# SimulatHEURE

## Modèle de conception et Architecture logique

présenté à

**Johnathan Gaudreau**

par

Équipe 0 — Fastrol

| *matricule* | *nom* | *signature* |
|---|---|---|
| 111 076 721 | Mercier, Rémi | |
| 111 066 466 | Magnan, Charles-Olivier | |
| 111 072 232 | Cloutier, Samuel | |
| 111 071 384 | Provencher, Jean-Michel | |

Université Laval

10 Avril 2015

| Historique des versions | | |
|---|---|---|
| *version* | *date* | *description* |
| 3.0 | 16 novembre 2015 | - Diagrammes d'états et d'activité |
| 2.0 | 24 octobre 2015 | - Modèle de conception et Architecture logique |
| 1.0 | 22 septembre 2015 | Itération 1 de la phase d'élaboration |

# Table des matières

# Chapitre 1

# Modèle de conception

## 1.1 Diagramme de classe de conception

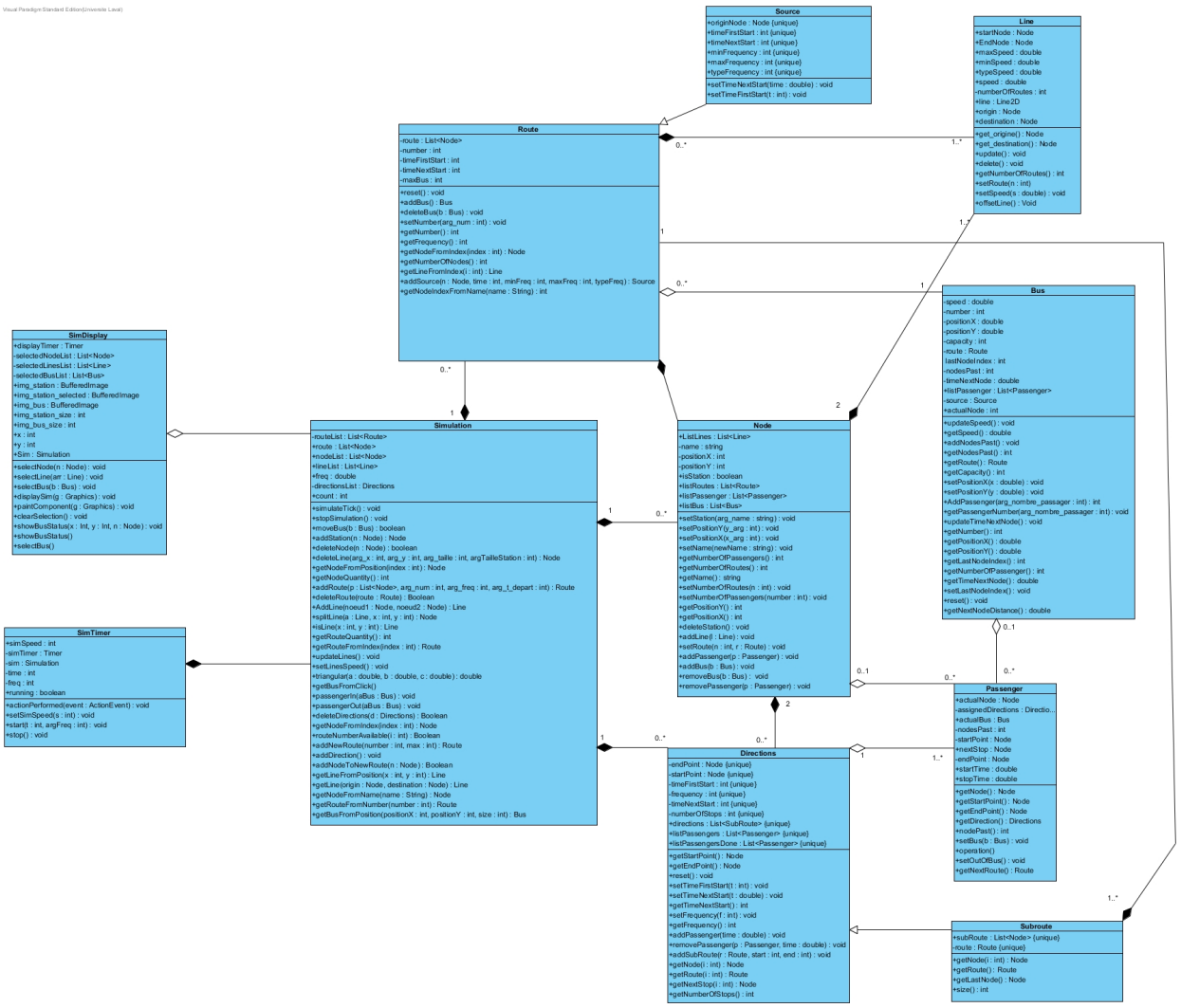


FIGURE 1.1 – Diagramme de classe de conception de SimulatHEURE

Dans la figure 1.1, on retrouve le diagramme de classes de conception. L'objet Simulation est le coeur de celui-ci et agit comme contrôleur logique. La classe SimDisplay permet quant à elle d'interagir avec l'utilisateur et de représenter de façon graphique la simulation. Lorsqu'une simulation est créé, il est possible de créer des noeuds (Node), des arêtes (Line) des besoins en transport (Directions) et des circuits (Route). Un passager quant à lui, possédera une position sur un noeud ou dans un Bus ainsi qu'un besoin en transport. Un circuit est composé de noeuds (Node) et possède des attributs représentant la liste des noeuds, la liste des bus sur le circuit ainsi que la fréquence par exemple. Plusieurs méthodes sont implémentés dans Route afin d'ajouter ou de retirer des bus et de retourner des informations comme le temps avant le prochain départ. Chaque route possède un ensemble de source (Source), correspondant à un point de départ pour un autobus. Chaque source a une fréquence donnée qui correspond au temps entre les départ des autobus. Dans la classe Line, les principaux attributs sont les noeuds de