chaque atelier et fournit la liste des ordres de fabrication précisant les produits à

réaliser, la quantité à produire, la date de lancement au plus tôt de la production et la date de fin au plus tard. [17]

Deux fonctions de pilotage en temps réel sont définies :

- La fonction conduite est chargée de réaliser la production stipulée dans le planning détaillé de production. A ce niveau, tous les problèmes liés à l'opérationnalisation des décisions prévisionnelles doivent être réglés, il s'agit notamment d'intégrer l'ensemble des contraintes de fabrication (contrôle qualité, arrêts liés la maintenance, etc.) et de réagir aux aléas de production par un suivi de l'exécution du planning de production. [16]

- Le dernier niveau décisionnel, appelé commande, est en relation directe avec le système physique. Son rôle consiste à traduire les ordres de production en séquences automatiques. Dans certain cas, un opérateur peut être sollicité pour assurer le suivi d'exécution de certains ordres de fabrication lorsque l'apparition d'une perturbation nécessite une action locale non intégrée par la commande. [16]

2.6 Conclusion :

Dans ce chapitre nous avons décrit le système d'assemblage robotisé qui est une plateforme Industrie 4.0, ensuite nous avons introduit au système de commande et supervision de la plateforme, nous avons proposé finalement la problématique de notre travail.

Dans le chapitre suivant nous décrivons les approches utilisées dans l'élaboration de la solution, ainsi que les détails techniques.

Chapitre 3 Description de la solution et détails techniques

3.1 Introduction :

Dans le but de décrire les outils et les logiciels utilisés dans notre travail ce chapitre est organisé en deux parties. Dans la première partie nous présentons la description et la modélisation de notre solution, et dans la seconde partie nous présenterons les logiciels utilisés pour notre simulation.

3.2 Modélisation de la chaine de production :

Afin de modéliser la cellule de production et de pouvoir tester ensuite le système Multi-Agent qui sera responsable de piloter la production. La première étape de notre travail consiste à établir une liste de variables, ainsi qu'un logigramme qui décrit le fonctionnement de notre modèle, cela afin de nous faciliter et simplifier l'introduction à la partie de programmation.

3.2.1 Les classes et leurs variables :

Dans un premier temps nous avons créé 3 classes : classe Machine, classe Opération et classe Produit. Chaque classe a ses propres variables, Les tables suivantes montrent les variables utilisées relatives à chaque classe.

Nous avons choisis le terme machine a la place de robot car sa représente une ressource constituée d'un robot et d'un système d'alimentation en matière première (palettes, pièces..)

Classe	Nom de Variable	Type de Variable	Description
machine	Ts	int	Temps de préparation (Setup Time)
machine	cmpset	int	compteur
machine	prete	boolean	machine prete
machine	libre	boolean	la machine libre

Tableau 3.1.les variables de la classe Machine.

class	Nom de variable	type de vriable	Description
produit	Id	int	identification de produit
produit	xxxx	int	Ordre de Passage
produit	op1	operation	operation numéro 1
produit	op2	operation	operation numéro 2
produit	op3	operation	operation numéro 3
produit	cmp	int	compteur d'operation
produit	fini	boolean	operation fini

Tableau 3.2. Les variables de la classe Produit.

class	Nom de variable	type de vriable	Description
operation	Top	int	la durée de l'operation
operation	Td	int	date de debut d'operation
operation	Machine	int	Machine

Tableau 3.3. Les variables de la classe opération.

class	Nom de variable	type de vriable	Description
prog princpale	toutes_les_operations_sont_finis	boolean	
prog princpale	Tx	int	le temp d'attent
prog princpale	Tt	int	le temp de transport de produit
prog princpale	Tp	int	temp total de production
prog princpale	Ts	int	Temps de préparation (Setup Time)

Tableau 3.4. Les variables du programme principal.

3.2.2 Logigramme :

La seconde partie de la modélisation, consiste à établir un logigramme. L'organigramme ou logigramme ou diagramme est un outil d'analyse qui permet de représenter de façon ordonnée et séquentielle, l'ensemble des taches ou évènements mis

en œuvre pour réaliser une activité donnée. Il est constitué d'un ensemble de symboles relié par des flèches. [21]

Chaque symbole représente un évènement ou une tache et la flèche matérialise la relation d'antériorité ou de succession entre deux taches consécutives. [21]

Notre logigramme est réalisé en moyennant Microsoft Visio qui est une application de création de diagrammes et de graphiques vectoriels et fait partie de la famille Microsoft Office.

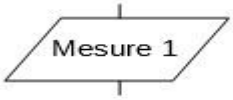
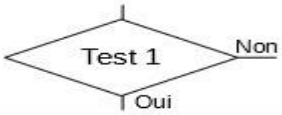
Symbole	Désignation
	**Début** ou **fin** d'un programme, d'un automatisme.
	**Etape**, **opération**, **instruction**, etc. ou opération pour laquelle il n'existe pas de symbole.
	**Entrée** ou **Sortie**, mesure, information capteur, ...
	**Test**, **condition** Décision d'un choix en fonction de la condition
	**Sous-programme** Appel d'un sous-programme
	**Commentaire**
	**Liaison** Les différents symboles sont reliés par des liaisons. Le cheminement va de haut en bas et de gauche à droite.

Figure 3.1. Symbole de logigramme.

Dans le cadre du travail cet organigramme devra simuler trois produits en production et retourne les variables suivantes :

- Le temps de production total du produit.
- Le temps d'occupation de chaque machine appart.
- Le temps total du produit.

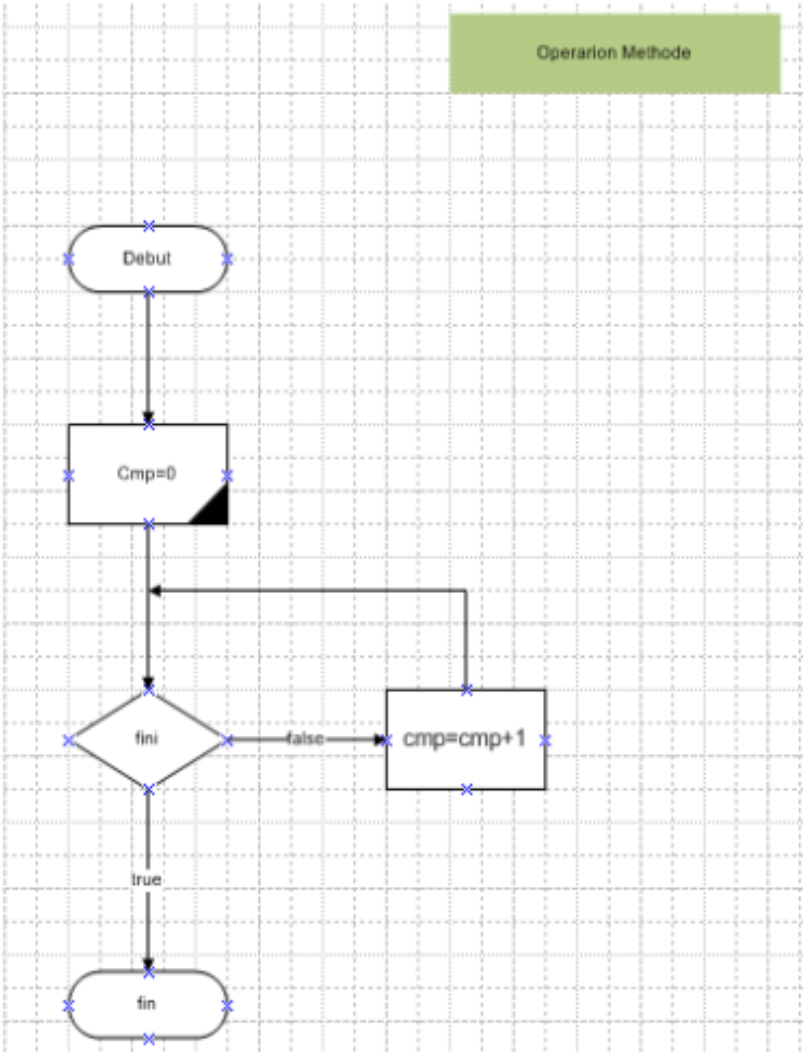


Figure 3.2. Logigramme de classe produit.

Au début le compteur d'opération cmp est à état initiale, il commence à s'incrémenter si le produit ne termine pas son opération.

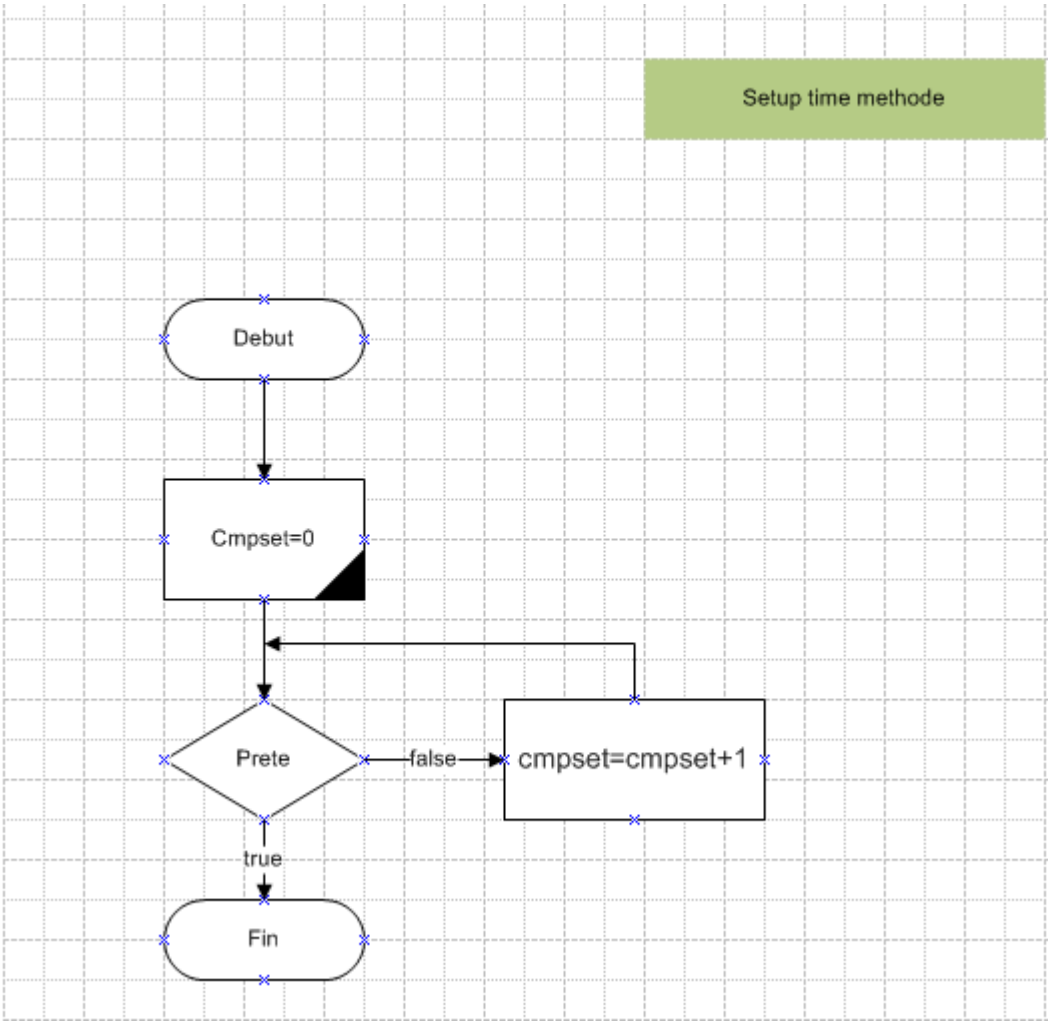


Figure 3.3.logigramme de la classe Machine.

