SUPERVISION D'UN SYSTEME DE PRODUCTION CONSTITUE DE ROBOTS COLLABORATIFS

Moussa Goumeye Souleymane, M'Hammed Sahnoun, David Baudry, Anne Louis

CONFERE'17
6 ET 7 JUILLET 2017, SEVILLE, ESPAGNE.

SUPERVISION D'UN SYSTEME DE PRODUCTION CONSTITUE DE ROBOTS COLLABORATIFS

Moussa Goumeye Souleymane[1], M'hammed SAHNOUN[1], David BAUDRY[1], Anne LOUIS[1]
[1] LINEACT (CESI), ROUEN, FRANCE

Résumé : En 2011, lors du salon de la technologie industrielle à Hanovre, un nouveau concept d'industrie a été mis en évidence pour la première fois. Ce nouveau concept est celui de l'industrie 4.0, qui constitue une sorte de nouvelle révolution industrielle ayant pour objectifs de mettre en place des usines dites intelligentes pouvant améliorer nettement la productivité et aussi s'adapter aux changements dans la production. Le projet européen COROT se lance dans cette optique d'industrie du futur. Ainsi ce projet consiste à la mise en place d'un atelier de production flexible, avec des robots collaboratifs autonomes évoluant dans le même milieu. Ce papier, propose une méthode qui permettra de superviser et de faire fonctionner d'une façon optimale cette flotte de robots collaboratifs.
Mots clés: Robots collaboratifs; Industrie 4.0; Dynamically Integrated Manufacturing System; Robot Operate System; Atelier flexible

1 INTRODUCTION

L'utilisation des robots mobiles ou des manipulateurs mobiles pour le transport des produits est une pratique qui commence à émerger dans les ateliers de productions flexibles afin de réduire la pénibilité des tâches répétitives et sans grande valeur ajoutée. Ceci ouvre des perspectives intéressantes dans le contexte de l'usine de futur [1]. Les ateliers flexibles (flexible manufacturing systems FMS) sont connus pour leur complexité et la difficulté de leur gestion d'où la nécessité de développer des systèmes de supervision pouvant gérer ce niveau de complexité.

La supervision est une technique industrielle de suivi et de pilotage informatique de procédés de fabrication automatisés. Dans le monde industriel deux principales approches furent développées au fil des années. La première est l'approche centralisée ou traditionnelle. Dans ce type d'approche on se contente de collecter et d'afficher des informations en vue de leur interprétation et utilisation pour le contrôle du système par l'expert humain [2]. C'est le cas du système de supervision SCADA (Supervisory Control and Data Acquisition) qui consiste à la centralisation des données, et à leur représentation semi-graphique sur des postes de pilotage. Dans ce système on collecte des données des machines se trouvant dans l'installation, pour les transmettre par la suite à un ordinateur central qui va contrôler et superviser l'installation [3]. SCADA est un outil d'aide à la décision concernant les procédés de fabrication, les choix stratégiques de conduite. Il est considéré aussi comme un facilitateur dans la supervision en temps réel et à distance des procédés de production embarqués sur des plateformes distantes [4].

Ce type de système de supervision est très utilisé dans les FMS, vu leur sa performance et son coût intéressant pour les industriels. Seulement avec l'avènement de l'industrie 4.0, qui rend possible la mise en place de systèmes complexes, les approches centralisées ou sous forme de grille, se voit atteindre leur limite avec de telles configurations. [5].

Afin d'éviter ce genre de limitations, des architectures alternatives basées sur des systèmes distribués ont montré leur efficacité [5]. Ce genre d'architecture propose des solutions distribuées, adaptatives et autonomes du système de production. La différence principale, entre les systèmes conventionnels et ces architectures distribuées, réside dans leur capacité d'adaptation rapide aux changements internes et externes que subit le système. Ceci est possible, grâce à leur modularité et leurs capacités d'auto-organisation [6] [7]. Ces architectures font généralement appellent à des systèmes multi-agents (Multi-agent systems MAS) pour leur structuration.

