Systèmes Multi-Agents: Chaîne de production

Cong-Minh Dinh

April 1, 2019

Abstract

Ce rapport a pour objectif de décrire la mise en œuvre d'une modélisation des principaux éléments de la Chaîne de production par Jade, sa visualisation de l'exécution sur une interface graphique avec l'aide de JPanel.

Contents

List of Figures

1 Introduction

Un système multi-agents est un système constitué d'un ensemble d'acteurs dans un environnement donné et de leurs interactions. Le système est déterminé par l'activité des agents.

Un agent peut être décrit comme autonome car il a la capacité de s'adapter lorsque son environnement change. Un système multi-agents est constitué que plusieurs agents existent en même temps, partagent des ressources communes et communiquent entre eux (Fig 1). La question clé dans les systèmes multi-agents est de formaliser la coordination entre les agents.

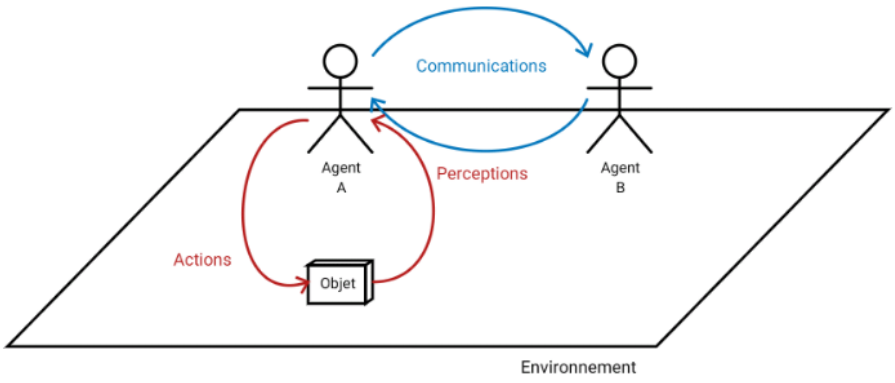

Figure 1: Schéma théorique d'un système multi-agents

Ce projet vise à construire une ligne de production. Pour cela, nous allons utiliser la plateforme Jade. Il s'agit d'une plate-forme distribuée d'agents en fonctionnement asynchrone (chaque agent est un thread). Chaque agent détermine ses actions en utilisant les comportements qu'il adopte à son tour. Les agents peuvent envoyer et recevoir des messages, mais ils peuvent également déclencher d'autres comportements, voire en créer de nouveaux.

Ce rapport fait le tour de notre travail mais n'entre pas nécessairement dans le centre du déploiement, ils sont commentés sur le code.

2 Description du problème

La chaîne de production regroupe toutes les opérations de fabrication nécessaires, à savoir la réalisation d'un produit fabriqué, des matières premières à la mise sur le marché.

Dans ce système, on ajoute le type et la quantité de produits à l'entrée, via le processus de transport, le changement d'emplacement, la destruction des produits cassés... avant l'expédition à sortie. Ce processus nécessite donc une cohérence claire et une communication importante entre les parties de la chaîne pour assurer son efficacité. L'application de SMA est donc très appropriée et contribuera à améliorer la qualité de la chaîne.

Dans la modélisation d'une chaîne de production, il est nécessaire de définir les agents et les relations entre eux: l'agent Produit, l'agent Transformateur, l'agent Vérificateur, l'agent Ligne, l'agent Entrée et l'agent Sortie.

En raison de la limitation de temps, on construit un système plus simple mais ayant tous les mêmes une fonction similaire, décrit en particulier la relation entre l'agent de Transformation et Line, cf *Section 3*. Ce choix de modèle simple aide également à construire l'interface dans la *Section 4*.

2.1 Structure globale du code

Le programme comporte les packages suivants:

1. agents: contient tous les agents.
2. behaviour: contient tous les comportements liés aux agents.
3. graphique: contient les classes pour la représentation graphique de la simulation.
4. main: contient la classe *Lanceur* pour créer les agents et ses comportements pour lancer la simulation, et *LireFichier* pour la configuration d'une chaîne de traitement.