Le produit n'entre pas directement dans les machines, il attend un temps Ts que la machine soit prête (Setup Time), le setup time d'une machine est initialisé à chaque fois que le produit finit son opération en cours de la machine.

Sera imprimé sur une feuille A3 pour qu il soit visible

En fin, nous avons obtient comme un logigramme principal :

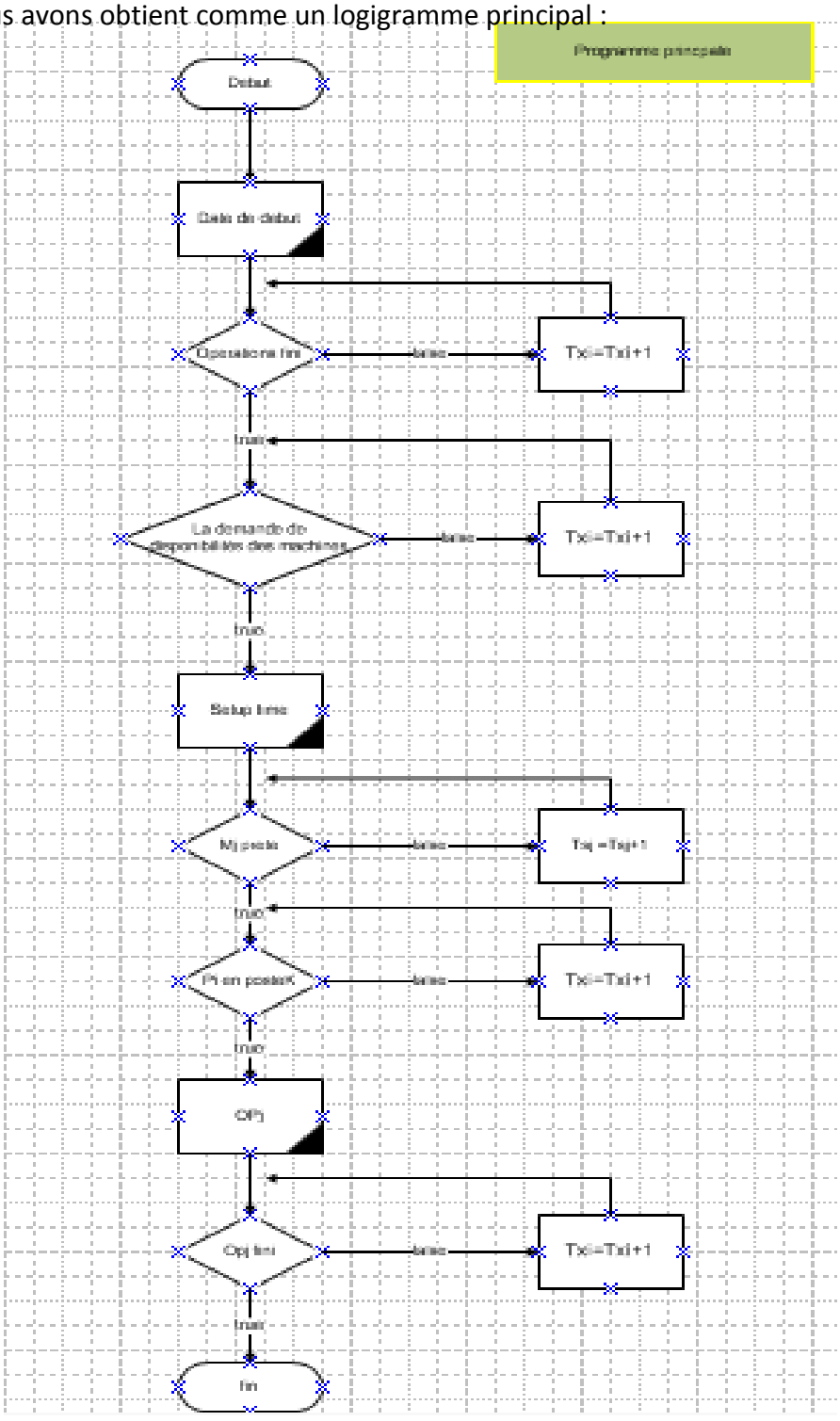


Figure 3.4. Logigramme de Programme principale.

3.3 Les logiciels utilisés :

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.

3.3.1 Le langage java :

Le langage java est un langage généraliste de programmation synthétisant les principaux langages existants lors de sa création en 1995 par SUN Microsystèmes, aujourd'hui racheté par oracle. Il permet une programmation orientée objet (à l'instar de Small Talk et dans une moindre mesure, C++) simple qui réduit le risque d'erreur et d'incohérence, modulaire (langage ADA) et reprend une syntaxe très proche de celle du langage C. [24]

Java est doté d'une riche bibliothèque de classe comprenant la gestion des interfaces graphiques (fenêtre, boite de dialogue).

Java permet de réaliser une très grande quantité d'applications différentes :

- ✓ Des applications, sous forme de fenêtre ou de console.
- ✓ Des applets, qui sont des programmes java incorporés à des pages web.
- ✓ Des applications pour appareil mobiles, avecJ2ME et bien d'autres J2EE, JMF, J3D pour la 3D.

a La programmation orientée-objet :

La programmation orientée-objet (introduite par le langage SmallTalk) propose une méthodologie centrée sur les données. Le programmeur Java va d'abord identifier un ensemble d'objets, tel que chaque objet représente un élément qui doit être utilisé ou manipulé par le programme, sous la forme d'ensembles de données. Ce n'est que dans un deuxième temps, que le programmeur va écrire les traitements, en associant chaque traitement à un objet donné. Un objet peut être vu comme une entité regroupant un ensemble de données et de méthodes (l'équivalent d'une fonction en C) de traitement. [26]

3.3.2 Eclipse :

« Eclipse IDE » est un environnement de développement libre permettant de créer des programmes dans de nombreux langages de programmation (Java, C++, PHP…). C'est l'outil que nous allons utiliser pour programmer. [14]

L'Eclipse est un logiciel ou environnement de travail qui va nous permettre de développer et de compiler nos applications et nos applets. [13]

Nous avons synchronisé Eclipse avec le Github et travaillé en collaboration via le Github.

Github fait penser à un site de réseau social sérieux sur lequel se retrouvent les développeurs de logiciels. Git permet de stocker le code source d'un projet et de suivre l'historique complet de toutes les modifications apportées à ce code, et permet aux développeurs de modifier et d'adapter et d'améliorer le logiciel gratuitement et facilite la programmation collaborative. [14]

Voici quelques étapes fondamentales pour commencer à programmer sous Eclipse :

Pour créer une classe nous allons suivre les étapes montrées sur les figures suivantes :

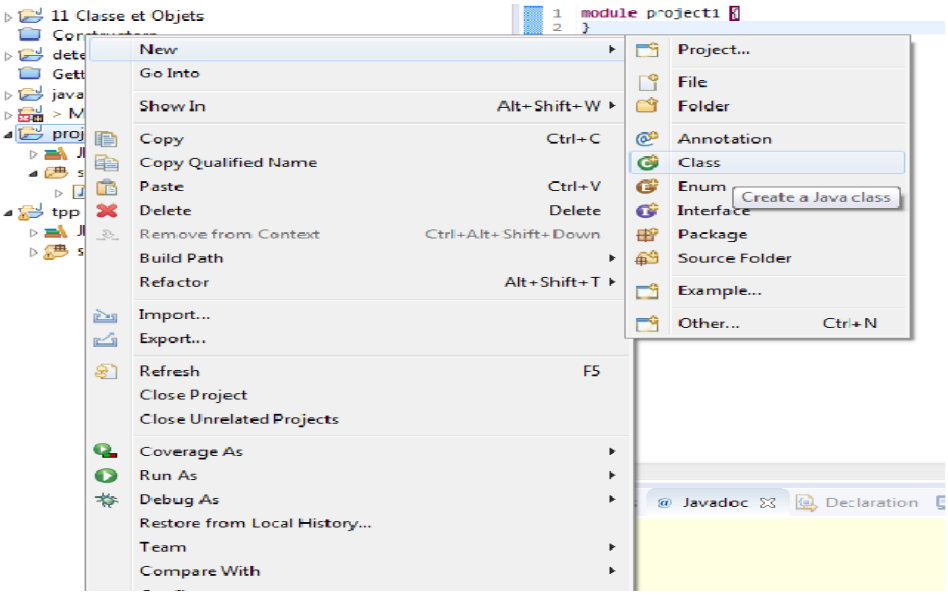


Figure 3.5.Création d'une classe-étape 1.

Ensuite nous allons nommer notre classe :

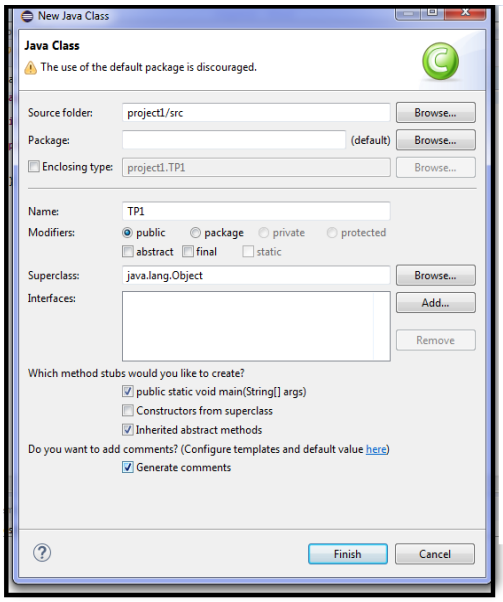


Figure 3.6. Création d'une class-étape 2.

La figure suivante illustre la fenetre principale d'eclipce :

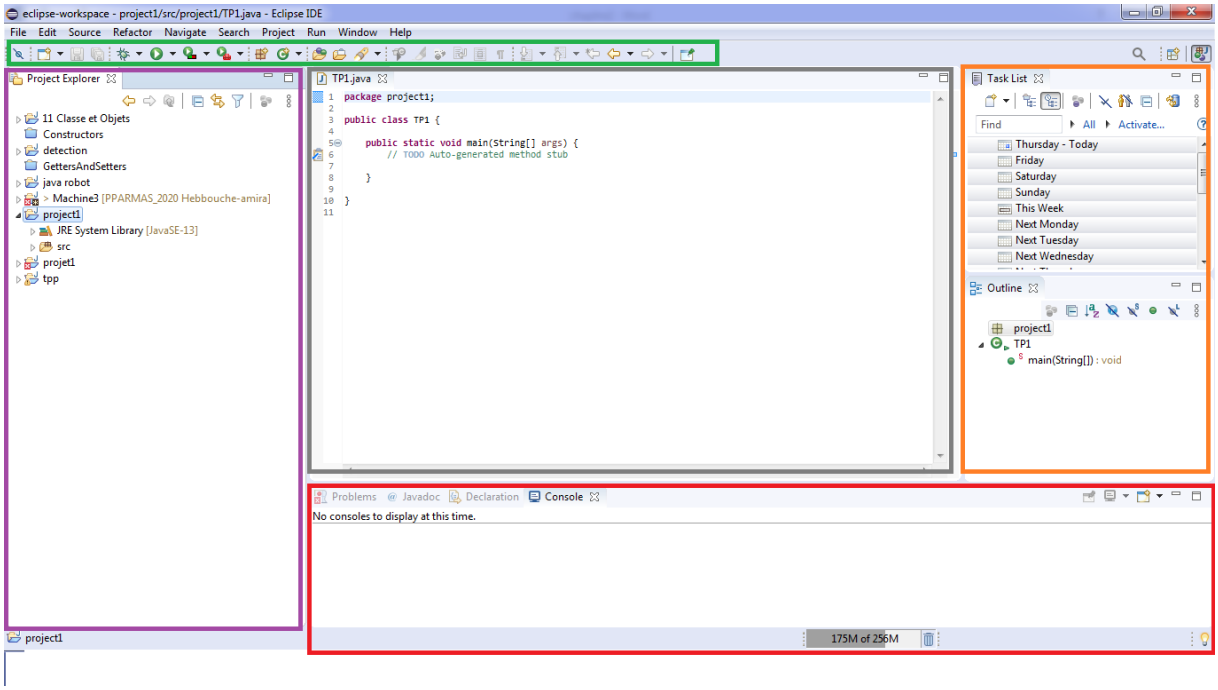


Figure 3.7.Fenêtre principale d'Eclipse.