Ainsi les systèmes multi-agents sont utilisés pour modéliser des systèmes avec des hiérarchies

complexes puisqu'ils peuvent englober une variété d'agents individuels qui prennent des rôles ou des responsabilités spécifiques. Ils sont capables de coordonner leurs connaissances, leurs capacités et leurs objectifs les uns avec les autres pour accomplir leurs tâches [8] [9]. En d'autres termes c'est un système logiciel offrant une plateforme pour implémenter, intégrer et gérer des entités indépendantes et intelligentes.

Divers travaux ont été menés afin de mettre en œuvre des architectures répondant à ces exigences.

Parmi, ces travaux on peut parler de l'ADS2 (Anytime Distributed Supervision of Distributed Systems) [2]. Ce système repose sur une architecture de supervision multi-agent où chaque agent est doté d'une vision locale et de capacités de décision, de diagnostic et de réparation. Cette architecture intègre trois composants : (1) un modèle décisionnel d'agent, (2) un mécanisme distribué de coordination et de resynchronisation de l'état du système par les agents, et (3) un algorithme local à chaque agent qui entrelace les phases de diagnostic et de réparation afin de réduire la taille de l'espace des diagnostics candidats.

De part ces caractéristiques, l'architecture multi-agents proposée est capable d'adapter dynamiquement et automatiquement son comportement à l'état du système supervisé et des liens de communications tout en limitant les échanges d'informations entre agents aux seules situations où cela s'avère nécessaire. Ainsi, et même lorsque les communications ont un coût nul, les agents communiquent uniquement lorsque les dysfonctionnements qu'ils cherchent à résoudre nécessitent l'atteinte d'un consensus, et seulement avec le sous-ensemble d'agents concerné par celui-ci [2].

Outre l'ADS2, on peut parler du Dynamically Integrated Manufacturing System ou en français système de production dynamiquement intégrée(DIMS) qui est un type d'architecture qui fournit des méthodes de modélisation et de contrôle systématique. Aidant ainsi les systèmes de fabrication à utiliser et à organiser de manière optimale et flexible les ressources de fabrication pour faire face aux changements dans l'environnement des entreprises. Le DIMS se fonde sur l'utilisation des systèmes multi-agents pour développer une architecture de modélisation et de contrôle dans laquelle les décisions opérationnelles et de restructuration peuvent être considérées simultanément [6]. Il a été développé par XMEC (Exeter Manufacturing Entreprise Centre). On dit d'un système qu'il est de l'architecture DIMS s'il respecte ces quatre conditions : (1) le système doit être capable de modéliser les hiérarchies complexes des systèmes de fabrication, (2) Il doit éviter un contrôle centralisé afin que les modèles de contrôle des systèmes de fabrication puissent être facilement et rapidement modifiés en réponse à des changements possibles dans les structures physiques, des systèmes de fabrication résultant de la restructuration dynamique du système, (3) une utilisation optimale des ressources de production dans les contraintes structurelles des systèmes de production est requise dans cette architecture, (4) enfin le système doit permettre d'évaluer dynamiquement les structures des systèmes de fabrication en fonction de l'analyse continue des décisions opérationnelles et de les reconfigurer au moment le plus approprié avec les décisions de restructuration les plus appropriées.

Dans DIMS, chaque agent est créé sur la base d'une architecture d'agent générique qui est composée de trois parties comprenant une unité de contrôle, un environnement commun et des sous-agents. Cependant, les sous-agents sont des agents indépendants. Ils sont impliqués dans l'architecture de l'agent uniquement parce qu'il existe des relations hiérarchiques entre les ressources de fabrication représentées par ces sous-agents et la ressource de fabrication représentée par un agent parent.

Cette dernière méthode de supervision distribuée semble plus complète que les autres, car les agents sont structurés d'une telle façon que leur fonctionnement est optimal. En outre c'est une méthode qui a fait ses preuves dans le monde industriel, c'est d'ailleurs pour cela qu'on l'a choisie pour la suite de nos travaux. L'implémentation de ces architectures pour des systèmes de productions flexibles et robotisées demande l'utilisation de plateformes logiciels spécifiques appelées middleware. La littérature scientifique et industrielle contient plusieurs exemples de logiciel de ce type. Nous allons faire une comparaison brève de quelques middlewares afin d'en choisir celui qui permet l'implémentation de DIMS plus facilement.

