Rapport Initiale PSTL : Implantation répartie des Réseaux de Petri en BCM4Java

Par Mohamed Mougamadoubougary, Denn Marsso, Gabriel Zamy.
Encadré par Jacques Malenfant

1 - Description du projet :

Le projet porte sur le sur la conception de réseaux de Petri en BCM4Java et l'intégration au sein de composants de BCM4Java. Les réseaux de Petri sont des modèles mathématiques servant à représenter divers systèmes travaillant sur des variables discrètes. Ils sont représentable par un graphe bipartite dont les entités sont les places et les transitions. L'information, données discrètes, se lit par la présence ou non de jetons dans les différentes place du réseau, ces jetons peuvent changer de place en passant par les transitions existantes. Dans le cadre de ce projet nous allons utiliser ces réseaux comme moyen de coordination des processus, permettant leur synchronisation et le déclenchement d'actions.

Ce projet vise à exploiter ces capacités en les intégrant dans une architecture de composants répartis. Cette architecture sera celle de BCM4Java, qui est une implantation de composants logiciel distribués développée par le LIP6. Son fonctionnement et ses principes sont majoritairement donnés par le cours de Composant du M1 STL.

Comme ce projet est à faire entièrement en Java nous avons fait le choix d'utiliser l'IDE Eclipse permettant d'intégrer les différentes bibliothèque plus aisément.

2 – Cahier des charges :

Pour arriver à l'objectif final de ce projet il y a plusieurs points étapes à effectuer avec pour chacune d'entres elles certaines contraintes données dès le début du projet.

En premier lieu il s'agissait de se renseigner et de comprendre les implémentations usuelles des réseaux de Petri grâce à la lecture d'articles de recherche déjà existants. Cela demande un effort de recherche bibliographique et de synthétisation qui a donné lieu à un premier rapport.

Cela mène ensuite à une première implémentation, tout d'abord dans un code minimaliste pour s'assurer de la compréhension, puis une version complète en BCM4Java est attendue. L'utilisation de BCM4Java permet d'assurer l'exécution répartie.Cette deuxième tâche à réaliser marque la fin de la première partie du projet et ouvre la partie sur les composants.

La seconde partie du projet doit démarrer par la conception d'interfaces permettant de lier l'implantation des réseaux de Petri avec le code des composants pour que l'exécution des réseaux de Petri puisse coopérer avec l'exécution du code des composants. Concrétement cela signifie que les méthodes d'un composant puissent lancés des transitions dans le réseau de Petri, et inversement qu'effectuer une transition du réseau puisse déclencher des méthodes des composants.

La dernière tâche à réaliser pour mener à bien ce projet est de créer un exemple permettant de valider notre implémentation. Nous devons donc implanter un exemple de calcul réparti sur

composants BCM4Java dont les tâches seront coordonnées par un réseau de Petri réparti et fournir un élément de validation visuel.

3 – Tâches déjà réalisées :

La majorité du travail de renseignement s'est faite sur l'article « Distributed implementation of discrete event control systems based on Petri Nets » qui a été le point de départ du projet. Il a fallu retrancher les informations avec des définitions et explications trouvées sur d'autres articles pour bien comprendre le fonctionnement car une partie de cet article incluait la communication en temps réel qui dans le cadre de notre projet sera prise charge par BCM4Java. La conclusion des recherches a été que nous allions avoir besoin de quatre classes objet java pour l'implémentation de base: Place, PlaceCommune, Transition et Reseau. Le tout est ensuite instancié et géré dans des classes à part créées dans la deuxième tâche.

Nous avons ensuite créé le projet java sur Eclipse pour proposer notre première implémentation des réseaux de Pétri. Le projet est structuré autour des classes Place, Transition, PlaceCommune et Reseau qui représentent un réseau de Petri. De plus, les classes Frontend et Backend sont responsables de la mise en œuvre du réseau et des interactions possibles. La classe Place comporte comme attribut un identifiant (uri), deux listes de transitions , une pour les transitions qui arrivent sur la place et une pour celles qui en sortent. Dernièrement cette classe conserve une quantité de jetons avec des fonctions pour les ajuster. PlaceCommune hérite de Place et permet le partage des places et donc possiblement des jetons sur différents réseaux. Transition s'occupe de la gestion des liaisons entre les places et de leur activation en fonction d'une fonction spécifique. Le réseau consiste en des emplacements et des transitions, et dispose de méthodes pour les intégrer et les administrer. L'élément Backend coordonne ces tâches : il permet de mettre en place des réseaux (CreateNetwork), d'ajouter des places (CreatePlace, CreatePlaceCommune), de les connecter par le biais de transitions (LinkPlaces) et de définir leur état initial (InitializePlace, InitializePlaceCommune). Enfin, Frontend est appelé depuis le main du programme et relaie les appels vers la classe Backend, rendant ainsi l'ensemble fonctionnel et exploitable. Notre implantation a été validée par un premier exemple donné par Jacques Malenfant.

Nous sommes actuellement en train de développer le code pour permettre la liaison entre notre implantation et le code des composants. Cela a démarré avec la création des interfaces pour les classes Reseau et PlaceCommune. Les classes nouvelles classes java se trouvent dans les packages placeCommune et reseau. Dans la plupart des fichiers on fait appel à certaines partie de la bibliothèque BCM4Java : «import `fr.sorbonne_u.components`». Pour l'instant nous avons créé les classes qui vont devoir représenter et gérer notre réseau de Petri en utilisant la bibliothèque que l'on vient d'intégrer. Nous sommes entrain de développer les fonctions pour les plugins dans les classes reseauPlugin et placeCommunePlugin qui toutes les deux doivent donc hériter de AbstractPlugin. Ces deux classes gardent les mêmes attributs que leur équivalent dans la première implantation java mais on y trouve également des attributs permettant la communication avec les composants de BCM, par exemple la configuration des input et ouput port.

4 – Prévisionel des tâches restantes :

Pour compléter ce projet il nous reste deux principaux points à valider. En premier lieu l'utilisation des plugins et leur implantation pour la communication avec les composants doit être complétée. Ensuite il faudra implémenter un exemple de calcul réparti sur composants BCM4Java dont les tâches seront coordonnées par un réseau de Petri réparti afin de valider l'ensemble du projet. L'objectif est d'avoir fini l'implantation avant le mardi 7 avril pour obtenir l'exemple lors de la prochaine réunion. Cela laissera le temps pour vérifier si tout fonctionne et dans le cas contraire retravailler le code pour arriver un résultat intéressant.