Nous commençons par l'encadré du haut coloré en vert, il présente la barre d'outils.

Dans l'encadré violet c'est le Project Explorer : ou on trouve le dossier de notre projet ainsi que son contenu, ici nous pourrons gérer notre projet.

Dans l'encadre gris c'est l'éditeur de texte qui nous permet d'écrire nos codes sources.

Dans l'encadré de droite (Orange) c'est le Outline et Task List, nous apporterons les fonctions et les objets avec leurs méthodes et variables.

Et pour finir dans l'encadré du bas (Rouge) c'est là que nous verrons apparaitre le contenu de nos programmes ainsi que les erreurs éventuelles. [14]

3.4 Technologie Multi-Agent et la plateforme JADE :

3.4.1 Agent et Systèmes Multi-agents :

a 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. [25]

Un agent peut être un processus, un robot, un être humain.

Un agent possède les propriétés suivantes :

Réactivité : Percevoir l'environnement et répondre, en temps réel, aux changements.
Proactivité : Capacité de prendre l'initiative / comportement orienté but.
Sociabilité : Capacité d'interagir avec d'autres agents ou utilisateurs.
Autonomie : Il peut prendre des décisions tout seul et il ne dépend pas des autres.
Généralement on ne trouve pas toutes ses caractéristiques en un seul agent. [26]

b Communication entre agents :

Dans un dialogue les agents alternent des rôles actifs et passifs, et échangent des séries de messages en respectant des protocoles bien précis, ce sont les protocoles de coordination, de coopération et de négociation. [24]

- Un agent passif doit accepter les questions des autres agents et répondre à leurs questions.
- Un agent actif doit proposer et envoyer des interrogations. [18]

✓ **Protocole de coordination :**

Les protocoles de coordination lui permettent de gérer ces engagements dans le cas où les circonstances dans lesquelles ils ont été élaborés, évoluent. Ils définissent aussi sous quelles conditions les engagements peuvent être revus et quelles sont alors les actions à prendre. [18]

✓ **Protocole de coopération :**

La coopération entre les agents consiste à décomposer les tâches en sous-tâches puis à les répartir entre les différents agents, il existe plusieurs décompositions possibles, le processus de décomposition doit donc tenir compte des ressources disponibles et des compétences des agents. [18]

✓ **Protocole de négociation :**

La négociation intervient lorsque des agents interagissent pour prendre des décisions communes, alors qu'ils poursuivent des buts différents.

Les langages de négociation : il s'agit d'étudier les primitives de communication pour la négociation, leur sémantique et leur usage dans les protocoles.

Le processus de négociation : il s'agit de proposer des modèles généraux de comportements des agents en situation de négociation. [18]

c La différence entre un agent et un objet :

On peut souligner des différences entre la programmation orientées objet et la programmation orientée agent :

- Les agents sont autonomes alors que les objets ne le sont pas ; un agent va décider par son propre processus de décision s'il exécute ou non une action requise.

• Les agents ont leurs propres buts et ils agissent d'une manière proactive pour atteindre leurs buts (par exemple, ils saisissent des opportunités) alors que les objets ne le font pas.

• Les agents sont capables d'un comportement social : ils peuvent s'engager dans des interactions complexes, par exemple coopération, compétition, négociation, avec d'autres agents, ce n'est pas le cas des objets. [9]

• Un système multi-agents est, normalement, un système dans lequel les agents correspondent à des chemins d'exécution séparés, chaque agent dans un système multi-agents a son propre « thread » d'exécution alors que les objets, à part les objets concurrents, ne présentent pas cette caractéristique. [9]

3.4.2 Système Multi-Agents (SMA) :

Un SMA est un ensemble d'agents évoluant dans un environnement commun et interagissant entre eux selon une certaine organisation. Ou encore, c'est un ensemble d'entités qui coordonnent leurs connaissances, buts, expériences et plans pour agir ou résoudre des problèmes, incluant le problème de la coordination inter-agent lui-même. Les agents composent un système, car ils constituent un ensemble cohérent autour d'un objet commun. L'objet qui les relie peut prendre diverses formes : ce peut être un objectif commun (par exemple, résoudre un problème), un langage commun le minimum étant un environnement commun. L'ensemble des agents forme un groupe à cause de leur interaction. Selon les objectifs qu'ils poursuivent et leurs capacités à les atteindre, l'interaction entre les membres du groupe peut être coopérative, conflictuelle, ou tout autre type de relation intermédiaire entre ces deux extrêmes. [26]

3.4.3 Domaine d'application des systèmes Multi-Agents :

Plusieurs domaines existent pour les SMA, nous citons :

- 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. [26]

3.4.4 Plateforme de développement des systèmes Multi-Agents :

Les plateformes multi-agents permettent aux développeurs de concevoir et de réaliser des applications plus rapidement. En effet, les concepteurs peuvent se passer des fonctions de base pour la création et l'interaction entre agents. Ils éliminent, dans la plupart des cas, la nécessité d'être familier avec les différents concepts théoriques des systèmes multi-agents. Parmi les plateformes fournies comme logiciels libres, il y'a quelques plateformes plus connues pour avoir été utilisées dans le développement de plusieurs applications : JADE, MACE, ZEUS, MADKIT, SWORM. Il faut noter que cette

liste n'est pas exhaustive, et qu'il y a aussi d'autres plateformes qui ont été utilisées avec beaucoup de succès pour bâtir diverses applications. [8]

Nous avons développé notre application sous la plateforme multi-agent JADE pour les raisons suivantes :

- ✓ Facilité d'installation.
- ✓ Documentation détaillée.
- ✓ Utilise un langage puissant et stable (JAVA).
- ✓ Intégration avec d'autres outils de developpement (intégration dans Eclipse via les plugins EJIP et EJADE).
- ✓ Licence libre.

3.4.5 La plateforme JADE (Java Agent Developpent Framework) :

Est une plateforme multi-agents crée par le laboratoire TILAB.

C'est un Framework qui permet le développement de systèmes multi-agents et d'applications conformes aux norms FIPA (foundation for intelligent physical agents : La FIPA est une organisation née en 1996 dont l'objectif est de produire des standards pour l'interopération d'agent logiciel hétérogène). [7]

Une application JADE est une plateforme déployée sur une ou plusieurs machines. La plateforme héberge un ensemble d'agents, identifiés de manière unique, pouvant communiquer de manière bidirectionnelle avec les autres agents. Chaque agent s'exécute dans un conteneur (container) qui lui fournit son environnement d'exécution il peut migrer à l'intérieur de la plateforme et toute plateforme doit avoir un conteneur principal qui enregistre les autres conteneurs. JADE a été utilisée dans plusieurs domaines a savoir les applications mobiles, Internet, Réseaux et les Systèmes Manufacturier, etc. [9]

Jade possède trois modules principaux (nécessaire aux normes FIPA), Ces trois modules sont activés à chaque démarrage de la plate-forme : [7]

- ✓ DF (directory facilitator) fournit un service de pages jaunes à la plateforme.
- ✓ ACC (agent communication Channel) gère la communication entre les agents.

✓ AMS (Agent Management System) supervise l'enregistrement des agents, leur authentification, leur accès et l'utilisation du système.

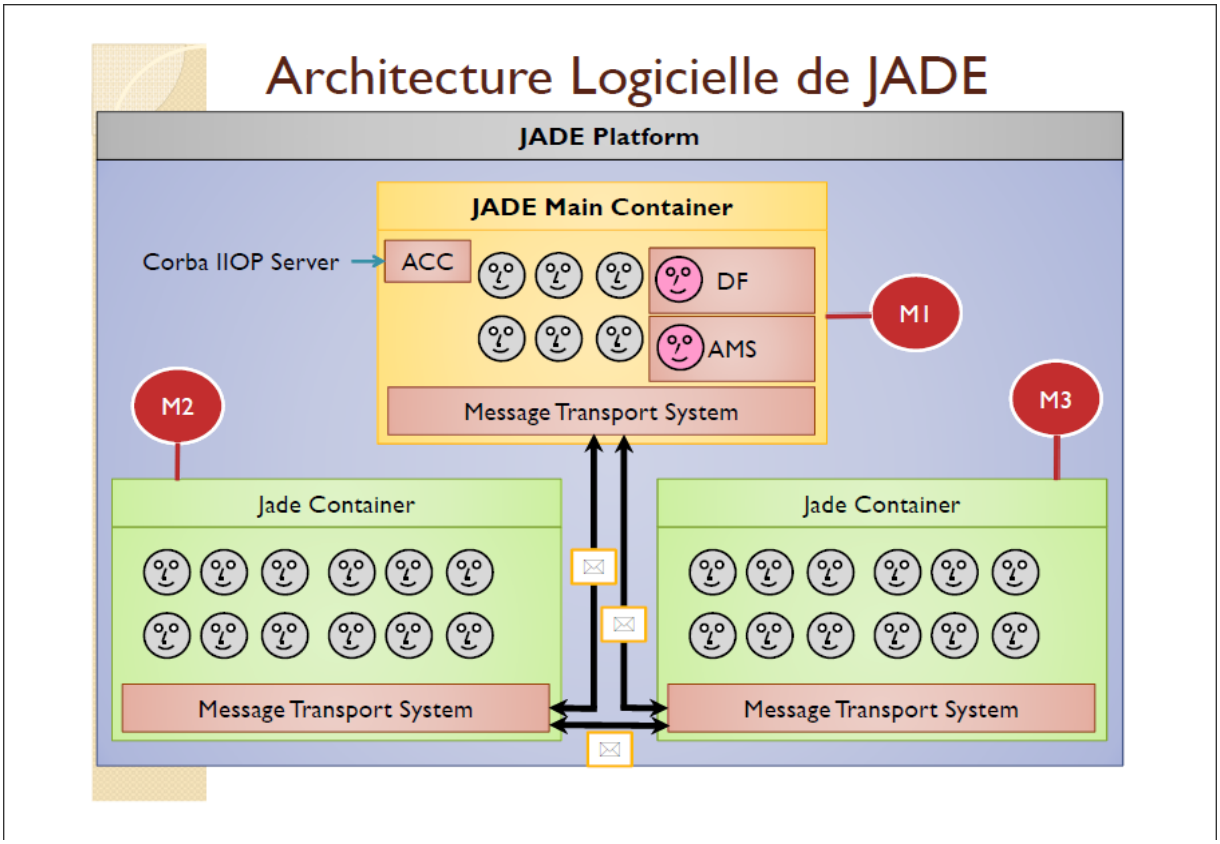


Figure 3.8. Plateforme Jade.