départ et d'arrivé ainsi que différents attributs permettant de quantifier la vitesse sur le segment. Les méthodes implémentées dans cette classe permettent notamment de mettre à jour une arète lorsque les noeuds sont déplacés et de la supprimer. L'objet Noeud, pouvant être créé à partir de Simulation, contient une liste d'arêtes associées, un nom (si l'attribut isStation est vrai), une position spatiale ainsi que le nombre de circuits et de passagers qui y sont rattachés. Les méthodes rattachées à Noeud permettent notamment d'y rattacher d'autres arêtes ainsi que de déplacer le noeud dans l'espace. La classe Bus, elle, possède des attributs de position dans l'espace ainsi que différentes informations comme la vitesse actuelle, le dernier noeud parcouru, le nombre de passagers, le circuit actuel et sa capacité. Les méthodes de cette classe permettent de mettre à jour la vitesse, le temps et la distance avant le prochain noeud. On peut également modifier le nombre de passagers et retourner l'index du dernier noeud parcouru dans son circuit. La classe SimTimer, dicte le rythme de la simulation et permet de contrôler la vitesse, le départ et l'arrêt de la simulation.

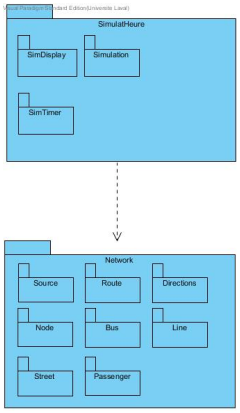


FIGURE 1.2 – Diagramme de package de SimulatHEURE

Dans le diagramme de package, on peut voir que le niveau de UI (SimulatHeure) est une couche au-dessus du package Network qui contient les différentes classes relié au réseau d'autobus. Le contrôleur logique (Simulation) est contenu dans le dossier Network et c'est

lui qui permet de faire le pont entre l'interface utilisateur est les différentes classes contenues dans Network.

## 1.2 Interface-utilisateur

Cette section présente un aperçu de l'interface-utilisateur offrant différentes fonctionnalités. Bien que l'aspect visuel de l'interface soit ici construit en Java directement, plusieurs des boutons et fonctions ne sont pas encore implémentées. Il s'agit donc d'une version préliminaire où rien n'est définitif.