Le middleware **PyRo**, est un environnement de développement robotique en Python qui offre un support simple pour le développement des applications de robotique. Le but de PyRo est de faciliter le développement des applications robotiques et d'assurer leur portabilité, sans la modification du code [10].

Par ailleurs **Player**, permet l'exécution des mêmes applications aussi bien en simulation que sur des robots réels. Il fournit un Framework de développement supportant différents périphériques

matériels, des services communs requis par les applications robotiques et transfère un contrôleur de la simulation à des robots réels avec le minimum d'effort possible. [11]
On a aussi **MIRO** qui fournit un Framework général pour le développement des applications robotiques. Malgré cette utilisation assez facile et commode de ces middlewares, ils demeurent difficiles à intégrer à un système multi-agent complexe. Par ailleurs le middleware ROS (Robot Operating System) parait plus complet et facilement intégrable au système. [10] [12]

En effet ROS offre une architecture souple de communication inter-processus et inter-machine. Les processus ROS sont appelés des *nodes* et chaque *node* peut communiquer avec d'autres *via* des *topics*. La connexion entre les *nodes* est gérée par un *master*. ROS débute par l'exécution du maître (Master) qui permet à tous les exécutables ROS (Nodes) de se retrouver et de communiquer entre eux [13] [10]. Il est open source, et beaucoup de développeurs l'utilisent. Ce qui fait que beaucoup de travaux et logiciels robotiques ont été développés dessus. Etant donné tous ces avantages qu'il possède et surtout sa facilité de communication inter-machine, on l'a choisi comme middleware à intégrer à DIMS.

Les travaux qu'on a effectués dans ce papier sont réalisés dans le cadre du projet européen COROT. Il se fonde sur le développement d'un système robotisé pour le transport de produits dans un atelier de production flexible, facilement reconfigurable et évolutif. Ce système est composé de robots mobiles collaboratifs, de bras manipulateurs collaboratifs, de machine communicantes, d'interfaces hommes machines au niveau des machines et des opérateurs, etc. L'ensemble du système est contrôlé grâce à un système de supervision distribué.

Le premier objectif du projet est de rendre accessible aux PME (petites et moyennes entreprises) l'utilisation de systèmes automatisés de transport. En outre COROT vise à développer un logiciel de simulation et de supervision distribuée basé pour les systèmes de production flexible utilisant des systèmes multi-agents.
Ce papier est organisé comme suit : La prochaine section propose une solution pour que DIMS soit applicable à la problématique de l'atelier flexible, en utilisant comme middleware ROS. La section 4 décrit comment contrôler et faire communiquer un robot avec une machine. La section 5, propose une solution de supervision de deux robots.