L'interaction entre agents se fait par envoi de messages dont le contenu est exprimé en ACL : ACL (Agent Communication Language) est un langage de haut niveau qui est basé sur la théorie des actes de langage : les messages sont des actions ou des actes communicatifs, car ils sont prévus pour effectuer une certaine action en vertu de l'envoi. Les spécifications de FIPA-ACL décrivent chaque acte communicatif avec une forme narrative et une sémantique formelle basée sur la logique modale. Ainsi, il maintient l'approche de distinguer le langage externe du langage interne. Le langage externe définit la signification prévue du message, le langage interne ou le contenu, dénote l'expression à laquelle s'appliquent les croyances, les désirs, et les intentions des interlocuteurs. [13]

Les agents JADE utilisent des messages conformes aux spécifications de la FIPA (FIPA-ACL). Les messages JADE sont des instances de la classe ACLMessage du package jade.lang.acl. Ces messages sont composés en général de :

- l'émetteur du message : un champ rempli automatiquement lors de l'envoi d'un message ;
- l'ensemble des récepteurs du message : un message peut être envoyé à plusieurs agents simultanément ;
- l'acte de communication : qui représente le but de l'envoi du message en cours (informer l'agent récepteur, appel d'offres, réponse à une requête…) ;
- le contenu du message ;
- un ensemble de champs facultatifs, comme la langue utilisée, l'ontologie, le timeOut, l'adresse de réponse…
- Pour envoyer un message, il suffit de remplir les champs nécessaires (l'ensemble des récepteurs et le contenu du message et l'acte de communication) d'un message JADE, puis d'appeler la méthode send() de la classe Agent. [18]

Un GUI de DF (directory facilitator) peut être lancé du menu outil RMA. Le DF est un composant qui fait office d'annuaire. C'est un service de « pages jaunes » qui permet de mettre en relations les agents avec leurs compétences. Un agent peut enregistrer ses compétences dans le DF ou interroger le DF pour connaître les compétences proposées par les autres agents. Le GUI permet d'associer ces DF afin de créer plusieurs niveaux de domaines. Le GUI peut accéder de la même manière à n'importe quel DF situé sur une plateforme distante d'agents non Jade. [14]

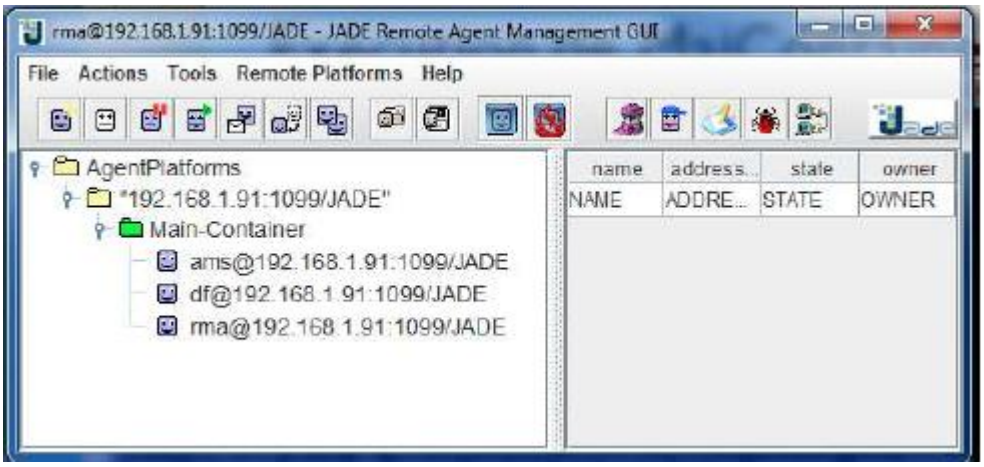


Figure3.9. Interface graphique de la plateforme JADE.

JADE est un intergiciel qui facilite le développement des systèmes multi agents (SMA). JADE contient :

- ✓ Un RuntimeEnvironnement : l'environnement ou les agents peuvent vivre.
- ✓ Une librairie de classes : que les développeurs utilisent pour écrire leurs agents.
- ✓ Une suite d'outils graphiques : qui facilitent la gestion et la supervision de la plateforme des agents.

Chaque instance du JADE est appelée conteneur « Container », et peut contenir plusieurs agents.

Un ensemble de conteneurs constituent une plateforme.

Chaque plateforme doit contenir un conteneur spécial appelé conteneur principal (main-container) et tous les autres conteneurs s'enregistrent auprès de celui-là dès leur lancement. [13]

3.5 Protocole de communication client/serveur :

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 établissement 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.

Une socket est connue comme un type de logiciel qui agit comme un point d'extrémité qui fonctionne en établissant une liaison de communication réseau bidirectionnelle entre l'extrémité du serveur et le programme de réception du client. On l'appelle aussi 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** (TCP Layer) de sorte que la couche puisse identifier la demande de partage de données. Une socket est capable de simplifier le fonctionnement d'un programme car les programmeurs n'ont plus qu'à se soucier de manipuler les fonctions de la socket, ce qui leur permet de compter sur le système d'exploitation pour transporter correctement les messages sur le réseau.

Nous avons créé un programme serveur d'un cote et de programme client de l'autre cote. Le programme client se connecte au serveur, et c'est le serveur qui gère les communications, cette coopération s'effectue sous la forme d'un échange de données où le client reçoit les résultats finaux délivrés par le serveur. [18]

A côté du serveur :

En général, un serveur exécute sur un ordinateur précis et il crée une socket serveur (associé à un port) et se met en attente. Le serveur enregistre son service sous un numéro de port. Puis, le serveur se met en attente sur ce service. La classe ServerSocket est utilisé coté serveur : elle attend simplement les appels du ou des clients.

A côté du client :

Les clients connaissent le nom d'hôte du serveur et le numéro du port que le serveur travaille avec le client peut établir une connexion avec le serveur en

demandant la création d'une Socket à destination du serveur pour le port sur lequel le service a été enregistré. Le client se connecte au socket serveur, deux sockets sont alors crées : un socket client, coté client, et socket service client coté serveur. Ces sockets sont connectés entre elles.

Nous avons intégré des sockets à nos programmes client-serveur Dans notre cas, le serve ur est au niveau du programme de simulation de la plateforme, et les agents du système de pilotage sont des clients, les sockets intégrées au système sont programmé en JAVA.

3.6 Conclusion :

Dans ce chapitre, nous avons brièvement présenté le domaine des systèmes multi-agents, la technologie ces dernières années, est imposée comme une approche importante pour le développement des systèmes de pilotage distribués et intelligents. Nous avons décrit aussi les éléments de base qui nous permet de réaliser notre travail.

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

Chapitre 4 Implémentation, simulation et résultats

4.1 Introduction :

Dans le but de décrire le système réalisé, ce chapitre est organisé en 3 parties : le programme principal, le système multi-agents et le protocole de communication.

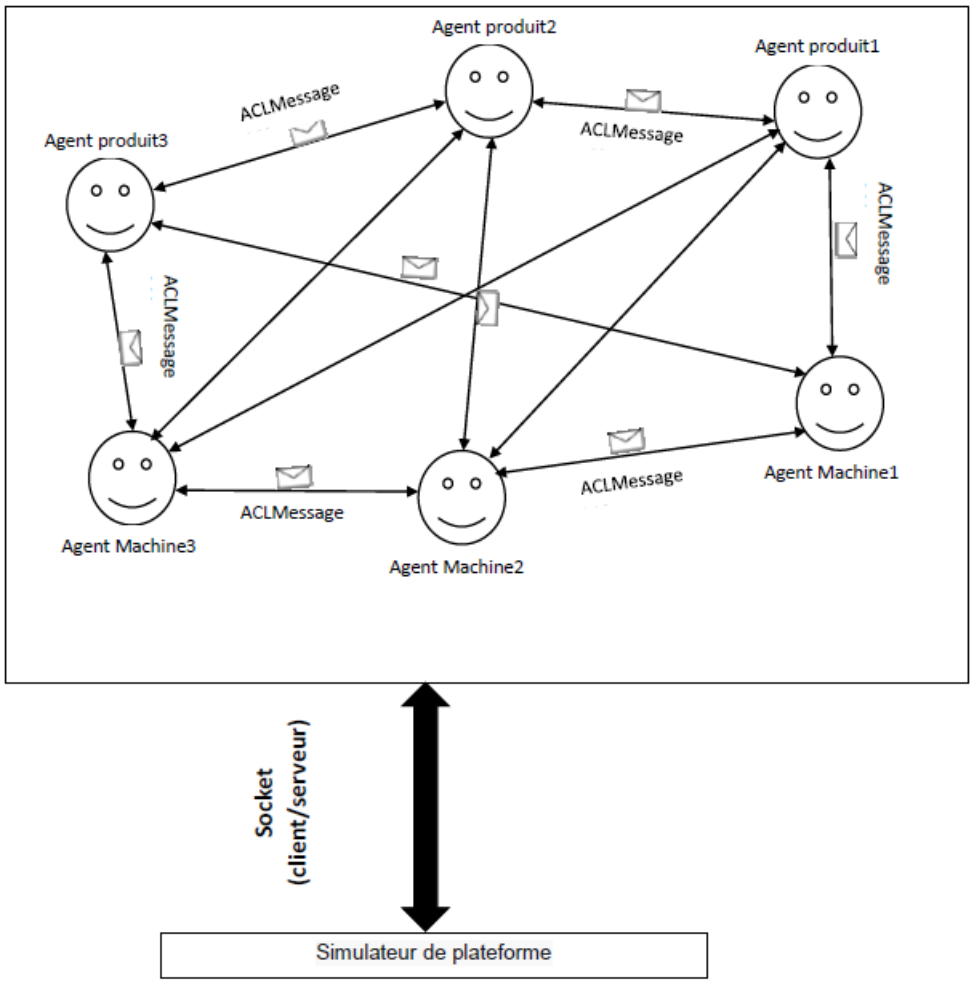


Figure 4.1. L'architecture SMA.

4.2 Rappel sur la plateforme :

La plateforme d'assemblage robotisé, décrite au chapitre 2, figure 2-1 se compose de quatre postes de travail, un système de convoyage des pièces est chargé de transposer la palette de poste en poste, constitué de quatre tapis roulants commandés par des moteurs triphasés asynchrones.

4.3 Le programme de simulation de la plateforme :

Afin de simuler la plateforme, le temps de chargement et déchargement des produits a été supposé nul, autrement dit, le premier poste de la plateforme responsable du chargement et du déchargement des produits ne sont pas considéré dans la simulation. Uniquement les postes opératoires de la plateforme ont été pris en compte, c'est-à-dire 3 postes, appelés conventionnellement « Machines ».

Le programme principal du simulateur écrit en JAVA, en utilisant Eclipse IDE est composé de trois classes :

Classe Machine, Classe Produit, Classe Opération.

Nous avons créé un programme principal dans le but de tester notre système multi-agents.

L'exécution d'une séquence de jobs est simplement l'ordre de passage des produits sur les machines. Pour exécuter une séquence de jobs, on a besoin d'un calendrier contenant des informations concernant l'ordonnancement des jobs. Ainsi, un diagramme de GANTT permet de visualiser un ordonnancement donné, chaque bloc de ce diagramme donne le temps de début et de fin d'exécution de chaque job. [12]

Un diagramme de Gantt est un outil permettant de visualiser dans le temps les diverses taches composant un projet.

Dans un diagramme, on a deux axes perpendiculaires. L'axe horizontal représente les unités de temps en secondes, tandis que l'axe vertical représente les produits. Le code couleur permet de distinguer les machines de notre plateforme.

En premier lieu nous avons proposé un plan d'ordonnancement de 10 produits de 4 gammes différent comme illustré dans le tableau ci-dessous :

Les temps des opérations (en second) sont de notre choix pour toutes les plans d'ordonnancement illustrés dans ce chapitre.

nom de produit	les gammes	temp d'operation1	temp d'operation2	temp d'operation3
produit1	1	54	79	16
produit2	2	83	3	89
produit3	3	15	11	49
produit4	4	71	99	15
produit5	2	83	3	89
produit6	2	83	3	89
produit7	3	15	11	49
produit8	4	71	99	15
produit9	1	54	79	16
produit10	3	15	11	49

Tableau 4.1. Plan d'ordonnancement de 10 produits de 4 gammes.