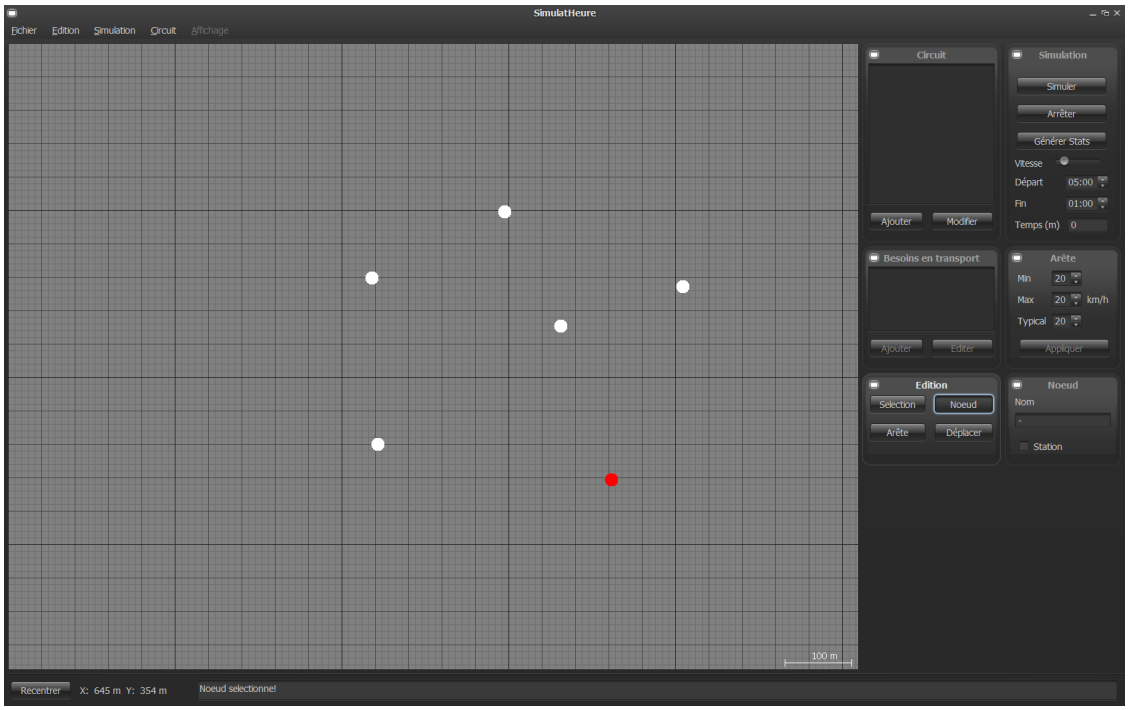


FIGURE 1.3 – Commande d'ajout d'un Noeud

La méthode par défaut pour ajouter un objet, en supprimer ou effectuer toute forme d'édition du schéma affiché et via le menu *Edition*. De cette façon, bien qu'une commande puisse être aussi effectuée à partir d'une des boites d'outils (*Circuit*, *Simulation* et *Edition* pour l'instant), il est toujours possible de retrouver la commande voulu dans le menu horizontal approprié. Le menu *Edition* permet entre autres de créer des objets *Noeuds*, de transformer ceux-ci en *Station* ainsi que de les relier entre eux via la création d'*Arrêtes*. La figure suivante montre un résultat possible, où les carés vides sont des *Noeuds*, les icônes d'arrêt de bus des *Stations* et les segments les rejoignant des *Arrêtes*.

