Groupe 18

Rapport Projet en Groupe - INFOB236 Projet de Programmation

Dans le cadre de ce projet, le thème abordé est la modélisation d'un réseau de transport de chemin de fer en montagne. Ce système se compose de différents éléments tels que de différentes gares, différentes lignes de convois, différents systèmes d'urgences ainsi que de diverses fonctionnalités. Ceux-ci seront détaillés ci-dessous.

Tout d'abord, la modélisation du système repose sur plusieurs machines (modèle initiale, et modèles raffinés) et un contexte. La machine initiale devient de plus en plus raffinée au fil des machines proposant ainsi un modèle concret. Le modèle est accompagné d'une animation en B-Motion apportant une représentation visuelle du système. Les machines sont développées de sorte, à ce qu'elles répondent à l'ensemble des fonctionnalités du système. Pour cela, les machines reposent sur un contexte pensé au préalable. Ce contexte comporte différents ensembles nécessaires au développement des machines. Ces ensembles ne peuvent être modifiés durant toute l'exécution du modèle.

Ensuite, le modèle proposé répond à un cahier de spécification ainsi qu'à une stratégie de raffinement. Dans ce cahier, les équipements sont décrits et détaillés. Les fonctionnalités et les équipements de celui-ci proviennent uniquement d'une analyse de l'énoncé fourni par l'auteur. Par conséquent, le cahier n'inclut que ce qui est mentionné dans les indications de l'auteur et ne comporte aucunement des fonctionnalités ou équipements qui ne sont pas explicités dans l'énoncé. En ce qui concerne la stratégie de raffinement, elle inclut les fonctionnalités et équipements définis dans le cahier de spécification pour que le modèle proposé se développe de manière compréhensible et concrète. Cette stratégie passe donc d'un modèle abstrait à un modèle plus concret. (Fourni en pièces jointes)

Puis, les machines reposent sur différentes variables. Elles permettent d'apporter de la profondeur au modèle. La nature de ces variables est décrite lors de la définition des invariants. Il est important de préciser ces invariants. Sinon, la machine ne pourra interpréter d'elle-même la signification de ces variables. Ces variables sont spécifiées par des commentaires dans l'intégralité des machines.

Enfin, les fonctionnalités décrites sont ajoutées en tant qu'événements. Des conditions de « garde » interviennent si l'une des fonctionnalités ne répond pas à certains critères permettant de préserver l'intégrité du modèle. Voici un exemple d'une condition de « garde » : il est possible de faire sortir un train d'une gare. Toutefois si l'utilisateur tente de faire sortir un train d'une gare alors que celle-ci ne dispose pas de trains, cette condition de « garde » empêchera de violer l'intégrité du système dans lequel un train peut sortir d'une gare uniquement s'il est présent dans celle-ci. Ainsi, les différentes fonctionnalités seront présentées ci-dessous de manière que les explications soient courtes et claires.

Avant toute chose, certains problèmes pratiques furent rencontrés lors de l'implémentation du modèle. Malheureusement, la raison de ces problèmes est encore inexplicable, car le logiciel sur lequel le modèle fut travaillé « Rodin » ne détecte aucune erreur. Ces problèmes sont expliqués au moyen d'une vidéo d'une minute (Lien : https://drive.google.com/file/d/14BORpE--GgRBSiasncj_5ljxDqFM1ThB/view?usp=share_link) Ces problèmes visuels se dénombrent au nombre de deux :

- Lorsque le modèle est initialisé, rien n'apparaît à l'écran. Les événements ne sont pas exécutables. Mais, il s'agit uniquement d'un problème visuel. Il suffit de passer le curseur de la souris sur l'événement pour que le problème visuel se corrige.