La figure 4.2 représente le plan d'ordonnancement exécuté sous forme de diagramme de Gantt à 10 produits de 4 gammes et 3 machines. La valeur du temps total d'exécution makespan (second) pour cet exemple s'incrémente avec le temps. Le produit 2 reste bloqué sur la machine M2 tant que la machine M3 n'est pas disponible. Ainsi, le produit 2 ne peut pas commencer son opération sur la machine M3 avant la fin de l'opération de produit 1 sur la machine M3. Les diagrammes de Gantt sont réalisés par Microsoft Excel 2013.

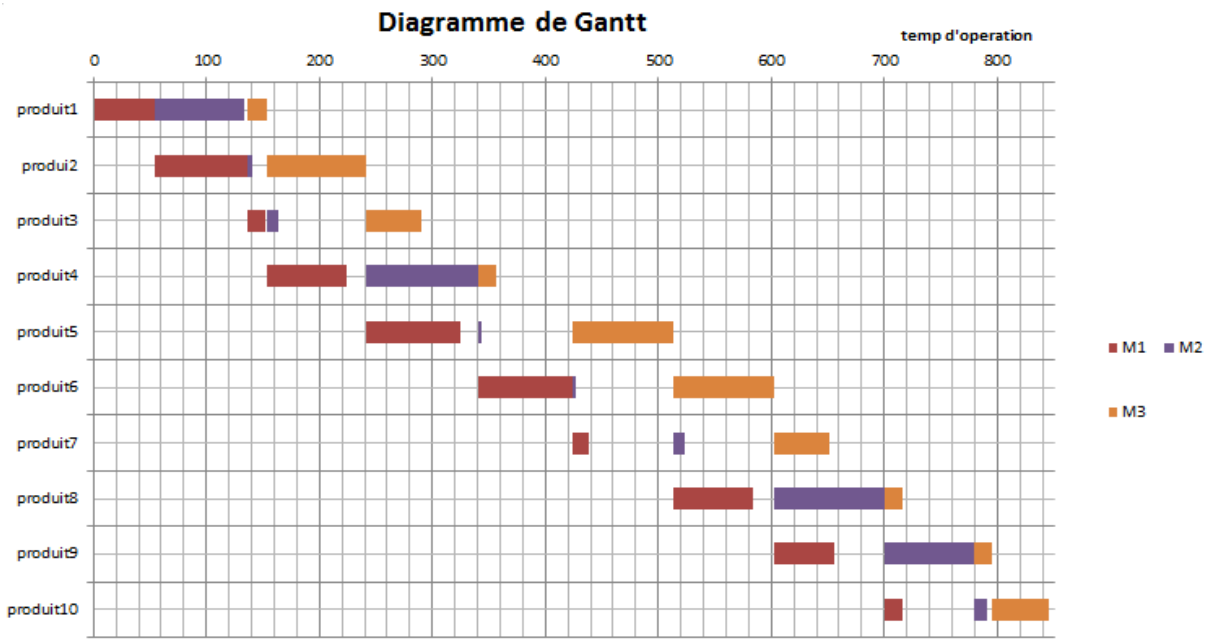


Figure 4.2. Diagramme de Gantt.

Le tableau 4.2 représente un plan d'ordonnancement de 10 produits de 10gammes :

nom de produit	les gammes	temp d'operation1	temp d'operation2	temp d'operation3
produit1	1	54	79	16
produit2	2	83	3	89
produit3	3	15	11	49
produit4	4	71	99	15
produit5	5	77	56	89
produit6	6	36	70	45
produit7	7	53	99	60
produit8	8	38	60	23
produit9	9	27	5	57
produit10	10	87	56	64

Tableau 4.2. Plan d'ordonnancement de 10 produits de 10 gammes.

La figure 4.3 représente le diagramme de Gantt d'un plan d'ordonnancement de 10 produits de 10 gammes :

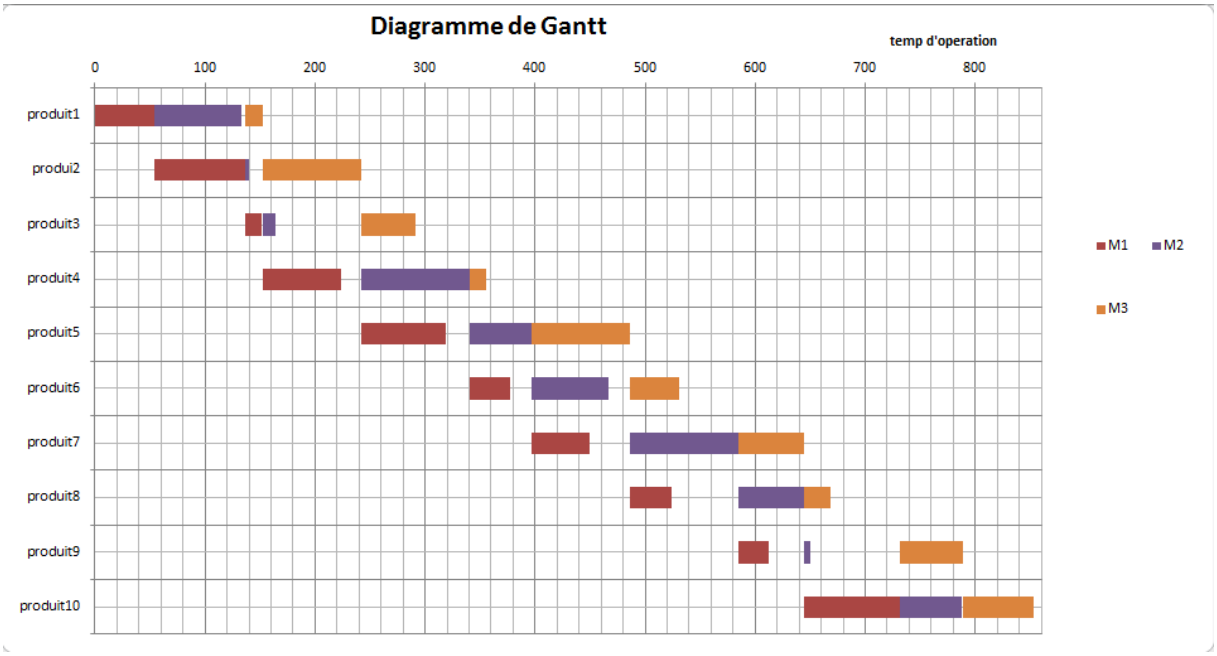


Figure 4.3. Diagramme de Gantt de 10 produits de 10 gammes.

4.4 Protocole de communication :

Le Programme de simulation de la plateforme et le système multi-agent communiquent entre eux via le protocole de communication Socket (client/serveur).

Comme nous avons cité dans le chapitre 3 le simulateur de la plateforme conçu comme un serveur et le system multi-agents comme un client.

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 :

```
Thread th1 = new Thread(new Runnable() {
    @Override
    public void run() {
        while (true) {
          try {
                dis = new DataInputStream(s.getInputStream());
                dout = new DataOutputStream(s.getOutputStream());

                String msg = dis.readUTF();
                System.out.println(msg);

                dout.writeUTF("client is connected");
                dout.flush();

        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        }
    }
});
th1.start();
```

Figure 4.4. L'implémentation de la réception.

La figure 4.5 présente le fragment code d'un socket serveur (le simulateur) :

```
try {
    ServerSocket ss=new ServerSocket(6666);
    Socket s=ss.accept();

    DataInputStream dis=new DataInputStream(s.getInputStream());
    DataOutputStream dout=new DataOutputStream(s.getOutputStream());

    dout.writeUTF("server is connected");
    dout.flush();

    String str = dis.readUTF();
    System.out.println(str);

} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Figure 4.5. Implémentation d'un socket serveur.

4.5 Système Multi-Agents :

Notre SMA comporte deux classes Agents : classe AgentsMachines, classe AgentsProduits.

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 autre 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 behavriour.Behaviours. Chaque objet de ce type dispose d'une méthode action () qui constitue le traitement à effectuer par celui-ci.

Un agent JADE est une classe qui hérite de la classe Agent et qui redéfinie des méthodes qui présentent le cycle de vie de l'agent dans la plateforme.

4.5.1 Identification d'un agent :

L'agent est identifié auprès de l'AMS par un identifiant unique, appelé GUI (global unique identifier), cet identifiant est un objet de type jade.core.AID. Un objet de type AID contient le nom de l'agent plus le nom de la plateforme (nom de domaine ou adresse IP) ainsi que le numéro de port de l'annuaire.

Les méthodes de la classe jade.core.AID utilisés dans notre SMA sont :

getAID() : de la classe Agent permet de récupérer l'identifiant unique de l'agent.

getName() :retourne le nom de l'agent.

getLocalName() :retourne uniquement le nickname (nom local). [26]

4.5.2 Les comportements utilisés :

JADE alloue un thread par agent, pour cela un agent exécute un Behaviour à la fois.

1 agent = 1 thread implémenté en Java selon l'API JADE, qui exécute un ensemble d'actions.

Les actions sont regroupées en comportements(Behaviour), nous avons utilisé principalement trois types de comportement :

- **One-shot Behaviour :**

Un one-shot Behaviour : estuneinstance de la classe jade.core.behaviours.OneShotBehaviour.

Il a la particularité d'exécuter sa tâche une et une seule fois puis il se termine.

La classe OneShotBehaviour implémente la méthode done() et elle retourne toujours true.

- **Cyclic Behaviour :**
  - Un cyclic Behaviour est une instance de la classe cde.core.behaviours.CyclicBehaviour.
  - Un cyclic Behaviour exécute sa tâche d'une manière répétitive.
  - La classe CyclicBehaviour implémente la méthode done() qui retourne toujours false.

- **TickerBehaviour :**
  - Le TickerBehaviour est implémenté pour qu'il exécute sa tâche périodiquement par la méthode onTick().
  - La durée de la période est passée comme argument au constructeur.

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

Voici la procédure d'envoi qui implémente l'OneShotBehaviour dans lequel les agents s'envoient des messages entre eux.