2 ARCHITECTURE DU SYSTEME

En se basant sur l'architecture DIMS, le système peut être modélisé sous forme d'agent parent et d'agents enfants. L'unité de supervision de l'atelier de production est considérée comme l'agent parent. Les bras robotisés et les robots mobiles sont considérés comme des agents enfants. Tous les agents (parent et enfants) ont la même structure. Ils sont constitués de deux parties essentielles : l'unité de contrôle et l'environnement commun. [6]
L'unité de commande se subdivise en quatre parties (figure1) qui jouent des rôles importants dans le fonctionnement du système. Le dispositif de communication inter-agent, la base de connaissances (KB), le contrôleur et le composant de simulation constituent les quatre parties clés de l'unité de commande. Parmi ces parties, la communication entre les agents est assurée par l'organe « Echange entre Agents ». Il permet l'envoi et la réception des informations des différents agents, assurant ainsi l'interaction entre eux. Les informations transmises ou reçues peuvent être des données sur la localisation des agents, la vitesse d'avancement, l'état du robot à savoir s'il est à l'arrêt ou s'il est en activité, sa position. Elles peuvent être aussi les résultats de la simulation pour faciliter la négociation entre agents, ou bien des ordres provenant de la hiérarchie c'est-à-dire l'agent parent. Par contre un agent ne peut pas donner d'ordre à un autre se situant au même niveau que lui, il ne peut qu'entamer une négociation avec lui [8]. Chaque agent peut simuler les scénarii possibles et prendre les décisions optimisées grâce aux prédictions du simulateur. La composante de simulation d'un agent peut être connectée dynamiquement à celles d'autres agents pour former un système de simulation distribué qui peut être utilisé pour évaluer d'autres structures de système. Quant à l'environnement commun des agents, il se compose de trois parties principales également, à savoir une base de données, un gestionnaire d'agent et un facilitateur. La base de données d'un agent est constituée d'informations concernant la capacité de traitement, l'état dynamique et l'état de la ressource de fabrication correspondante, ainsi que les informations d'enregistrement concernant ses agents enfants. L'information d'inscription d'un agent enfants comprend le nom / l'identité, la capacité et l'emplacement physique de la ressource de fabrication associée à l'agent enfant.

