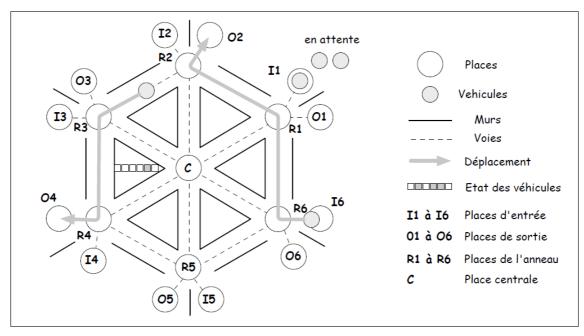
Module LO43 – sujet de projet

Flotte de véhicules autonomes

© UTBM Octobre 2014

Le projet consiste à simuler l'**exploitation concurrente** d'une infrastructure de circulation partagée par une flotte de véhicules autonomes. Le schéma ci-dessous décrit l'infrastructure (et/ou son synoptique).



Des passagers souhaitent se déplacer d'une place d'entrée vers une place de sortie. Pour cela ils émettent des requêtes de déplacement auprès d'un contrôleur. Lors d'une requête, le contrôleur recrute un véhicule libre et lui demande de se rendre vers la place d'entrée, puis lui attribue une destination (place de sortie). Le véhicule se charge alors d'établir une trajectoire sûre en réservant les ressources (places traversées). Lorsqu'il dispose de toutes les ressources nécessaires, le véhicule embarque le passager et se rend à la destination souhaitée. Lorsqu'il arrive en fin de mission (place de sortie) il prévient le contrôleur qui a pour rôle de libérer le véhicule pour une mission future. La circulation le long des voies est possible dans les deux sens (sauf pour les entrées et les sorties) mais un seul véhicule peut occuper une place ou une voie de circulation à un instant donné. Pour qu'un véhicule puisse se déplacer vers une place il doit avoir acquis le privilège d'utilisation de cette place. Pour aller d'une place d'entrée Ii à une place de sortie Oj plusieurs routes sont possibles. Les sources fournis se limitent au cas simple consistant à toujours choisir la route la plus courte ne passant par le centre que pour les missions diagonales. Par exemple, la route allant de I6 à O2 est I6-R6-R1-R2-O2.

Le partage de l'infrastructure pose un problème d'accès concurrent aux places. Plusieurs algorithmes sont envisageables. La solution retenue consiste, pour chaque véhicule, à acquérir à l'avance la totalité des ressources de la route qu'il souhaite emprunter et à libérer ces ressources au fur et à mesure de son parcours. L'allocation des ressources doit se faire selon une règle précise pour éviter les inter-blocages; ce qui serait le cas si 2 véhicules acquièrent chacun une partie des ressources et essaient d'acquérir celles que l'autre a déjà acquises.

Description des principaux objets

1) Véhicule

Les véhicules se déplacent de manière autonome et sont modélisés par des objets **tâches**. L'entrée **Enroll** sert à appeler un véhicule pour qu'il rejoigne une place d'entrée. Lorsque le véhicule a rejoint la place d'entrée le contrôleur lui attribue une **Mission** vers une place de sortie. Une fois la mission acceptée le véhicule définit la trajectoire qu'il va suivre et effectue son déplacement. Quand il arrive sur la place de sortie et après avoir déchargé ses passagers, il accepte **Release** qui le libère de ses obligations et lui permet de quitter la place

de sortie. Les entrées Enroll, Mission et Release seront appelées par le **contrôleur**. La séparation d'une demande de mission en 3 entrées sert essentiellement à permettre la visualisation des véhicules en attente près d'une place d'entrée (Enroll) et libérer correctement la place de sortie (Release). Cette séparation se clarifiera lors de l'avancement du projet.

2) Contrôleur

Le contrôleur est un **objet actif** (tâche). Il exploite une référence (sous la forme d'un **discriminant accès**) vers une boîte aux lettres des arrivages qu'il partage avec les véhicules. Son rôle est d'accepter les demandes de transfert des passagers, réserver les véhicules, leur attribuer des missions, rester à l'écoute des notifications d'arrivée issus des véhicules et libérer les véhicules en fin de mission (**Release**).

Les véhicules sont regroupés (enregistrés) dans une **flotte de véhicules**. Le **contrôleur** peut réserver un **véhicule libre** de la flotte et lui attribuer une mission. Il est également à l'écoute des fins de missions des véhicules via une boîte aux lettres qu'il partage avec les différents véhicules. Le contrôleur est en principe la seule tâche à accéder à la flotte de véhicules et en toute logique cette dernière peut être réalisée à l'aide d'un **objet passif non protégé**. Le contrôleur et les véhicules communiquent via une boîte aux lettres.

Exclusion mutuelle pour l'accès à l'infrastructure

Le problème principal de ce projet à savoir l'exclusion mutuelle des véhicules lorsqu'ils exploitent de manière simultanée l'infrastructure.

Un véhicule auquel a été attribué une mission doit acquérir l'ensemble des ressources associées à sa mission (y compris les places d'entrée et de sortie) avant de pouvoir se déplacer. Il libère les ressources au fur et à mesure qu'il quitte les places qu'il a réservées. Il est donc clair qu'un véhicule doit **acquérir plusieurs ressources** simultanément, ce qui peut poser un problème si l'allocation n'est pas faite correctement. Par exemple si un véhicule 1 doit aller de I1 à O2 il doit acquérir {I1, R1, R2, O2} et si un véhicule 2 doit aller de I2 à O1 il doit acquérir {I2, R2, R1, O1}. Si l'allocation se fait dans l'ordre des «listes» précédentes il peut y avoir un blocage si le véhicule 1 a acquis {I1, R1} et le véhicule 2 a acquis {I2, R2}, car le véhicule 1 cherchera à acquérir R2 que le véhicule 2 a déjà réservé, il en va de même pour le véhicule 2 qui cherchera à acquérir R1 qui est déjà réservé par le véhicule 1, ce qui conduit à une **étreinte fatale**. Une solution simple pour éviter ce genre d'inter-blocage consiste à définir une relation d'ordre sur les ressources et à toujours allouer les ressources en respectant cette relation d'ordre. Par exemple si on a la relation:

Le véhicule 1 réservera les ressources dans l'ordre I1, R1, R2, O2 et le véhicule 2 réservera les ressources dans l'ordre I2, R1, R2, O1, il ne peut donc plus y avoir d'étreinte fatale entre les 2 véhicules. Afin de mettre en oeuvre cette stratégie d'allocation on utilisera des **requêtes sous forme de bitmap** (des vecteurs de booléens). Par exemple pour les 2 requêtes précédentes on aurait les deux **Request Map** suivantes:



Travail demandé

- Conception orientée objet en UML.
- Coder les différentes classes en Java en se basant sur le paradigme (MVC).
- Créer une Interface graphique pour illustrer le fonctionnement du système.
- Rédiger un rapport décrivant l'architecture générale du système, la conception et solution proposée.