```
addBehaviour(new OneShotBehaviour() {
    String Str = "message";
    @Override
    public void action() {

        ACLMessage Aclmessage = new ACLMessage(ACLMessage.INFORM);
        Aclmessage.addReceiver(new AID("Machine1" , AID.ISLOCALNAME));
        Aclmessage.setContent(Str);
        send(Aclmessage);
    }
});
```

Figure4.6. L'implémentation de l'OneShotBehaviour.

Pour envoyer un message, il suffit de remplir les champs nécessaires (l'ensemble des récepteurs et le contenu du message et l'acte de communication) d'un message JADE, puis d'appeler la méthode send() de la classe Agent.

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

```
addBehaviour(new CyclicBehaviour() {

    @Override
    public void action() {
        MessageTemplate messageTemplate = MessageTemplate.MatchPerformative(ACLMessage.INFORM);
        ACLMessage AclMessage =receive(messageTemplate);
        if (AclMessage != null) {
            gui.showMessage("sender : " + AclMessage.getSender().getLocalName(), true);
            gui.showMessage(" communication act : " + ACLMessage.getPerformative(AclMessage.getPerformative()), true);
            gui.showMessage("message content : " +AclMessage.getContent(),true);
        }
        else  block();
    }
});
```

Figure 4.7. L'implémentation de CyclicBehaviour.

La réception d'un message est aussi simple que l'envoi. Il suffit d'appeler la méthode receive() de la classe Agent.

La classe MessageTemplate dispose d'un ensemble de méthodes statique de fabrique d'objet MessageTemplate selon différents critère.

Comme le fragment de code illustré dans la figure 4.7 tous les messages ignorés sauf ceux qui ont l'option performative est INFORM.

Parmi les outils qui ont été développés dans la plateforme JADE afin de supporter la difficulté de la tâche du débogage des applications multi-agents c'est l'agent sniffer, quand un utilisateur décide d'épier 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 du sniffer, l'utilisateur peut avoir et enregistrer tous les messages.

La figure 4.8 représente le diagramme de séquence qui est extraite à partir de l'outil agent sniffer de la plateforme JADE.

Nous avons utilisé l'agent sniffer de JADE pour suivre l'échange de message entre les agents.

Nous pouvons lire les contenus des messages en double-cliquant sur la flèche et aussi lire le type des messages de chaque agent.

La figure 4.8 représente le diagramme de séquence de 10 produits de 4 gammes :

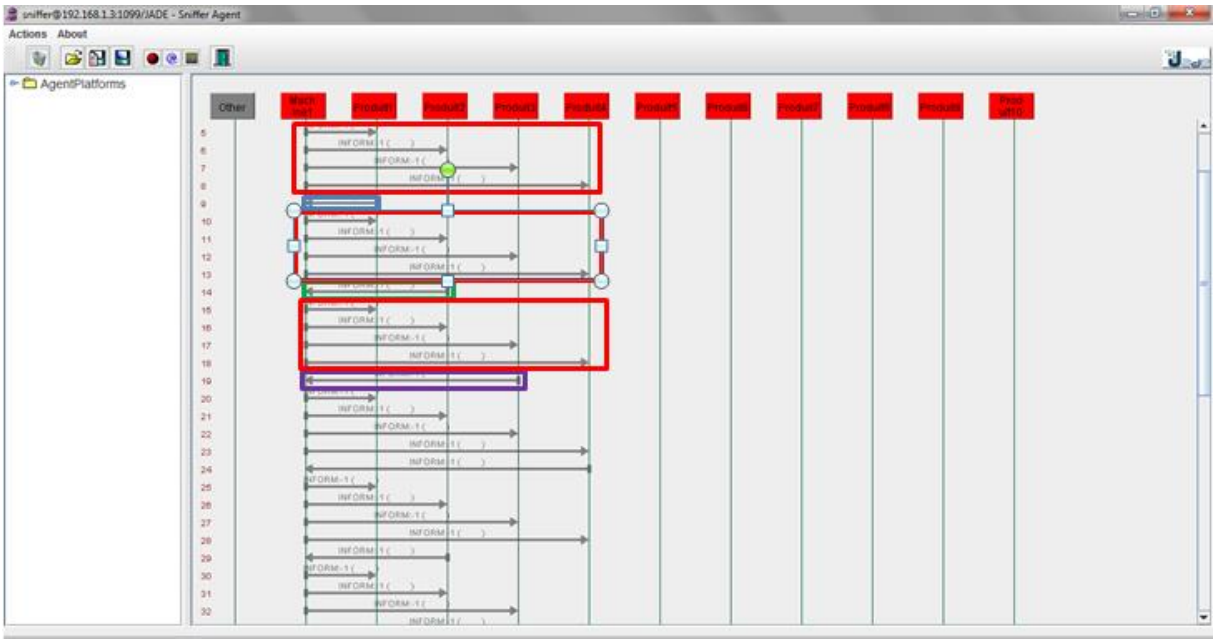


Figure 4.8. Diagramme de séquence(1).

Encadrées en rouge : l'agent machine1 envoie un message **ACL** à tous les agents de type **INFORM** qui contient l'état de la machine.

Encadrées en bleu : l'agent produit1 envoie un message **ACL** à l'agent machine1 de type **INFORM** qui contient la date de début de production de produit1 de la gamme 1 et le temps de l'opération selon l'ordre de passage.

Encadrés en vert : l'agent produit2 envoie un message **ACL** à l'agent machine1 de type **INFORM** qui contient la date de début de production de produit2 de la gamme 2 et le temps de l'opération selon l'ordre de passage.

Encadrés en mauve : l'agent produit2 envoie un message **ACL** à l'agent machine1 de type **INFORM** qui contient la date de début de production de produit3 de la gamme 3 et le temps de l'opération selon l'ordre de passage.

La figure 4.9 représente le diagramme de séquence de 10 produits de 10 gammes :

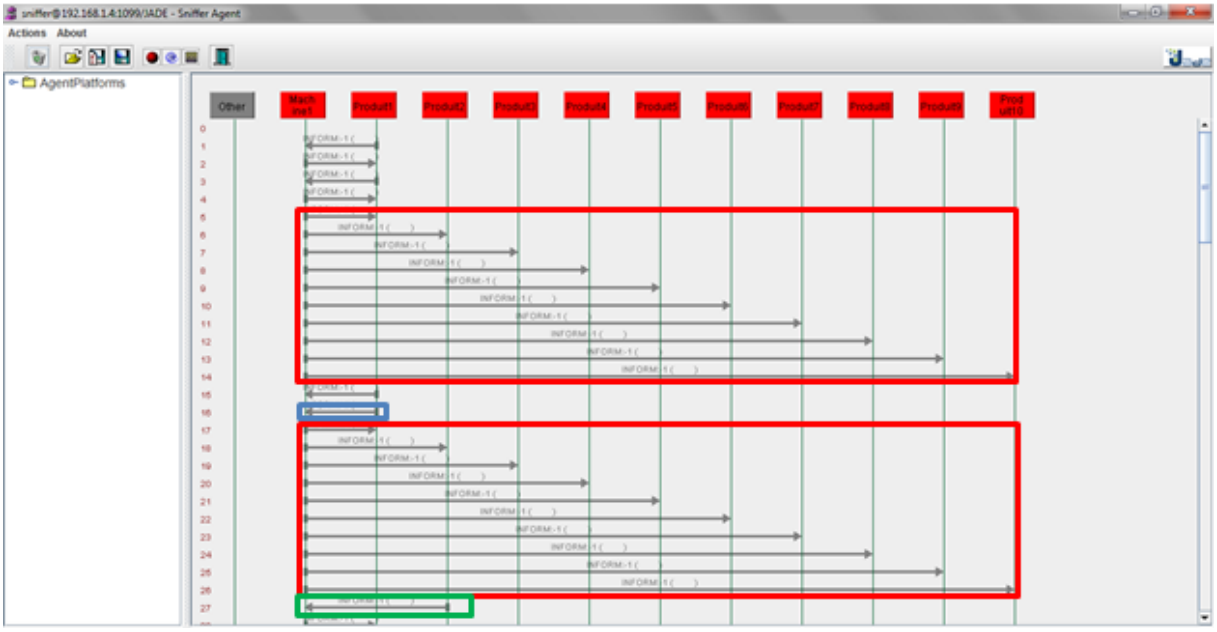


Figure 4.9. Diagramme de séquence(2).

Encadrés en rouge : l'agent machine1 envoie un message **ACL** à tous les agents de type **INFORM** qui contient l'état de la machine.

Encadrés en Blue : l'agent produit1 envoie un message **ACL** à l'agent machine1 de type **INFORM** qui contient la date de début de production de produit1 de la gamme 1 et le temps de l'opération selon l'ordre de passage.

Encadrés en vert : l'agent produit2 envoie un message **ACL** à l'agent machine1 de type **INFORM** qui contient la date de début de production de produit2 de la gamme 2 et le temps de l'opération selon l'ordre de passage.

La figure 4.10 représente le diagramme de séquence de 10 produits de 10 gammes de diffèrent ordre de passage :

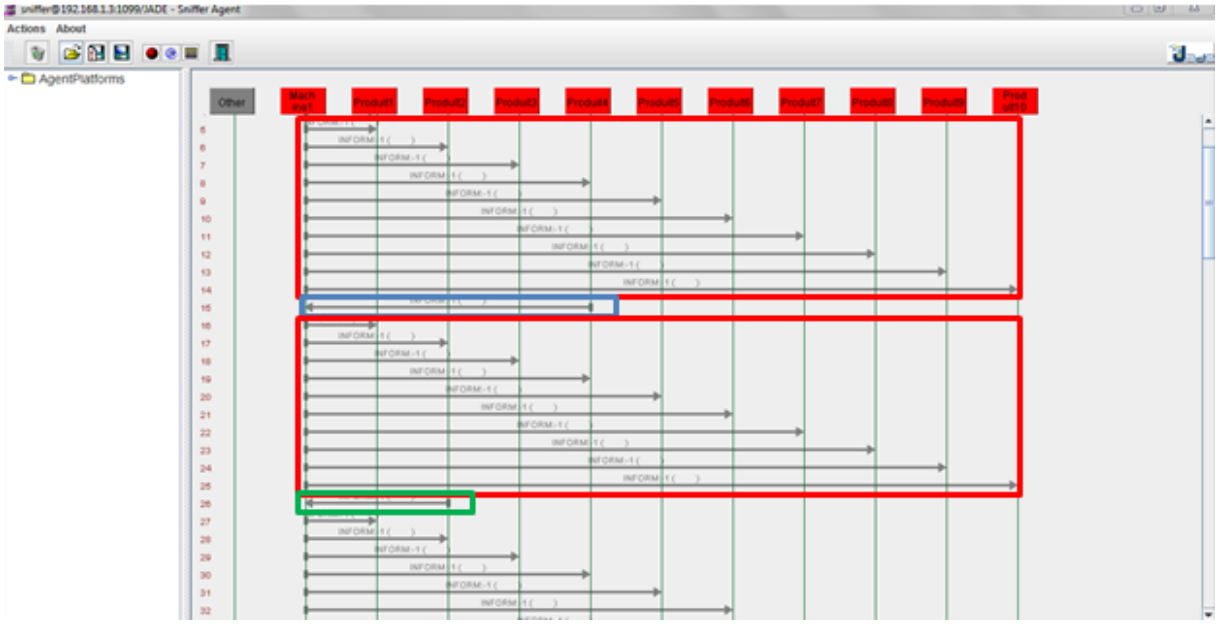


Figure 4.10. Diagramme de séquence(3).

Encadrés en rouge : l'agent machine1 envoie un message **ACL** à tous les agents de type **INFORM** qui contient l'état de la machine.

Encadrés en Blue : l'agent produit4 envoie un message **ACL** à l'agent machine1 de type **INFORM** qui contient la date de début de production de produit4 de la gamme 4 et le temps de l'opération selon l'ordre de passage.

Encadrés en vert : l'agent produit2 envoie un message **ACL** à l'agent machine1 de type **INFORM** qui contient la date de début de production de produit2 de la gamme 2 et le temps de l'opération selon l'ordre de passage.

La figure 4.11 et 4.12 représente le diagramme de séquence de 10 produits de 10 gammes de diffèrent ordre de passage :

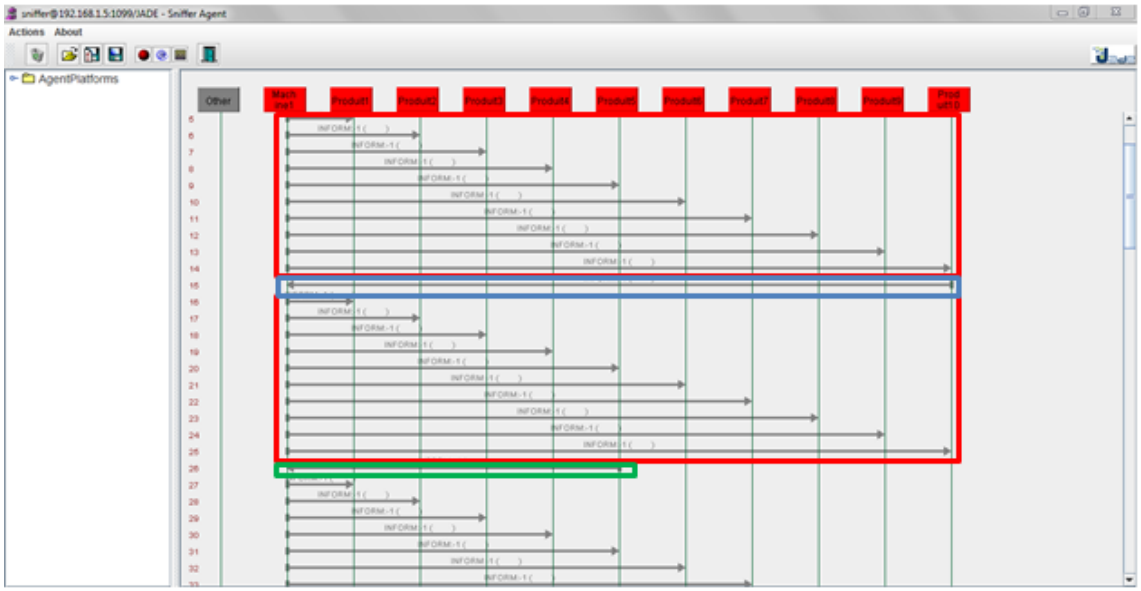


Figure 4.11. Diagramme de séquence(4).

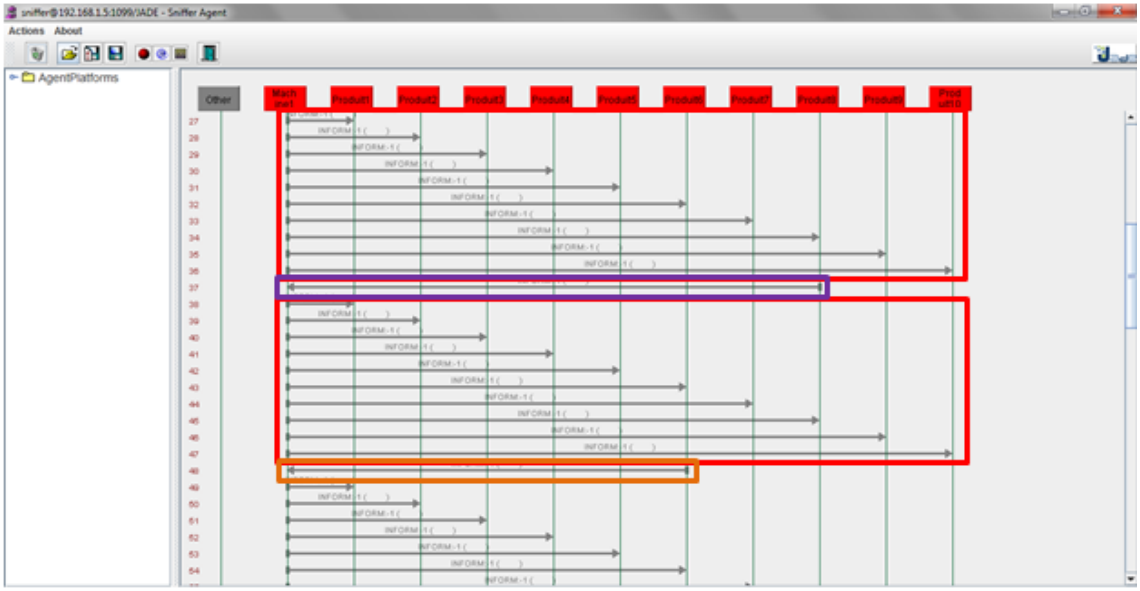


Figure 4.12. Diagramme de séquence(5).

Encadrés en rouge : l'agent machine1 envoie un message **ACL** à tous les agents de type **INFORM** qui contient l'état de la machine.

Encadrés en Blue : l'agent produit10 envoie un message **ACL** à l'agent machine1 de type **INFORM** qui contient la date de début de production de produit10 de la gamme 10 et le temps de l'opération selon l'ordre de passage.

Encadrés en vert : l'agent produit5 envoie un message **ACL** à l'agent machine1 de type **INFORM** qui contient la date de début de production de produit5 de la gamme 5 et le temps de l'opération selon l'ordre de passage.

Encadrés en mauve : l'agent produit8 envoie un message **ACL** à l'agent machine1 de type **INFORM** qui contient la date de début de production de produit8 de la gamme 8et le temps de l'opération selon l'ordre de passage.

Encadrés en orange : l'agent produit6 envoie un message **ACL** à l'agent machine1 de type **INFORM** qui contient la date de début de production de produit6 de la gamme 6 et le temps de l'opération selon l'ordre de passage.