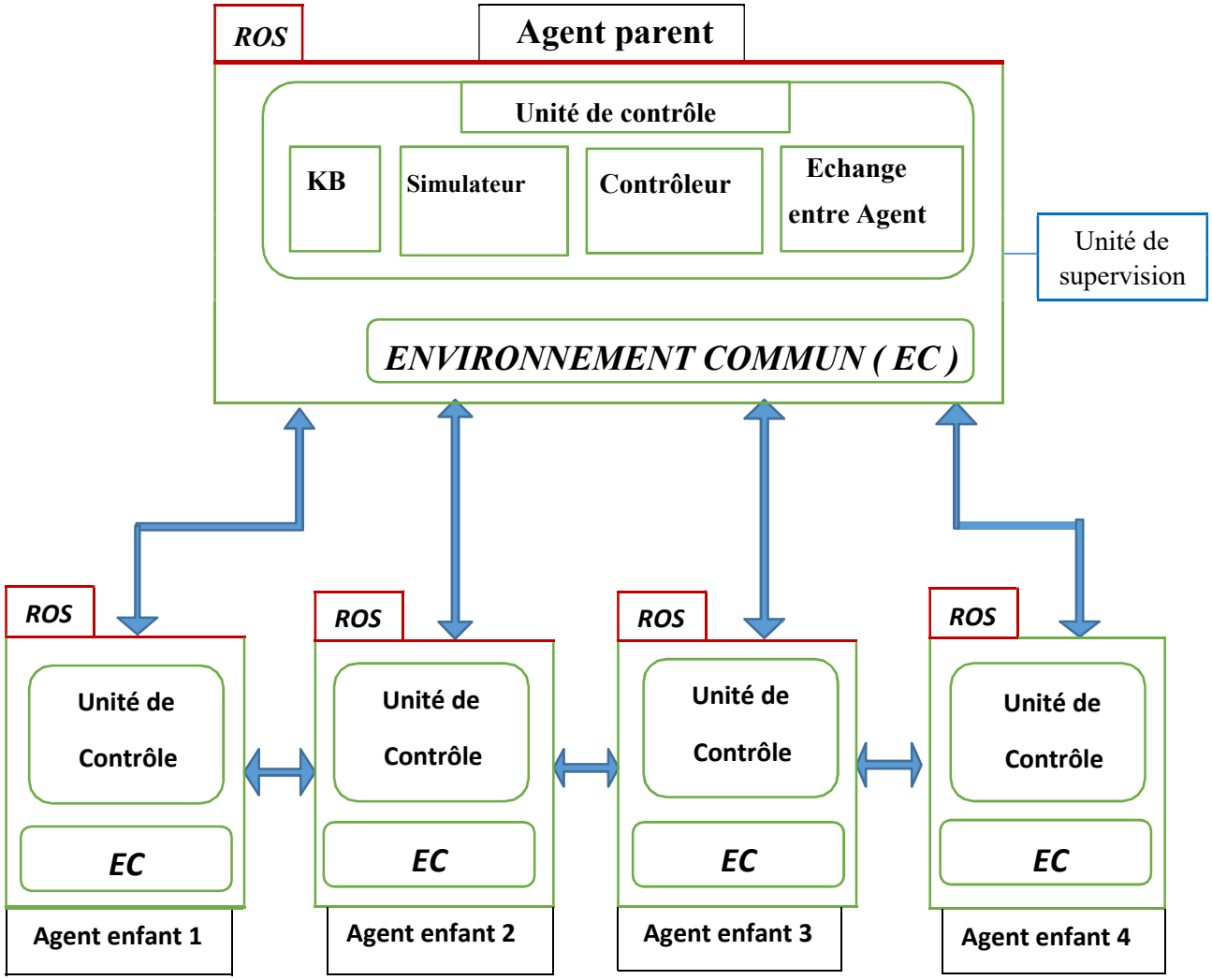

Figure 1. Architecture globale du système de supervision

3 SUPERVISION D'UN SYSTEME MONO AGENT (BRAS ROBOTISE)

Notre système est composé d'un bras manipulateur collaboratif de type Universal robot (UR5). Le but est de superviser le robot à partir d'un ordinateur équipé de ROS. C'est une étape importante dans l'implémentation de l'architecture multi-robot, car elle permet de comprendre comment on peut communiquer et contrôler un robot avec ROS. Pour se faire il a fallu installer le package Universal robot qui comporte des sous-packages tels que : ur_modern_driver, ur_bring, ur_description, ur_gazebo, ur_moveit_config. Ces packages sont importants pour l'expérience, par exemple ur_bringup importe le robot réel sur la machine. Le package ur_modern_ driver installe les pilotes du robot, et est la nouvelle version du package comme le montre le terme « modern ». En effet l'ancienne version nommée ur_driver, n'est compatible qu'avec les anciennes versions 1.x du firmware de universal robot. Pour les versions 3.x , il est impératif de télécharger la version moderne du driver.

Quant au ur_description il contient les fichiers urdf (Unified Robot Description Format) des robots. Ces fichiers permettent une représentation cinématique du robot, et contribuent aussi à visualiser le robot sur la machine.

Ainsi grâce au ficher descriptif, nous arrivons à visualiser le robot sur la machine avec le Ros

Vizualisation (Rviz). Cet outil permet d'envoyer et de recevoir des commandes des nodes. Comme l'illustre les images qui suivront. Le robot est connecté sur le même réseau que la machine grâce à un câble Ethernet. La figure 2 montre le contrôle du robot par la machine, le robot en couleur orange représente le robot virtuel, et le gris, le robot réel importé. La machine envoie des commandes vers le robot. Ainsi on peut agir sur différents paramètres du robot tels que sa vitesse et l'accélération grâce à l'interface de Rviz. En outre il est possible de visualiser la trajectoire que va effectuer le robot réel avec l'envoi des commandes grâce à l'option *Plan.*