- Les événements proposent à l'utilisateur de choisir des paramètres. Ces paramètres respectent un certain ordre. Mais « Rodin » affiche ces paramètres dans un ordre aléatoire et il est impossible de changer cela. Voici un exemple avec trois paramètres : choisir la gare de départ, choisir la gare d'arrivée et choisir le nom du train composé avec les gares de départ et d'arrivées. Cela ressemble donc à cela (Namur, Bruxelles, Namur -> Bruxelles), mais sur « Rodin », il s'affiche aléatoirement. Voici un cas possible d'affichage : (Namur, Namur -> Bruxelles, Bruxelles). Il est donc important de faire un clic droit pour afficher la boite de dialogue et de vérifier les paramètres demandés.

Le cahier de spécification ainsi que la stratégie de raffinement furent modifiés après la séance avec le professeur/assistante. Nos modèles ne répondent donc plus aux dépôts déposés sur Webcampus. Le nouveau cahier de spécification et la nouvelle stratégie de raffinement sont disponibles en pièces jointes.

Voici une courte explication des différentes machines. À la fin des explications, des exemples de suite d'événements sont fournis pour vérifier le modèle.

Machine_Initiale :

Cette machine répond au modèle initial de la stratégie de raffinement. Pour rappel, ce modèle doit être sûr avant tout. PS N'oubliez pas les problèmes visuels décrits plus haut !

- Apparition_train fait apparaître un train dans le système dans la gare choisie (non d'urgence). Le nombre de trains dans le système est limité à 4 trains.

- Station_To_Line introduit un train sur la première ligne de convoi. Pour cela, il est nécessaire de choisir une gare de départ et une gare d'arrivée. Le dernier paramètre est le nom du train qui prend le nom de la ligne du trajet. (Exemple : Nom de train = Bruxelles -> Namur)

- Line_To_Second_Line. Pour rappel, il est possible que deux trains soient sur la même ligne de convois, mais qui arrive de sens opposé. Pour différencier les lignes de convois de sens contraire, deux lignes de convois furent implémentées. Une ligne de convois ainsi, qu'une seconde ligne de convois. Ces lignes prennent le nom de la ligne de leur trajet entre deux gares. Voici un exemple concret : Line = (Namur -> Bruxelles). Second_Line = (Bruxelles -> Namur). Pour des raisons de gardes d'événements, les paramètres désignent la gare de départ du train, et la gare d'arrivée. Soit les gares formant la ligne de trajet et donc le nom du train.

- Line_To_Emergency_Station introduit un train présent sur une ligne de convois vers une gare d'urgence. Dans le cas, où deux trains sont sur de mêmes lignes, mais provenant de sens opposés. Cet événement est possible uniquement si le changement de l'aiguillage est possible. Pour ce faire, il faut choisir le nom du train (défini par le trajet de sa ligne de convoi) et une gare d'urgence.

- Emergency_To_Second_Line introduit un train présent dans une gare d'urgence vers une ligne de convoi. Pour ce faire, il faut choisir la gare d'urgence où le train se trouve et le nom du train (défini par le trajet de sa ligne de convoi).

- Second_Line_To_Station introduit un train présent sur une ligne de convoi vers une gare (non d'urgence). Pour des raisons de gardes d'événements, les paramètres désignent la gare de départ du train, et la gare d'arrivée. Soit les gares formant la ligne de trajet et donc le nom du train.

- Choose_Aiguillage permet de choisir la position de l'aiguillage des lignes de convois. (TRUE pour aller vers une gare d'urgence, FALSE vers une gare normale)

- Freinage_Urgence intervient lorsque deux trains sont présents sur la même ligne de convoi dans des sens opposés. Cet événement est possible uniquement si le changement de l'aiguillage est impossible. De plus, l'événement introduit l'événement suivant qui est la marche arrière en modifiant l'état de certaines variables. Pour cela, il faut choisir les trains qui portent le nom de leurs lignes de convois ainsi qu'une gare disponible pour une marche arrière. Les gares disponibles n'incluent pas les gares d'urgences.