3 Modélisation multi-agent

3.1 Actions des agents

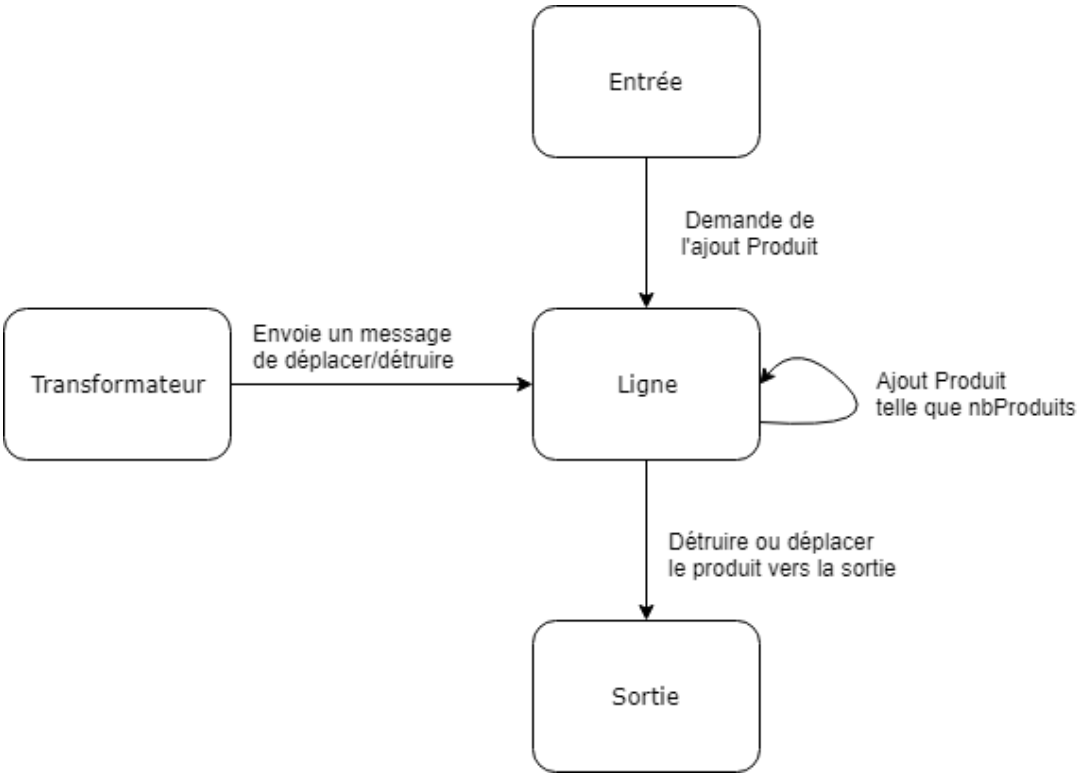

Figure 2: Protocoles d'interaction entre les agents

On utilise un modèle simplifié en raison d'une limite de temps pour réaliser la modélisation de la chaîne de production. Les comportements sont décrits comme sur le schéma.

L'agent Entrée demande à l'agent Ligne d'ajouter des produits. L'agent Ligne gère une liste de produits et ajoute des produits jusqu'à la limite de la capacité de la chaîne.

Lorsque l'agent Transformateur est créé, il envoie un message à l'agent Ligne pour déplacer ou détruire le produit. L'Agent Ligne reçoit le message et détruit ou déplace les produits vers la Sortie de la ligne de production.

L'agent de Ligne est le centre de notre modèle, exécutant les principales activités de la chaîne.

1. *AjouterProduit*: L'agent Ligne ajoute des produits.
2. *EnvoieMessage*: L'agent Transformateur envoie des messages pour le changement de position dans la liste de produits.
3. *MessageChangementPosition*: L'agent Ligne reçevoit les messages et effectue les déplacements de position.
4. *MessageDestruction*: L'agent Ligne reçevoit les messages et détruit des produits.

En réalité, l'agent doit travailler pendant un certain temps pour obtenir le résultat. Dans notre simulation, on définit alors le temps pour la fabrication d'un produit est 1500 et le temps pour son déplacement ou sa destruction est 1000.

4 Visualisation de l'exécution

Afin d'avoir une représentation graphique claire des différentes parties de la chaîne pour suivre l'exécution du système, on a choisi de définir une interface classique en construisant un agent d'interface qui recevra les messages d'autres agents et mettra à jour une interface simple. Pour ce faire, l'extension JPanel est choisie pour ses méthodes associées au dessin de rectangles et de lignes colorées. On s'assure les conditions ci-dessous pour créer l'interface:

- que tous les agents connaissent l'agent interface pour lui envoyer des messages d'information. Pour simplifier le problème tout en conservant l'objectif du projet, nous avons choisi l'interface Line pour dessiner l'interface.
- de stocker les informations (état des machines, position des produits sur les lignes) pour pouvoir afficher l'état du système à chaque instant (et à l'aide d'un repaint après chaque nouveau message reçu)
- d'ajouter dans les agents ligne et machines des informations de position (en coordonnées cartésiennes) qui permettront de dessiner la chaîne de production.

