Automne 2013

Chaîne de montage en anneau

Rapport Projet LO41

Hans Sébastien DEMANOU <hans.demanou@utbm.fr>
Kleverson PEREIRA DE SOUSA <kleverson.pereira-de-sousa@utbm.fr>

Table des matières

Introduction

Dans le cadre de l'UV de LO41 "Architecture et utilisation des systèmes d'exploitation", il est demandé à l'étudiant d'effectuer un projet d'une durée de 4 semaines en fin de semestre.

Notre travail a pour objectif de réguler l'approvisionnement de composants matériels et la gestion du processus de fabrication d'une gamme de produits. Nous nous intéressons à la définition et la mise en œuvre d'une chaîne de montage « en anneau » pilotée par des robots. Sa particularité est que la chaine à la forme d'un anneau circulaire de 16 sections pouvant contenir soit des composants servant à la fabrication de produit, des produits en cours de fabrication ou des produits finaux.

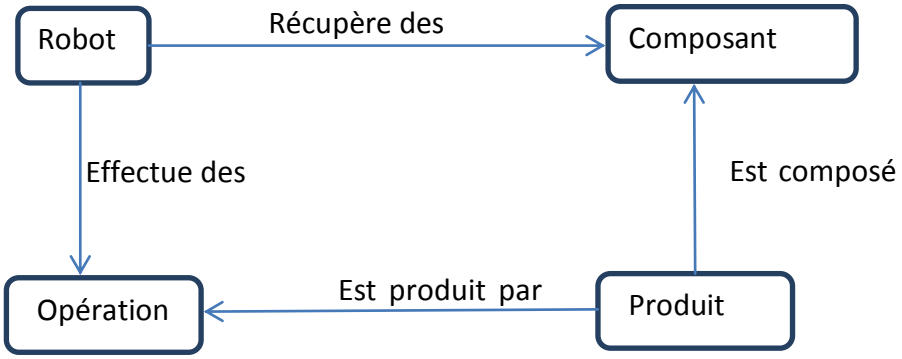
Dans notre projet nous avons tenté de mettre en place un système informatique qui se rapproche le plus de la réalité. Nous avons utilisé un maximum d'outil qui nous ont été fournis durant notre apprentissage.

Dans ce rapport nous allons présenter tout d'abord l'aspect technique et les contraintes imposées pour la réalisation de notre programme. Ensuite nous allons expliquer les démarches choisies pour la mise en place de notre système informatiques ainsi que les outils utilisés. Enfin, nous allons présenter le réseau de pétri qui présentera différentes situation d'inter-blocage ainsi que le diagramme de séquence qui présentera la séquence de fonctionnement.

I - Compréhension du sujet

Il s'agit de réaliser un système qui simule la production de produits manufacturés. Nous devons présenter la production de 4 **produits** différents par 6 **robots**. Pour arriver à la production des 4 produits le système doit réaliser des **opérations** différentes sur des **composants**.

Tout cela semble très complexe à comprendre, nous allons tenter de simplifier la compréhension par un diagramme.



On peut aussi présenter, sous formes de tableau, la quantité de composants pour chaque produit, ainsi que les opérations que les robots peuvent effectuer dans les deux modes de fonctionnement (mode normal ou dégradé).

Nom Produit	Séquences	Nombre de Produits	Nombre de composants nécessaires au démarrage T0 de l'activité
Prod1	Op1 – Op2 – Op3 – Op5	10	3 C1
Prod2	Op2 – Op4 – Op1 - Op6	15	3 C2
Prod3	Op1 - Op3 – Op5 – Op1 – Op3	12	1 C3
Prod4	Op4 – Op6 – Op1	8	2 C4

Fig1 : Processus de fabrication des produits

Nom Robot	Mode		Type de produits modulo Mode	
	Normal	Dégradé	Normal	Dégradé
R1	Op1	Op1 – Op2 –Op5	Prod1, Prod2, Prod3, Prod4	Prod1, Prod2, Prod3, Prod4
R2	Op2	Op2 – Op1	Prod1, Prod2	Prod1, Prod2, Prod3, Prod4
R3	Op3	Op3 – Op4 – Op6	Prod1, Prod3	Prod1, Prod2, Prod3, Prod4
R4	Op4	Op4 – Op3	Prod2, Prod4	Prod1, Prod2, Prod3, Prod4
R5	Op5	-	Prod1, Prod3	-
R6	Op6	-	Prod2, Prod4	-

Fig2 : Mode opératoire des robots

Dans le mode normal les robots n'ont seule opération à réaliser. Cependant, ont mode cascadé, les robots ont la possibilité de *jouer* avec une liste d'opérations possible. Un robot devient donc multitâche en mode cascadé. Le mode cascadé permet de gérer une éventuelle panne dans la chaine de fabrication ; si un robot tombe en panne en mode normal, un autre qui est susceptible de réaliser la même opération passe en mode dégradé.