- Marche_Arriere_To_Station intervient lorsque deux trains sont présents sur la même ligne de convoi dans des sens opposés. Il est nécessaire que l'événement « Freinage d'urgence » fût enclenché pour lancer cet événement. De nouveau, il faut choisir les trains qui portent le nom de leurs lignes de convois ainsi qu'une gare disponible pour une marche arrière. Les gares disponibles n'incluent pas les gares d'urgences.

First_Refine :

Pour rappel, le premier raffinement introduit les gares principales et secondaires. Ces gares comportent des quais. Deux quais pour les gares principales pour le transport et une autre réservée exclusivement à une marche arrière. Et un seul quai pour les gares secondaires.

- Apparition_train fait apparaître un train dans le système dans une gare principale sur le quai numéro 1

- Line_To_Secondary_Station introduit un train dans une gare secondaire et la gare n'est plus considérée comme gare "d'urgence"

- Secondary_Station_To_Second_Line introduit un train disponible dans une gare secondaire et non d'urgence

- Second_Line_To_MainStation désigne le fait qu'un train arrive dans une gare principale.

- Marche_Arriere_To_Station désigne une marche arrière d'un train vers une gare principale sur le quai numéro 2.

Pour les événements restants, il n'y a aucun changement notoire

Second_Refine :

Pour rappel, le deuxième raffinement introduit les feux de signalisation.

- MainStation_To_Line, si un train est déjà présent sur la ligne, il sera impossible de faire sortir le train de la gare.

- Line_To_Secondary_Station, si un train rentre dans une gare secondaire, il sera impossible d'en faire rentrer un second

- Second_Line_To_MainStation, si un train rentre dans une gare secondaire, il sera impossible d'en faire rentrer un second

Aucun changement notoire pour les autres événements, car un chauffeur peut ignorer les feux de signalisation. Cependant, le système reste sûr grâce au modèle initial malgré le fait qu'un chauffeur puisse brûler un feu.

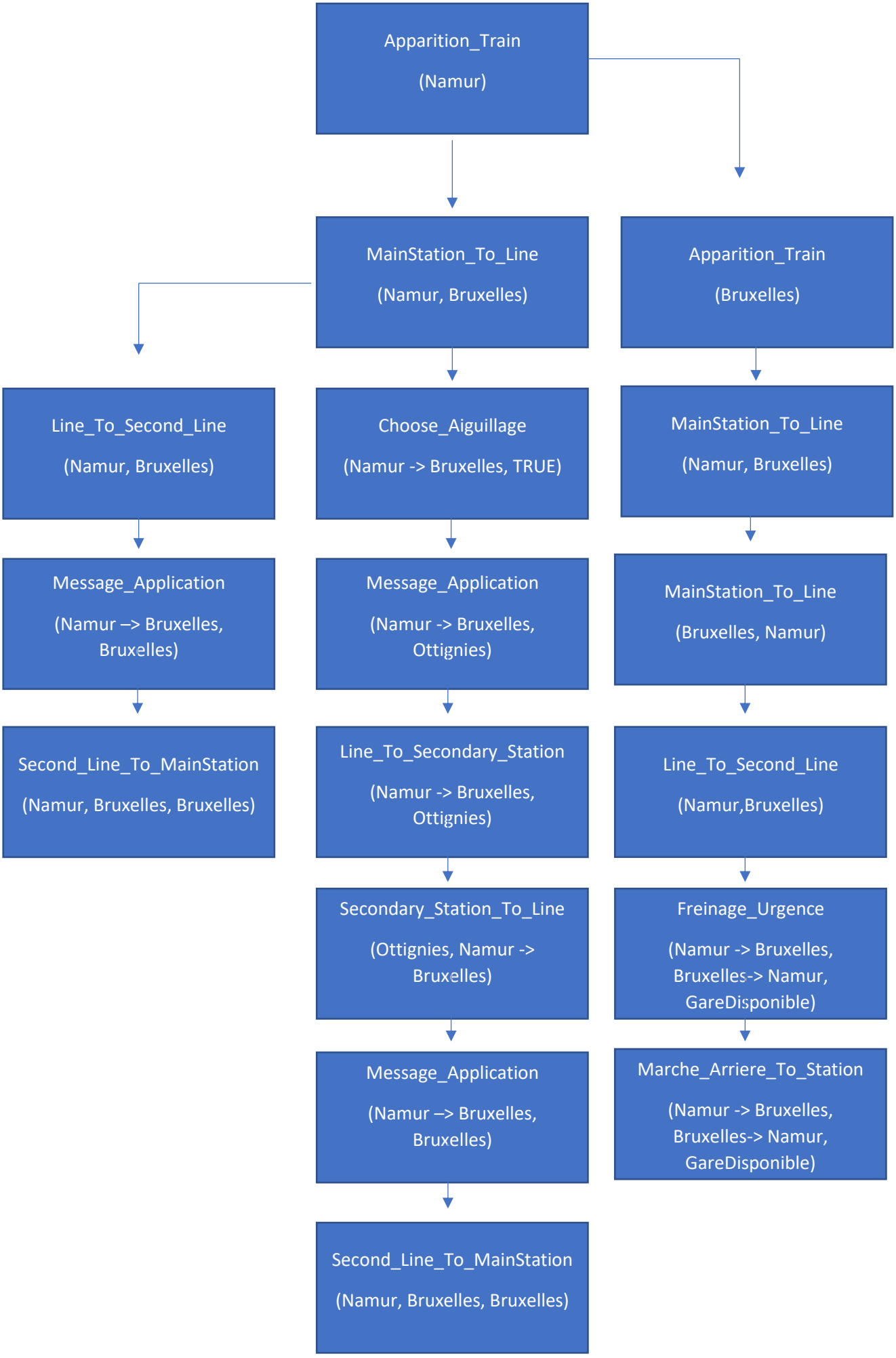
Third_Refine :