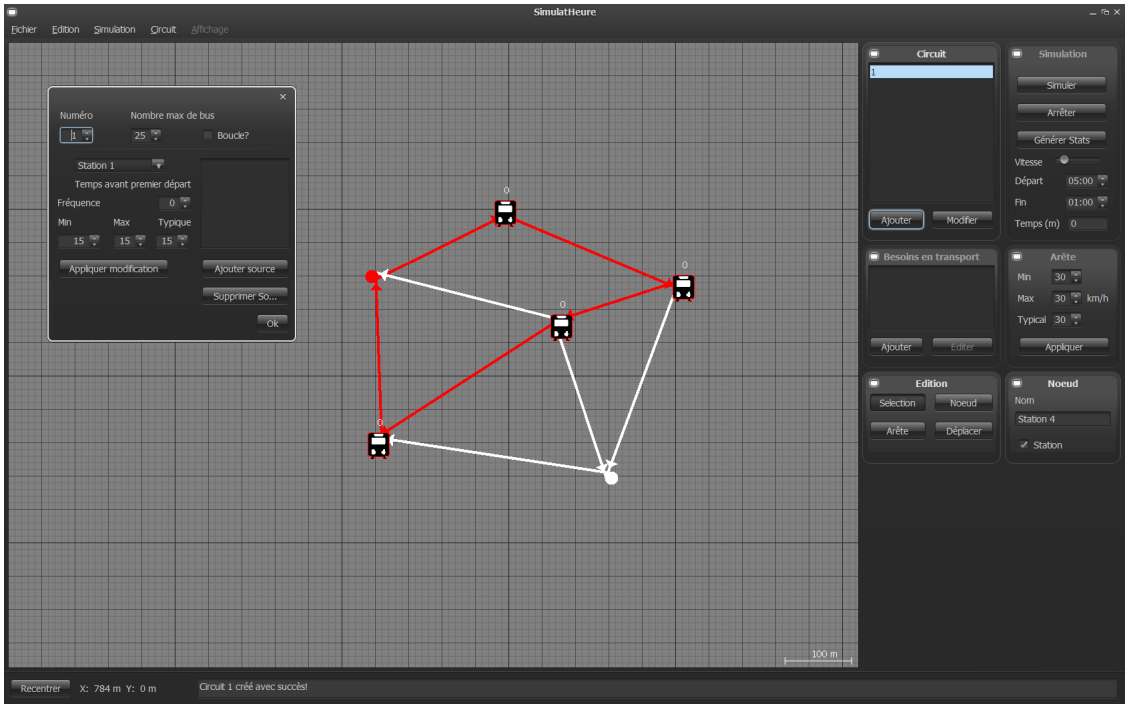


FIGURE 1.4 – Création d'un nouveau circuit

Une fois chaque éléments du réseau mis en place, l'utilisateur peut organiser le trafic des autobus en créant des *Circuits* à partir du menu Circuit ou de la boite à outils appropriée. Une boite de dialogue apparaît alors permettant de régler certains paramètres propres à un *Circuit*, tel que son numéro d'identification, la fréquence des départs ainsi qu'un délai avant le tout premier départ. Lorsque l'utilisateur clique sur "Ok", il sélectionne une série de *Stations* et de *Noeuds*, dans l'ordre voulu de déplacement des bus. Le *Circuit* est finalement créé et ajouté à la liste des *Circuits* à simuler.