Dans notre simulation, la chaine de fabrication présente un nombre de composant minimum pour la fabrication de 10 produits1, 15 produits 2, 12 produits 3, et 8 produits 4. Cela revient à un stock de composants minimum de 30 composants 1, 45 composants 2, etc. Cela devra être modifié si un robot tombe en panne et qu'il détient des composants en stock.

II – Technique

Dans cette partie nous allons présenter les différents outils techniques utiliser pour a réalisation de notre simulation. Tout d'abord, nous avons développé plusieurs processus qui simulent le fonctionnement de différents *agents* du système. La synchronisation a été assurée par des sémaphores.

II.1 - L'anneau

Plus précisément, l'anneau est un processus qui initialise une mémoire partagée. Dans cette mémoire, nous avons défini une structure nommée *Anneau* caractérisée par un identifiant (son PID), un tableau de 16 cases qui simulent les 16 sections de l'anneau circulaire, un tableau définit les différentes connexions des robots et du serveur sur l'anneau. La rotation de l'anneau est simulée par la permutation du contenu de différentes cases du tableau. La cadence de rotation est définie en milliseconde (par défaut 2000 millisecondes). Elle est initialisée en argument de la ligne de commande du lancement du processus *anneau*.

Après chaque rotation, le processus *anneau* envoie un signal SIGUSR1 à tous les *agents* connectés sur la mémoire partagée.

II.2 – Le serveur

Le serveur est la partie la plus importante du système. Il créé un thread ''le coordinateur '' qui a pour but de coordonner l'activité des robots.

Le serveur gère le stock de composants. En fonction du nombre de connexions de robots sur l'anneau et du nombre de cases vides, il sélectionne le composant qui sera ensuite placé sur l'anneau.

II.3 – Le coordinateur

Le coordinateur est chargé de :

- définir la position des robots sur l'anneau,
- vérifier le bon fonctionnement des robots : lorsqu'un robot est défectueux il attribue la tâche de ce dernier à un autre.

II.4 – Les robots

Nous avons opté de définir les Robots comme étant de processus indépendants qui seront en communication par le biais de files de messages avec le coordinateur.

Une fois connecté le robot, à chaque pas de rotation de l'anneau, vérifie le contenu de la case qui lui a été affectée au départ. SI la case contient un composant, il vérifie s'il est capable de produire un produit avec ce composant. Ce n'est qu'après avoir vérifié qu'il peut stocker ce composant que le robot prend ce dernier. Si la case contient un produit en cours de fabrication, il vérifie s'il dispose de l'espace pour stocker le produit, et si oui, il s'assure qu'il est capable d'effectuer l'opération attendue sur le produit. Ensuite il récupère ce produit et effectue l'opération attendue.

III –Mode dégradé

Les robots ont la capacité de fonctionner en mode *NORMAL* ou *DÉGRADÉ*. Au démarrage du robot, le signal *SIGUSR2* est armé. La réception de ce signal permet au robot de passer du mode *NORMAL* au mode *DÉGRADÉ* et inversement.