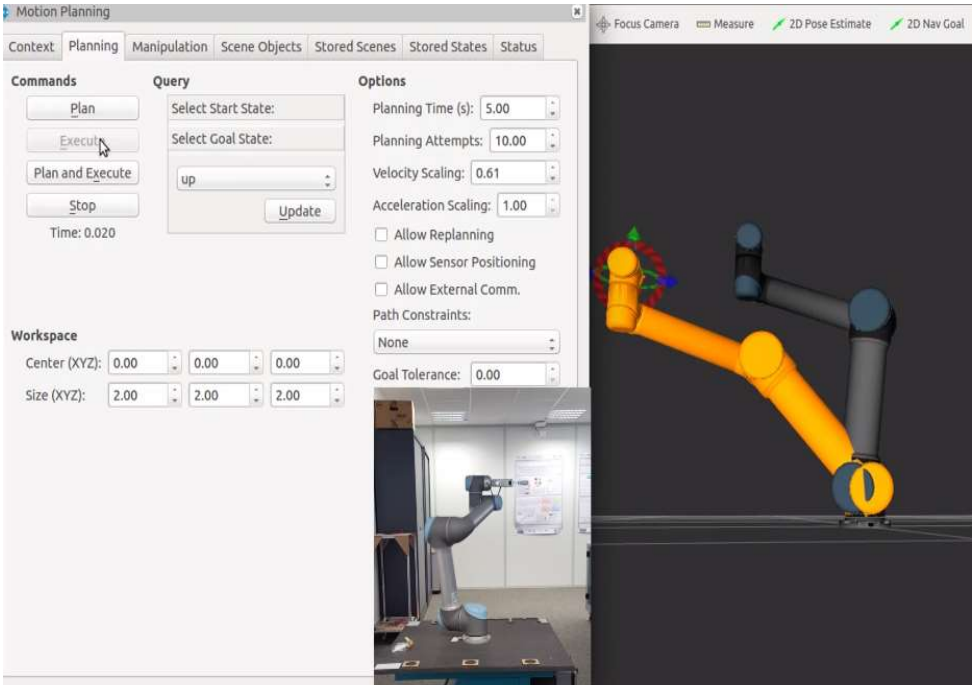

Figure 2. Contrôle du robot via la machine

La figure 3, représente le processus inverse de la figure 2. On agit sur le robot réel pour avoir une visualisation en temps réel sur le robot de la machine. Le robot orange reste statique dans ce cas.