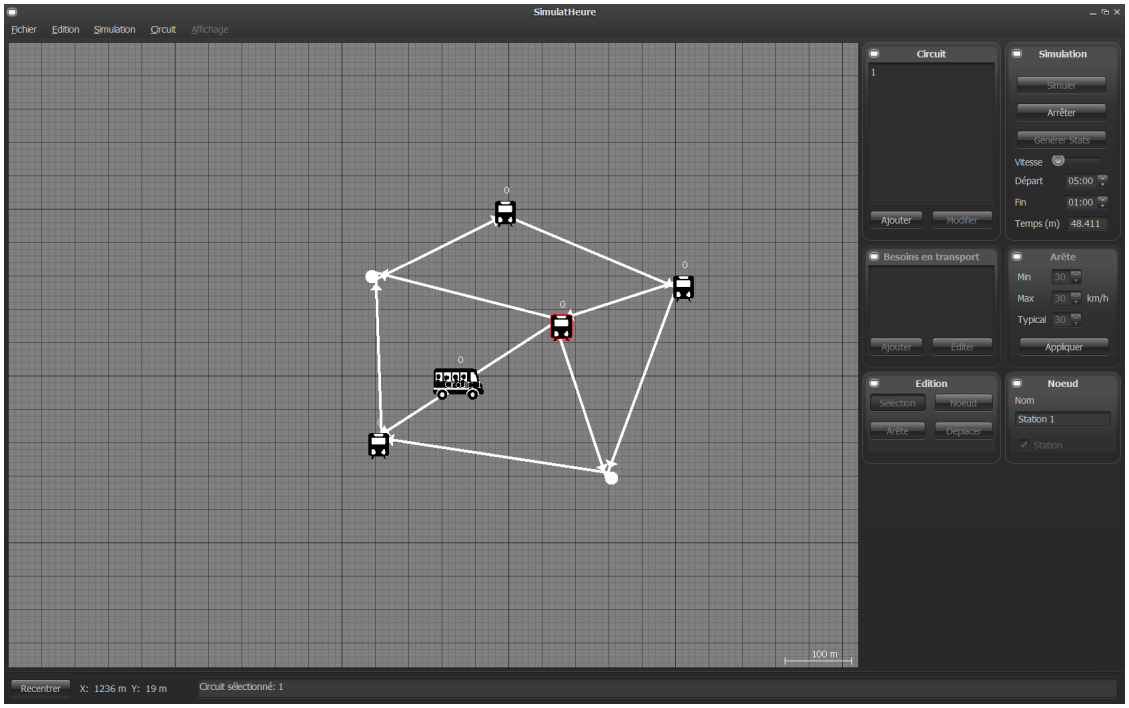


FIGURE 1.5 – Simulation du circuit ajouté à dans la figure 1.4

La boite d'outils Simulation permet, lorsqu'au moins un *Circuit* existe, de contrôler l'état, la vitesse et la longueur de la simulation des bus. Lors de celle-ci, les bus sont créés au point de départ du circuit auquel ils appartiennent et se déplacent suivant l'itinéraire pré-établi. Tout au long de l'animation, l'utilisateur peut accélérer, ralentir et suspendre le processus avec le bouton glissant.

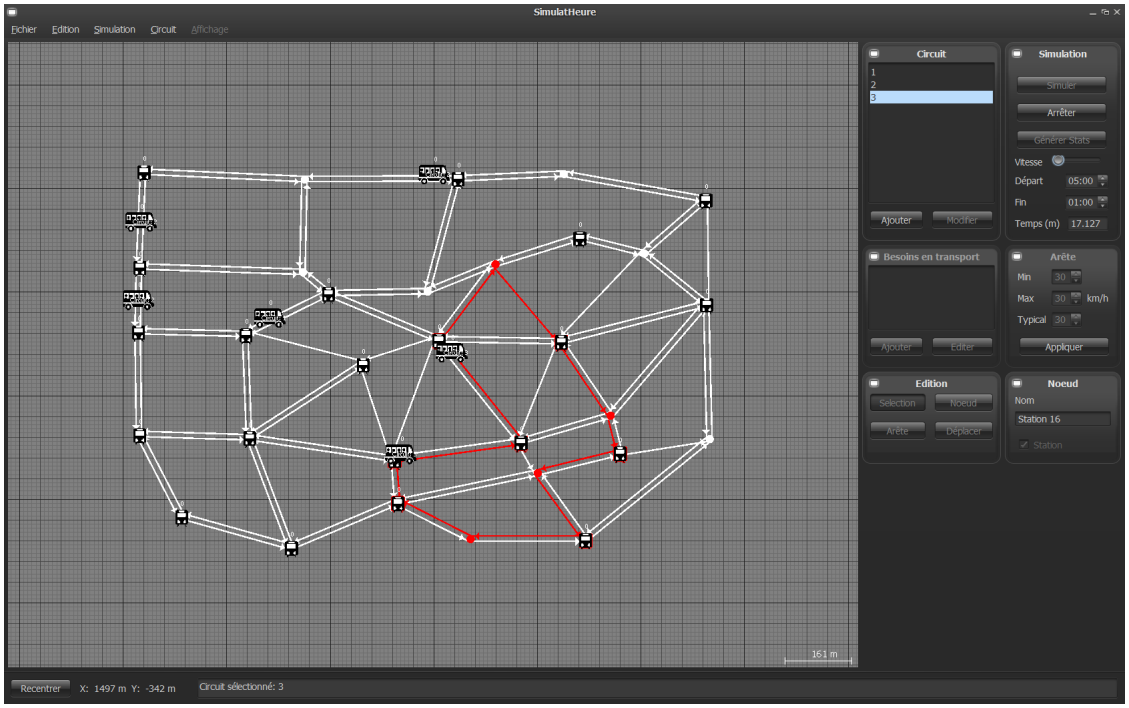


FIGURE 1.6 – Zoom-out du graphe en simulation

À tout moment, il est possible de "dézoomer" le graphe, afin d'avoir un vue plus d'ensemble du réseau créé en roulant la molette de la souris. Cette pratique, largement commune aux logiciels, est pratiquement une commande par défaut dans l'esprit d'un utilisateur standard. De plus, l'ensemble du réseau peut être déplacé latéralement par "glissement", une façon intuitive de naviguer des éléments graphiquement denses.