IV –Synchronisation

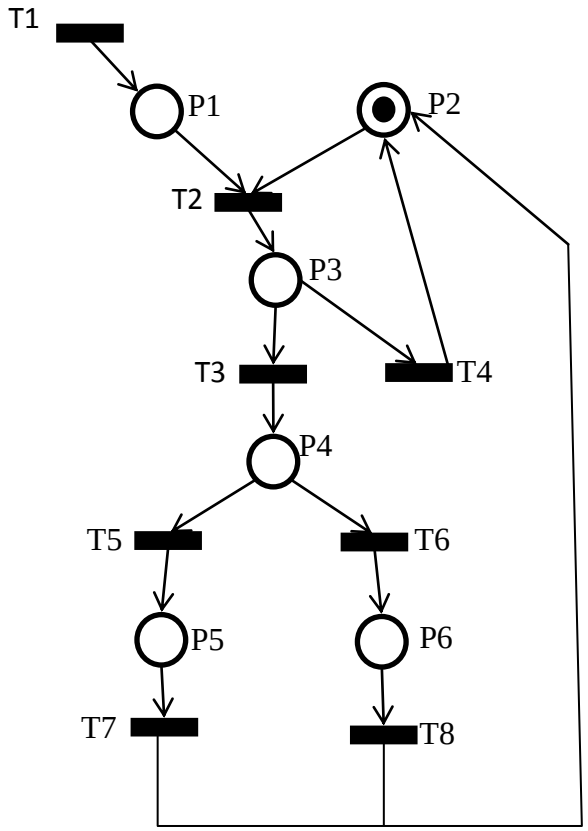
La synchronisation est assurée par des sémaphores. Le processus *anneau* verrouille l'anneau avant de la faire tourner. Ce n'est qu'après que les *agents* peuvent accéder, chacun à son tour, à l'anneau.

V –Inter-blocage

Dans cette partie nous allons présenter les différents cas d'éventuels inter-blocages. Un des cas d'inter-blocage que nous avons dû résoudre dans notre application est celui où les robots récupèrent un composant attendu par un autre robot. En effet, nous avons pu constater qu'en fin de séquence de production, le serveur envoyait des composants et que les robots récupéraient le premier composant qui était présenté dans leur case dédiée. Néanmoins, un produit a parfois besoin de plus d'un seul composant pour être terminé. Or, si un robot qui est placé en premier récupère un composant et que le deuxième robot récupère un autre, ils seront tous les deux en attente du composant que l'autre détient.

Nous proposons donc t'étudier ce cas grâce à un réseau de pétri :

Cas normal, sans traitement du problème d'inter-blocage :



T1 : Flux d'arrivée d'une nouvelle case

T2 : La case est pleine et le robot est libre

T3 : Le composant convient à l'attente du robot

T4 : Le composant ne convient pas au robot

T5 : Le nombre de composant inférieur au nombre nécessaire à la production

T6 : Le nombre de composant est égal au nombre nécessaire à la production

T7 : Le composant est stocké

T 8 : Le produit est créé

P1 : Arrivée d'une case

P2 : Le robot est disponible

P3 : Le composant est vérifié

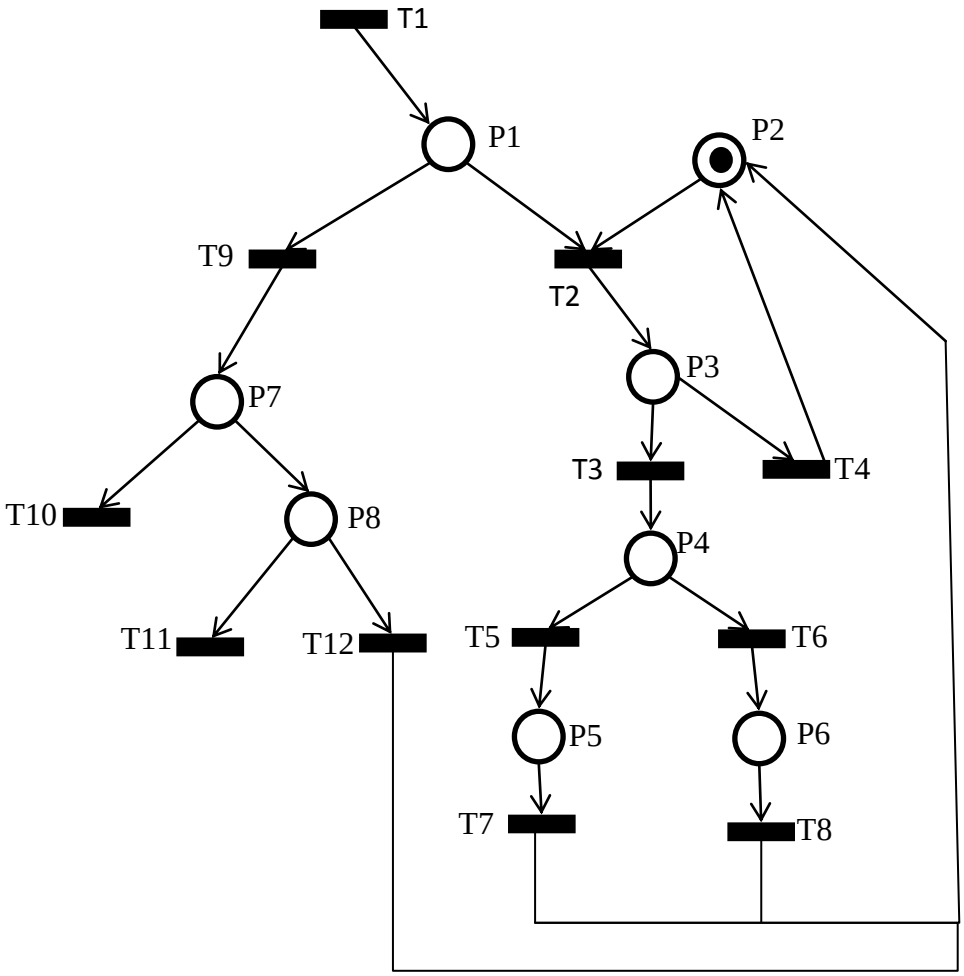
P4 : Le nombre de composants est vérifié