4.6 L'exécution du simulateur et SMA :

La figure 4.11 présente une partie de l'exécution de notre simulateur sachant que celui qui illustre le fonctionnement de notre système multi-agents.

Dans notre simulation les temps de transport sont supposés négligeables, les temps de préparations des machines sont initialisé à Ts = 0, les machines ne sont pas polyvalentes, les pannes ne sont pas prises en compte dans notre simulation de plateforme.

Figure 4.13. Exécution du Programme de simulation de la plateforme.

Dans cette exécution nous constatons que :

Dans un premier temps la machine 1 se prépare, quand elle est prête l'opération sur le produit 1 commence à td=0 min. Quand toutes les opérations seront finies sur tous les postes, la machine 1 sera libre et de même pour les autre machines.

Lorsque toutes les machines sont disponibles, et après un temps de préparation, les machines sont prêtes, le produit 1 se déplace au poste 2 pour commencer la deuxième opération. Le produit 2 sera au poste 1.

Quand les opérations sont finies, et les machines sont disponibles et prêtes, le produit 1 est transporté au poste 3, le produit 2 déplace au poste 2 et le produit 3 sur poste 1 et ainsi de suite.

La figure 4.12 illustre l'exécution de notre SMA :

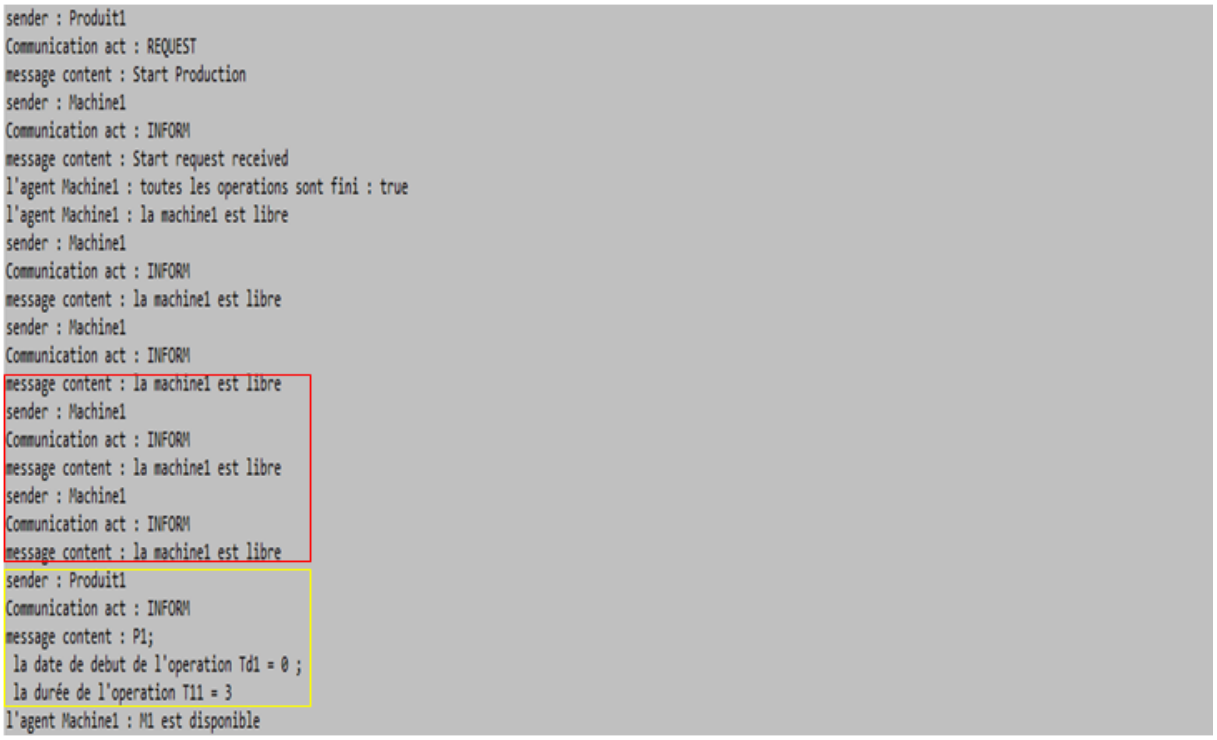


Figure 4.14. Exécution de SMA.

Le cadre en rouge indique les messages envoyés depuis l'agent machine 1 à tous les agents produits pour informer que la machine 1 est libre.

Le cadre en jaune indique les messages envoyés depuis l'agent produit 1 qui contient la date de début d'opération de produit1 de la gamme 1 et la durée de l'opération.

4.7 Résultats finaux :

Afin de calculer le **makespan** (temps total de production), le **Cmax** est définit par :

Cmax $=\max_{i\in\{1,\ldots,N\}}\{S_{iM}+P_{iM}\}$.

P_{iM} = durée d'exécution du produit i sur la machine M.

S_{iM} = la date de debut du produit i sur la machine M.

La figure 4.13 représente les résultats de la simulation de 10 produits de 4 gammes différents :

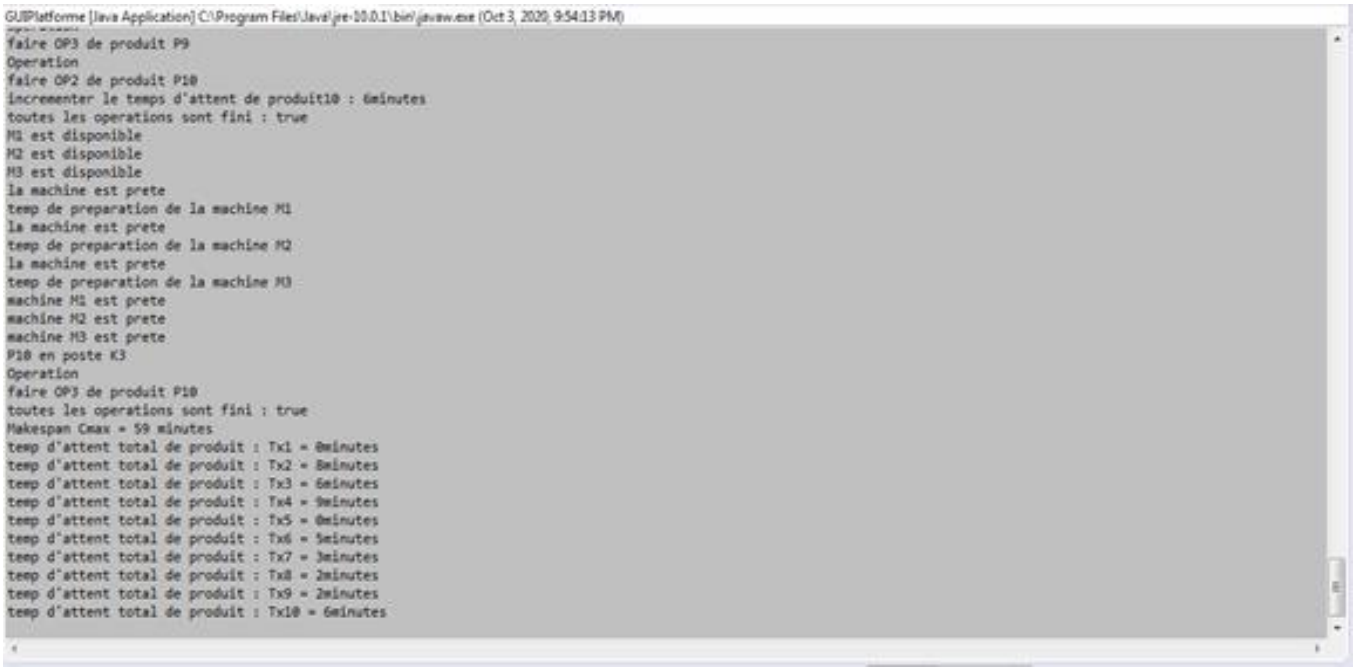


Figure 4.15. Résultat de la simulation.

Le tableau 4.3 représente les résultats finaux de la simulation de 10 produits de 4 gammes :

Column1	Column5	Column2	Column3	Column4
les prdouits	les gammes	temp total d'execution	temps mort	date de fin
produit1	1	149	4	153
produit2	2	175	13	242
produit3	3	75	129	341
produit4	4	185	86	424
produit5	2	175	96	513
produit6	2	175	86	602
produit7	3	75	202	701
produit8	4	185	82	780
produit9	1	149	45	796
produit10	3	75	69	845
			Makespan	Cmax = 845 second

Tableau 4.3. Tableau des résultats (1).

Le tableau 4.4 représente les résultats finaux de la simulation de 10 produits de 10 gammes :

Column1	Column5	Column2	Column3	Column4
les prdouits	les gammes	temp total d'execution	temps mort	date de fin
produit1	1	149	0	149
produit2	2	175	13	242
produit3	3	75	129	341
produit4	4	185	59	397
produit5	5	222	22	486
produit6	6	151	93	585
produit7	7	212	36	645
produit8	8	121	125	732
produit9	9	89	115	789
produit10	10	207	1	853
			Makespan	Cmax = 853second

Tableau 4.4. Tableau des résultats(2).

Le tableau 4.5 représente les résultats finaux de la simulation de 10 produits de 10 gammes de diffèrent ordre de passage :

Column1	Column5	Column2	Column3	Column4
les prdouits	les gammes	emp total d'executio	temps mort	date de fin
produit4	4	185	0	224
produit2	2	175	13	313
produit1	1	149	129	369
produit5	5	222	59	458
produit6	6	151	22	557
produit7	7	242	93	617
produit10	10	207	36	681
produit9	9	89	125	741
produit8	8	121	115	764
produit3	3	75	1	813
			Makespan	Cmax = 813second

Tableau 4.5. Tableau des résultats(3).

Le tableau 4.6 représente les résultats finaux de la simulation de 10 produits de 10 gammes de diffèrent ordre de passage :

Column1	Column5	Column2	Column3	Column4
les prdouits	les gammes	emp total d'executio	temps mort	date de fin
produit10	10	207	21	228
produit5	5	222	8	317
produit8	8	121	103	388
produit6	6	151	108	487
produit4	4	185	84	586
produit7	7	212	65	665
produit1	1	149	112	748
produit9	9	89	130	805
produit2	2	175	54	894
produit3	3	75	120	943
			Makespan	Cmax = 943second

Tableau 4.6. Tableau des résultats(4).

4.8 Interprétation des résultats :

Nous avons exécuté une simulation pour le cas de 10 produits de 4 gammes différentes.

Dans le tableau 4.3 nous avons 2 produits de la gamme 1, 3 produits de la gamme 2, 3 produits de la gamme 3 et 2 produits de la gamme 4. Les produits de chaque gamme ont le même temps d'opération.

Les produits de Chaque gamme ont un temps mort d'opération et la date de fin différents.

Nous avons comme résultat notable **Cmax**=845 secondes, il s'agit de la date de fin d'exécution, appelé aussi **makespan**.

Nous avons exécuté une seconde simulation avec 10 produits de 10 gammes différentes, c'est-à-dire qu'il n'y a pas deux produits qui se ressemblent. Les résultats obtenus pour différents ordres de passage sont donnés dans les tableaux 4.4, 4.5 et 4.6, ce qu'il faut en retenir :

- Tableau 4.4 : le premier ordre de passage donne un Cmax=853 secondes ;
- Tableau 4.5 : le deuxième ordre de passage donne un Cmax=813 secondes ;
- Tableaux 4.6 : le troisième ordre de passage donne un Cmax= 943 secondes ;

Il est clair que le plan d'ordonnancement le plus intéressant est celui illustré par le tableau 4.5 ayant le Cmax le plus couts des trois. Grace à cela notre système sera capable par la suite d'exécuter le plan d'ordonnancement optimal pour n'importe quel groupe de produits, avec tout ce que cela peut impliquer de positif pour la production, en termes de rapidité, de coût, d'économie d'énergie.

4.9 Conclusion :

Nous avons présenté dans ce chapitre la partie Implémentation, simulation et résultats de notre système.

Pour ce faire, nous avons adopté une validation en deux temps. En premier lieu nous avons simulé notre système de production sous java, en deuxième temps nous avons introduit le système multi agent qui représente notre système de pilotage, ainsi que le protocole qui permet la communication entre le programme simulateur et le système multi-agents. Nous avons tout d'abord l'architecture de notre SMA qui est, de type distribuée, ne nécessite pas d'agent superviseur cela permet aux agents de communiquer directement entre eux : pour superviser le programme de simulation et exécuter un plan d'ordonnancement. Ensuite nous avons présenté comment les agents interagissent entre eux et comment ils échangent les différents messages grâce au protocole de communication ACLMessage.

