Université de Caen Normandie
UFR des Sciences
Département Informatique

1ère année de master informatique

Rapport : Projet Multi-agents

Sacha Pronost - 21901956

Decembre 2022

Table des matières

0.1 Architecture générale

Il nous était demandé pour ce projet de créer un système multi-agents, implémentant un atelier de production utilisant des robots pour ses chaînes de montage. Il est simple d'imaginer donc que pour implémenter une telle architecture il nous faudra deux types d'agents distincts, les robots et l'atelier. C'est pourquoi j'ai créé dans mon projet deux classes, une nommée **Atelier**, et l'autre **Robot**, afin de pouvoir implémenter l'ensemble des fonctionnalités nécessaires au bon fonctionnement du système. Il est important de savoir qu'il ne va y avoir qu'un seul atelier de créer, et un certain nombre de robots, ainsi, dans mon cas, j'ai décidé que l'atelier pourrait communiquer avec les robots et inversement, mais que les robots n'auraient pas besoin de communiquer entre eux. C'est pourquoi, l'atelier se chargera dans mon architecture de répartir les tâches aux robots et de gérer les produits, alors que mes robots eux, se chargeront seulement d'appliquer leurs compétences sur leurs produits. A noter que mes robots ne peuvent prendre en fil d'attente que 3 produits maximum afin de répartir au mieux les tâches sur tous les robots.

Cette architecture implique donc que l'atelier doit choisir intelligemment à qui envoyer quel produit, et si un robot n'accepte pas d'en prendre un, il doit tout de même envoyer le produit à un robot compétent. C'est pourquoi l'atelier va créer un score pour chaque robot (en fonction du produit qu'il veut envoyer) afin de pouvoir envoyer le produit au meilleur robot, ou aux deuxièmes meilleures, et ainsi de suite. Cette manière de faire me permet non seulement d'envoyer le produit au robot qui serait le plus efficace sur celui-ci, mais aussi si jamais un robot possède trop de produit, de l'envoyer à l'autre plus performant, ce qui rend le système plus efficace.

0.2 Représentation des produits et des compétences

J'ai décidé de représenter mes produits à l'aide d'une classe Java nommée **produit**. Cette classe me permet de factoriser mon code, et de rendre la gestion des produits plus naturelle et plus compréhensible. Ainsi ma classe produite contient un **nom**, un **boolean** me permettant de vérifier s'il est terminé ou non, et un **HashMap** me permettant de lister les compétences nécessaires pour terminer ce produit. Ce HashMap contient en clé le **nom** de la compétence, et en valeur un **boolean** pour vérifier si cette même compétence est finie ou non. Pour créer un produit il suffit donc de lui indiquer son nom, ainsi qu'une liste de compétence (il est possible dans mon programme de rentrée n'importe qu'elle produit souhaité dans le fichier de configuration, en faisant cela l'agent atelier se chargera de créer tous les produits indiqués). En plus de factoriser mon code, le fait de créer une classe pour mes produits me permet de rendre ces objets sérialisables, et ainsi de les donner par message au robot et à l'atelier, ce qui me permet de me faciliter la tâche pour l'échange de messages dans le système.

Comme expliqué précédemment, chaque robot se voie attribuer un certain nombre

de compétences, ainsi qu'un degré de compétence permettant de différencier les robots entre eux, et de leur donné a chacun leur spécificité. Pour implémenter cette notion, j'ai décidé d'intégrer pour chacun des robots un **HashMap** représentant les compétences de celui-ci. Ce dictionnaire aura pour clé le nom de la compétence que possède le robot, et pour valeur un **Float** entre 0 et 1 permettant de représenter le degré de compétence que le robot possède. Ainsi si le robot doit appliquer sa compétence sur un produit, le temps d'attente sera plus ou moins élevé en fonction de son degré de compétence. (Il est aussi possible de régler le temps moyen que les robots vont prendre pour appliquer une compétence dans le fichier de configuration de mon programme)

0.3 Comportement et protocoles de communication

Chacun de mes agents possède plusieurs comportements que je vais lister ici.

Pour mon agent **Atelier** :

— Comportement **sendProduct** : Il s'agit d'un comportement du type **TickerBehaviour**, qui me permet dans sa globalité de générer pour un produit les scores de chaque robot, de le stocker, et d'envoyer un message contenant le produit sérialisé au robot avec le meilleur score. Il faut savoir que tant que le robot ne répond pas, ou qu'aucun autre robot n'a accepté de faire le produit, le score ne sera pas recalculé. sendProduct me permet aussi de vérifier si tous les produits on était finie ou non, et ainsi de stopper l'agent si tous les produits ont été finis.

— Comportement **receptionMessage** : Il s'agit d'un comportement du type **CyclicBehaviour**, qui me permet de réceptionner les messages de l'agent. L'agent peut recevoir trois types de message différents, soit une **acceptation** d'un produit envoyé à un agent par le comportement précédent, auquel cas il supprime le produit de sa liste, et il nettoie le score des agents pour ce produit, soit un **refus** de produit, dans ce cas il envoie un message à l'autre agent avec le meilleur score pour le produit, soit un message de **rendu** de produit, auquel cas il va vérifier si le produit est fini, si c'est le cas il l'ajoute dans la liste des produits finie, sinon il l'ajoute dans sa liste de produits.

Pour mon agent **Robot** :

— Comportement **applySkills** : Il s'agit d'un comportement du type **TickerBehaviour**, qui me permet d'appliquer le maximum de compétences du robot sur un des produits qu'il possède en liste d'attente. Il faut savoir que le temps d'application de la compétence est plus ou moins long en fonction du degré de compétence que possède le robot. A la fin de ce comportement, le robot va renvoyer à l'atelier le produit, et le supprimer de sa liste de produits.

— Comportement **receptionMessage** : Il s'agit d'un comportement du type **CyclicBehaviour**, qui me permet de recevoir les messages de l'agent. Un robot ne peut recevoir qu'un type de message, une demande de produit. Ainsi si le robot possède

déjà trois produits dans sa liste d'attente, il ne va pas accepter la demande, et il va répondre à l'atelier avec un message du type **ACLMessage.REFUSE**. Si au contraire il peut récupérer le produit, il va l'ajouter dans sa liste de produits, et renvoyer un message du type **ACLMessage.ACCEPT_PROPOSAL** a l'atelier.