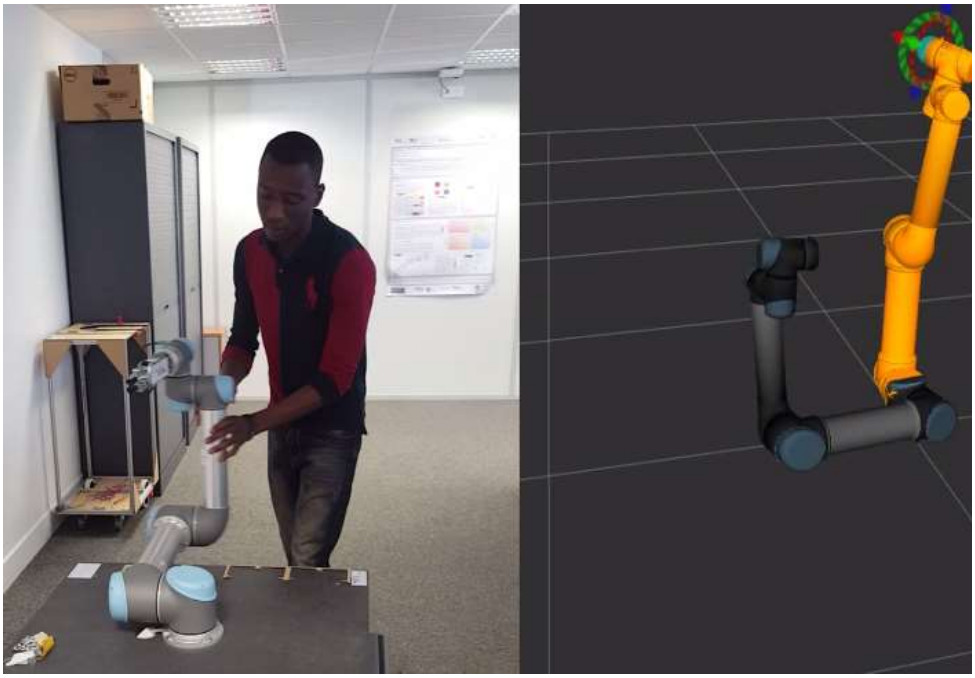

Figure 3. Envoi des commandes du robot vers la machine

4 SUPERVISION D'UN SYSTEME BI-ROBOT (DEUX UR5)

Après la prise en main du robot, nous sommes passés au à la supervision de deux universal robots. La supervision des deux robots passent par l'utilisation du ROS multi-machines, étant donné que chaque robot est représenté dans ce cas par une machine sur laquelle on a préalablement installé ROS. Une troisième machine jouant le rôle de superviseur est aussi utilisée, après lui avoir implémenté ROS comme le montre la figure 5. Toutes les machines étant configurées sur le même réseau, il faut définir la machine superviseuse comme étant le ROS-Master, puis exporter son adresse son adresse IP sur les autres machines pour que celles-ci la reconnaissent comme étant leur Master. Après cette étape les agents peuvent être considérés comme étant sur le même réseau ROS, avec un unique Master. Le processus de communication entre les agents pourrait s'avérer simple, en installant tout simplement le package universal robot sur chaque machine, et faire échanger des informations entre les différents nodes. Mais il existe une subtilité, dû au fait que les packages vont générer les mêmes nodes avec les mêmes noms, ce qui créerait un conflit de namespace. La solution à ce problème est l'utilisation de la variable *name* dans les fichiers descriptifs urdf, rendu possible grâce au xacro (xml macro) qui un langage macro xml. Ainsi nous avons créé un package commun avec une variable *name,* à placer sur chaque machine représentant les robots. La variable est ensuite remplacée par le nom choisit pour le robot, ce qui permet d'éviter le conflit de namespace.