Lors de la création de l'interface en Java, on a besoin d'une classe *Dessin* contenant toutes les informations nécessaires pour dessiner l'interface de visualisation. Dans cette classe, on peut trouver tous les paramètres tels que la largeur, la longueur... des trois blocs Entrée,

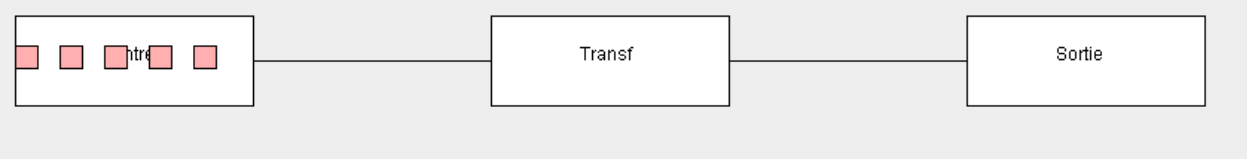

Figure 3: L'interface de visualisation de la ligne de production - Début

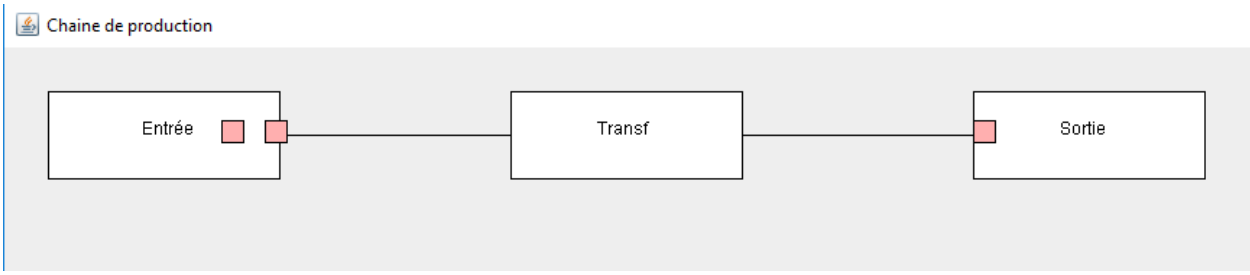

Figure 4: L'interface de visualisation de la ligne de production - Fin

Transf, Sortie, la distance les séparant (i.e les traits) et la couleur dessinée. Après chaque composant, on ajoute également un *ReDessiner* pour mettre à jour le dessin.

L'interface réalisé a été bien mise à jour après chaque composant de l'agent: ajout de produit, déplacement de produit, destruction de produit (Fig 3 et 4).

5 Configuration aisée d'une chaîne de traitement

L'une des étapes nécessaire du projet consiste à charger la liste de configuration de la chaîne à partir d'un fichier *.txt*. Pour être transparent, on définit un langage simple pour charger les configurations de la chaîne. La configuration ci-dessous est stocké dans data.txt:

```
transf:transfo-1;[produit1:1];[ligne2:produit2:2];2000;.02;(200,100)
transf:entree-1;[];[produit1:1];0;0;(100,100)
transf:sortie-1;[produit2:1];[];0;0;(300,100)
ligne:ligne1;entree-1;transfo-1;5
ligne:ligne2;transfo-1;sortie-1;1
```

Avec la syntaxe définie comme la suivante:

```
Type: Paramètres
```

Le *type* est soit *transf* (machine de transformation ou entrée), soit *ligne*, en particulier:

- Pour les lignes: Les paramètres sont le nom de la ligne, la machine source, la machine cible et la capacité.
- Pour les machines: Les paramètres sont le nom de la machine, la liste des entrées, la liste des sorties, le temps de traitement, la probabilité d'erreur et la position dans l'interface de visualisation.

- la liste des entrées: est composée de n éléments séparés par une virgule. Chaque élément est un couple (2 parties séparées par deux points) indiquant le type de produit attendu et la quantité de produit requise pour une transformation.
- la liste des sorties: est composée de n éléments séparés par une virgule. Chaque élément est un triplet (3 parties séparées par deux points) indiquant le nom de la ligne sur laquelle le produit apparaît, le type de produit obtenu et la quantité.

Il est simple d'extraire l'information *type*. Les paramètres, séparés par des points-virgules, dépendent du type. Il faut donc plus de temps pour traiter ces données.

Ainsi, on stocke toutes les machines dans une map et toutes les lignes dans une autre map. Ensuite, on combine ces deux cartes dans une hashmap pour faciliter la récupération des données. Cette implémentation est écrite dans le fichier *main.LireFichier*.

6 Conclusion

Dans ce projet, on a développé un prototype d'application qui permet d'effectuer une modélisation simple sur la chaîne de production. Le modèle développé dans ce projet n'est pas la meilleure solution, mais il nous permet également d'imaginer comment exploiter un système multi-agents dans la vie réelle.

Dans un délai limité, on ne pouvait pas effectuer suffisamment de simulations. Cependant, l'avantage de l'ajustement par système multi-agents est que la simulation est très légale, permettant aux utilisateurs de personnaliser leur modèle en fonction de leurs besoins et de leurs métadonnées. L'interface qu'on a développée aide également à visualiser le système et facilite la simulation pour les utilisateurs.