P5 : Stockage du composant

P6 : Création du produit

Dans ce cas nous ne vérifiant pas si l'anneau a effectué un tour complet et qu'il est vide. Le robot peut donc détenir un composant attendu par un autre robot.

Voici le réseau de pétri avec le traitement du problème d'inter-blocage :



T9 : La case est vide

T10 : Un tour de l'anneau est effectué

T11 : Le nombre de composants dans le stock est nul

T12 : Les composants stockés sont libérées

P7 : Vérification du nombre de cases vérifiées (un tour complet ou pas)

P8 : Vérification des composants dans le stock

Le nombre de cases vérifiées est incrémenté par le processus P7. Nous allons donc vérifier que le stock de composant n'est pas inférieur au nombre de composant nécessaire. Dans le cas où le robot détient toujours des composants nous allons passer à la libération des composants stock inutilement.

Bien évidemment, notre solution a été implémentée en mettant en place des sémaphores lors de la vérification de la case qui se présente au robot, et lors de la libération des composants. Ce procédé permet de bloquer les autres processus pour que les valeurs soient protégées lors de la modification.

VI – Compilation et exécution

Le projet utilise la puissance du *Makefile* pour simplifier le processus de compilation. Il suffit d'exécuter le fichier *install.sh* et de lancer dans chaque terminal les différents exécutables générés.

- **COMPILATION**
  - **Méthode 1**

    Exécuter le fichier install.sh

    $ ./install.sh

  - **Méthode 2**

    1. Ouvrir le terminal et se positionner dans le répertoire du projet
    2. Exécuter la commande:

    $ mkdir run
    $ mkdir build
    $ make

- **Exécution**

  1. Démarrage de l'anneau

     Exécuter dans un terminal la commande:

     $ ./start_anneau.sh *# Démarre l'anneau avec une fréquence de 500 ms*

  2. Démarrage du serveur

     Exécuter dans un terminal la commande:

     $ ./start_server.sh

  3. Démarrage des robots

     À partir du répertoire principal, exécuter dans chaque nouveau terminal les commandes:

     $ ./start_robot_1.sh
     $ ./start_robot_2.sh
     $ ./start_robot_3.sh
     $ ./start_robot_4.sh
     $ ./start_robot_5.sh
     $ ./start_robot_6.sh

Conclusion

Notre projet a été mené à terme pour les parties que nous avons initiées. En effet, dans notre simulation nous avons pu développer le fonctionnement normal et dégradé des robots.

Durant ce projet nous avons eu l'opportunité de revoir les outils qui nous ont été présentés dans l'UV. Cela est dû au fait que ce projet a pour objectif de mettre l'étudiant fasse à un cas très concret de programmation système.

Toute la démarche utilisée fait appel à des connaissances qui ont été acquises grâces à l'UV. L'organisation du travail a été aussi un challenge pour pallier au problème de temps, mais avons pu mettre en place un travail de groupe assez conséquent.