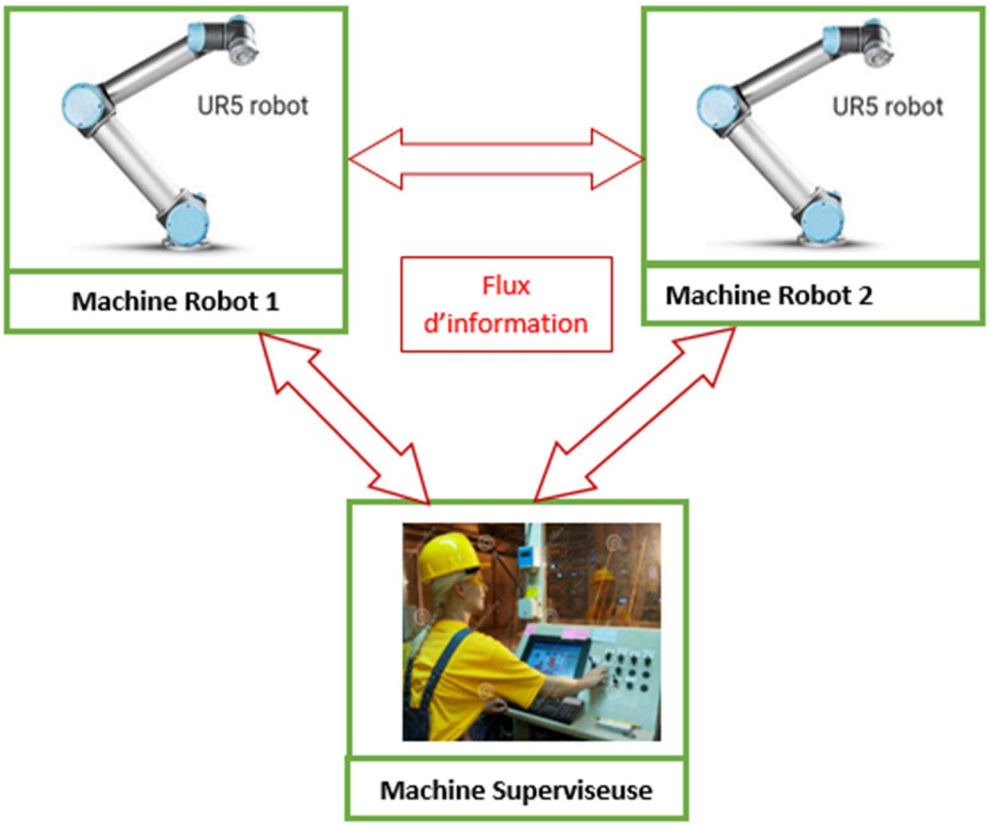

Figure 5. Schéma de la supervision deux robots

La figure 5 montre le schéma du processus. Chaque robot est relié à un ordinateur, dans lequel est adapté le package Universal, avec les namespaces adéquats. Ils échangent ainsi des informations entre eux, mais aussi avec la machine superviseuse, qui contient aussi le même package. Chaque agent effectue ses propres calculs et simulations puis envoie ses trajectoires déjà calculées à la machine superviseuse, ce qui permet de ne pas réduire ses performances. La figure 6, est une image qui illustre la supervision des UR. On peut ainsi voir sur les robots importés sur leurs machines respectives, et l'ensemble sur l'ordinateur superviseur qui se trouve en bas.

Figure 6. Image illustrative des robots avec leurs machines

5 CONCLUSION

La supervision d'un système de production flexible (système multi-agents) doit nécessairement passer par une maitrise et une connaissance de la structure de chaque composant (agent) constituant le système. Nous y sommes parvenus par une étude complète et poussée sur les robots mobiles et bras robotisés. Ainsi une implémentation de ROS a été effectuée sur un ordinateur tenant lieu d'agent superviseur, envoyant des commandes via ROS pour pouvoir commander un bras robotisé. Le robot réagit à ces commandes et renvoie des informations à son tour à l'agent superviseur. L'expérience a ensuite été étendue sur deux robots. La prochaine étape consiste donc à intégrer une intelligence aux robots pour qu'ils puissent collaborer entre eux.

Toutes ces expériences sont rendues possibles grâce aux projets InterregVA COROT.

REFERENCES

[1] Z. Bi, L. Da Xu et C. Wang, «Internet of things for enterprise systems of modern manufacturing,» *IEEE Transactions on Industrial Informatics,* vol. 10, pp. 1537-1546, 2014.

[2] C. Herpson, «Approche multi-agents pour la supervision adaptative des systèmes distribués,» 2012.

[3] G. H. Baker et A. Berg, «Supervisory control and data acquisition (SCADA) systems,» *The Critical Infrastructure Protection Report,* vol. 1, 2002.

[4] Z. BOUNAB, «Etude d'un système de supervision et de contrôle SCADA de la région de transport est RTE Skikda».

[5] D. Trentesaux, «Distributed control of production systems,» *Engineering Applications of Artificial Intelligence ,* vol. 22, n° %17, pp. 971-978, 2009.

[6] N. He, «Agent-based Hierarchical Planning and Scheduling Control in Dynamically Integrated Manufacturing System,» 2011.

[7] J. P. Müller et K. Fischer, «Application Impact of Multi-agent Systems and Technologies: A Survey,» *Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks,* pp. 27-53, 2014.

[8] A. I. Anosike, «Agent-based modelling simulation and control of dynamically integrated manufacturing systems,» 2005.

[9] C. a. S.-B. C. HANACHI, «Introduction aux Systèmes Multi-Agents».

[10] S. Kchir, «Faciliter le développement des applications de robotique,» 2014.

[11] M. Kranz, A. Schmidt, B. Hornler, G. Rigoll, R. B. Rusu, A. Maldonado et M. Beetz, «Sensing technologies and the player-middleware for context-awareness in kitchen environments,» chez *Networked Sensing Systems, 2007. INSS'07. Fourth International Conference on*, 2007.

[12] H. Utz, S. Sablatnog, S. Enderle et G. Kraetzschmar, «Miro-middleware for mobile robot applications,» *IEEE Transactions on Robotics and Automation,* vol. 18, pp. 493-497, 2002.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler et A. Y. Ng, «ROS: an open-source Robot Operating System,» chez *ICRA workshop on open source software*, 2009.

[14] S. Rusell et P. Norvig, «Artificial intelligent: A modern approach,» 2003.

[15] D. Z. Zhang, A. Anosike et M. K. Lim, «Dynamically integrated manufacturing systems (DIMS)—A multiagent approach,» *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans,* vol. 37, pp. 824-850, 2007.

Contact : Moussa Souleymane

Email : smoussa@cesi.fr

Tel : +33 7 58 80 35 75