Tout au long de la création et du test de notre SMA, nous avons pu faire :

- Création d'un simulateur de production d'un atelier flow-shop à 3 machines.

- Exécution de plusieurs plans d'ordonnancements selon l'ordre de passage des produits des différents gammes et calcul du **Cmax.**
- La communication entre SMA et le simulateur de la plateforme après avoir implémenter (deux projet java séparés) le protocole de communication socket (client/serveur).
- Les behaviours implémentés dans les agents facilitent l'envoie et la réception des messages entre eux en utilisant ACL message.
- Le diagramme de séquence nous a permis de suivre l'échange des messages entre les agents.
- Le **makespan** est affiché lors e simulation.
- L'ordre de passage est très important pour minimiser la valeur de **makespan.**

Conclusion générale

Le travail présenté dans ce mémoire porte sur le Pilotage d'une Plateforme d'Assemblage Robotisée par les Systèmes Multi Agents, nous avons eu comme objectifs la mise en place du système de pilotage et de supervision multi agent distribué, et la conception d'une architecture SMA capable d'exécuter un plan d'ordonnancement.

La validation du Travail se fait en deux étapes :

La première étape consiste à valider la solution sur le modèle d'un Flow shop synchrone muni de quatre Machines et d'un système de convoyage en configuration rectangulaire de **N** Produits.

Les principaux points réalisés sont :

- Etude de l'architecture de pilotage Distribuée Multi Agents, en prenant en considération le système à piloter qui est un Flow shop synchrone muni de quatre postes et d'un système de convoyage, et de **N** Produits.
- Réalisation d'un programme de simulation de la plateforme, sur lequel on a pu tester et mettre au point notre SMA
- Programmation de l'Architecture Multi Agents sous JADE utilisant l'Editeur Eclipse.
- Etablissement des règles de comportement des Agents Machines et des Agents Produits.

Le travail réalisé durant ce projet a nous permis :

- Enrichir nos connaissances théoriques dans le domaine électronique appliqué à l'instrumentation industrielle, pilotage de production et les systèmes multi-agent.
- Lancer à programmer en JAVA, en utilisant Eclipse IDE.

- Utilisation de protocole de communication, et faire communiquer les agents entre eux.

Le covid-19 nous a ralenti pour terminer notre travail, alors nous proposons comme perspectives :

- Pilotage de la cellule avec prise en compte des aléas de production (priorité sur un produit, annulation de commande...).
- Etablissement d'une Lois d'ordonnancement et d'optimisation de la production et implémentation sur les Agents JADE.

Bibliographie

[1] R. Ohlmann : ˈComprendre l'industrie 4.0et ses défisˈ, industrie 4.0 magazin, 2015.

[2] Frédéric Abbal : ˈIndustrie4.0 l'usine connectéeˈ, Editorial, Septembre 2013. Consulté le 11/03/2020.

[3] W.Wahlster :ˈcyber-physical Poduction systems for Mass customizationˈ, Professor Wolfgang Wahlster CEO of DFKI, German, 2016.

[4] VDE Association for electrical, ˈThe German standardization ROADMAP industrie4.0ˈ Electronic & information technologies, 2014.

[5] L. Sébastien : ˈConception d'un simulateur pour l'optimisation de la commande d'une plate-forme multi-sources représentant un véhicule électrique à hydrogèneˈ, PFE memoire, 2013.

[6] G. Habchi :ˈConceptualisation et modélisation pour la simulation des systèmes de productionˈ,Université de Savoie, 2001.

[7] A. Ferr et F. Barachi : ˈConception et réalisation d'un environnement d'émulation multi-agent pour l'évaluation des approches de pilotage par le produit des systèmes manufacturiersˈ, PFE mémoire, 2010.

[8] J. Ferber : ˈLes systèmes Multi Agent : vers une intelligence collectiveˈ, Inter Editions, 1995.

[9] E. Hind : ˈApproche méthodologique pour l'intégration des systèmes contrôlés par le produit dans un environnement de juste-à-tempsˈ, Application à l'entreprise Trane, SEMANTICSCHOLAR, 2008.

[10] X. Ye :ˈ Modélisation et simulation des systèmes de production : une approche orientée-objetsˈ, 1994.

[11] B. Brahim : ˈmodélisation & simulation sur ordinateurˈ, 2004.

[12] G.Roucairol :ˈLa simulation haute performance au service de la compétitive des entreprisesˈ,2013.

[13] G.Picard, L.Vercouter : initiation à la programmation orientée-objet avec langage Javaˈ,2014.

[14] Cysboy : ˈApprenez à programmer en Javaˈ, 2013.

[15] I,Hammouda : ˈMise en place d'un système de supervision des machines de fabrication dans une unité de production d'emballageˈ,2016.

[16] S.El Bahloul : ˈFlow-shop à deux machines avec des temps de latence : Approche exacte et heuristiqueˈ,2008.

[17] M. Samieh : ˈModélisation du processus de pilotage d'un atelier en temps réel à l'aide de la simulation en ligne couplée à l'exécutionˈ, 2009.

[18] H. Mazouz : ˈIngénierie des protocoles d'interaction des systèmesˈ, 2001.

[19] M. Bachir: ˈA Rule-Based Harmony Search Simulation-Optimization Approach for Intelligent Manufacturing Controlˈ, 2015.

[20] J.-P. SANSONNET : ˈSystèmes multi-agentsˈ, 2005.

[21] B.Ferrah, K.Baba : ˈapproche evolutionnaire pour la résolution de problème d'ordonnancement de type job shop flexible dynamiqueˈ,2010.

[22] astrolilli M., Gambardella L. M :ˈEffective neighbourhood functions for the flexible job shop problemˈ, Journal of Scheduling 3, 3–20, 2000.

[23] F.Blondel : ˈGestion de la production, 3 ème éditionˈ, DUNOD, Paris 2004

[24] N.Taghzout :ˈ conception et développement d'un système multi-agents d'aide à la décision pour la gestion de production dynamiqueˈ, 2011.

[25] D.Roudi :ˈ A dynamic multi-agent based scheduling for flexible flow line manufacturing system accompanied by dynamic customer demandˈ, 2015.

[26] D.Bouhalouan : ˈproposition d'une nouvelle architecture des systèmes multi-agents pour le pilotage des systèmes de production application des algorithmes génétiques hybridesˈ, 2009.