Pour rappel, le troisième raffinement introduit l'application pour le conducteur et pour les voyageurs.

- Message_Application est un événement dans lequel les voyageurs sont avertis d'un train en approche d'une gare (Principale ou Secondaire). Pour ce faire, certains gardes sont implémentés pour que cet événement devienne obligatoire avant l'entrée de chaque train dans une gare.

- Cependant, il fut impossible d'implémenter une application pour les conducteurs comme l'exige l'énoncé. Ainsi, le modèle suppose que le conducteur connaît l'état du système lorsqu'il choisit les différents paramètres avant d'exécuter un événement. Bien sûr, les paramètres proposés à l'utilisateur sont d'abord vérifiés par le Dispatching central au moyen de gardes d'événements.

Enfin, vous verrez différentes situations à la page suivante pour comprendre le système de manière plus simple et ainsi avoir une idée des différents événements. Voici une suite d'événements possible.


Apparition_Train
(Namur)
MainStation_To_Line
(Namur, Bruxelles)
Apparition_Train
(Bruxelles)
Line_To_Second_Line
(Namur, Bruxelles)
Choose_Aiguillage
(Namur -> Bruxelles, TRUE)
MainStation_To_Line
(Namur, Bruxelles)
Message_Application
(Namur –> Bruxelles, Bruxelles)
Message_Application
(Namur -> Bruxelles, Ottignies)
MainStation_To_Line
(Bruxelles, Namur)
Second_Line_To_MainStation
(Namur, Bruxelles, Bruxelles)
Line_To_Secondary_Station
(Namur -> Bruxelles, Ottignies)
Line_To_Second_Line
(Namur,Bruxelles)
Secondary_Station_To_Line
(Ottignies, Namur -> Bruxelles)
Freinage_Urgence
(Namur -> Bruxelles, Bruxelles-> Namur, GareDisponible)
Message_Application
(Namur –> Bruxelles, Bruxelles)
Marche_Arriere_To_Station
(Namur -> Bruxelles, Bruxelles-> Namur, GareDisponible)
Second_Line_To_MainStation
(Namur, Bruxelles, Bruxelles)