## 1.3 Diagrammes de séquence

### 1.3.1 Ajout d'un noeud

Le diagramme suivant montre la séquence de l'ajout d'un noeud à la simulation.

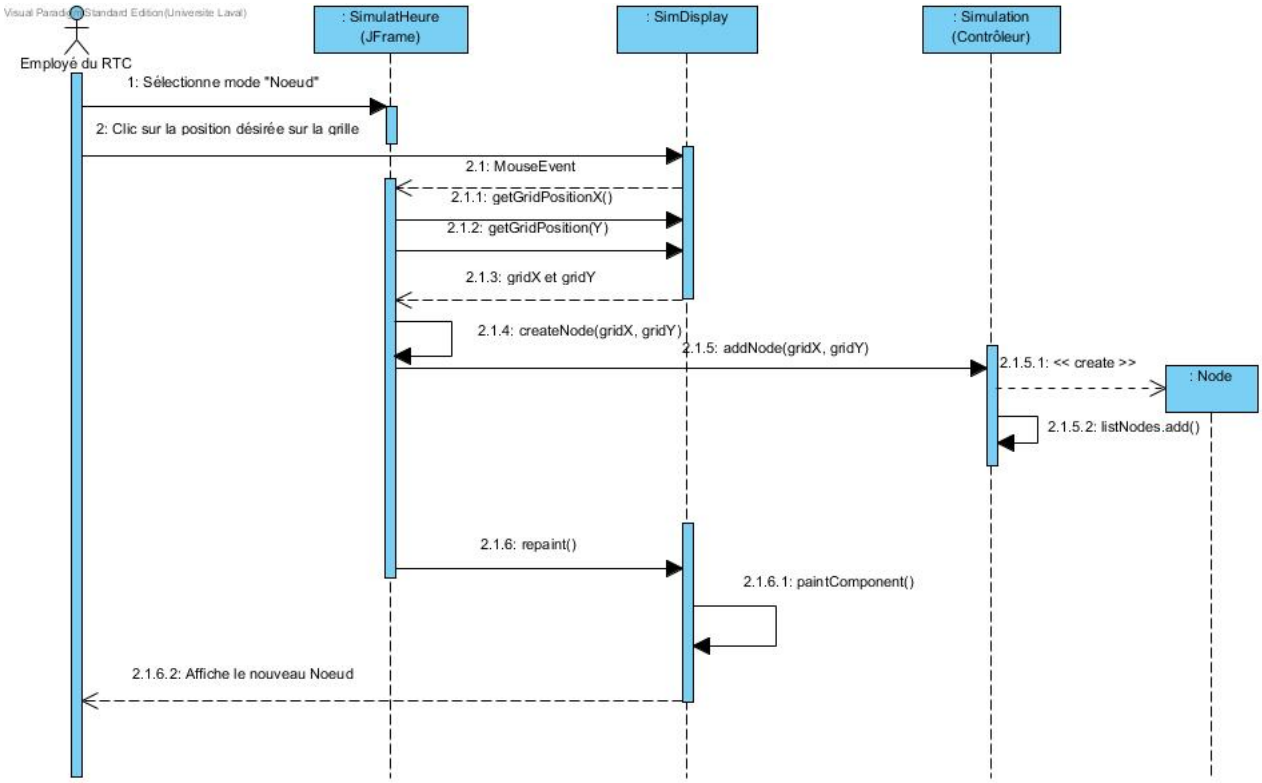


FIGURE 1.7 – Diagramme de séquence : ajout d'un noeud

Lors de l'ajout d'un noeud, l'employé du RTC sélectionne le mode "Noeud" dans l'interface gratuite. Après, celui-ci clique sur la carte. Lors du clic, un MouseEvent est déclenché. Les positions X et Y sont obtenues avec les fonctions getGridPositionX() et getGridPositionY() de l'affichage.

Par la suite, la fonction createNode() est appellée, où un noeud est créé à l'aide des coordonnées et rajouté à la liste de noeuds du contrôleur. L'affichage est ensuite rafraîchi afin d'inclure le nouveau noeud. Le contrôleur peut obtenir la position des noeuds à partir de la liste de noeuds.

### 1.3.2 Ajout d'une arête

Le diagramme suivant montre la séquence de l'ajout d'un objet Line à la simulation à partir de deux points (noeuds).

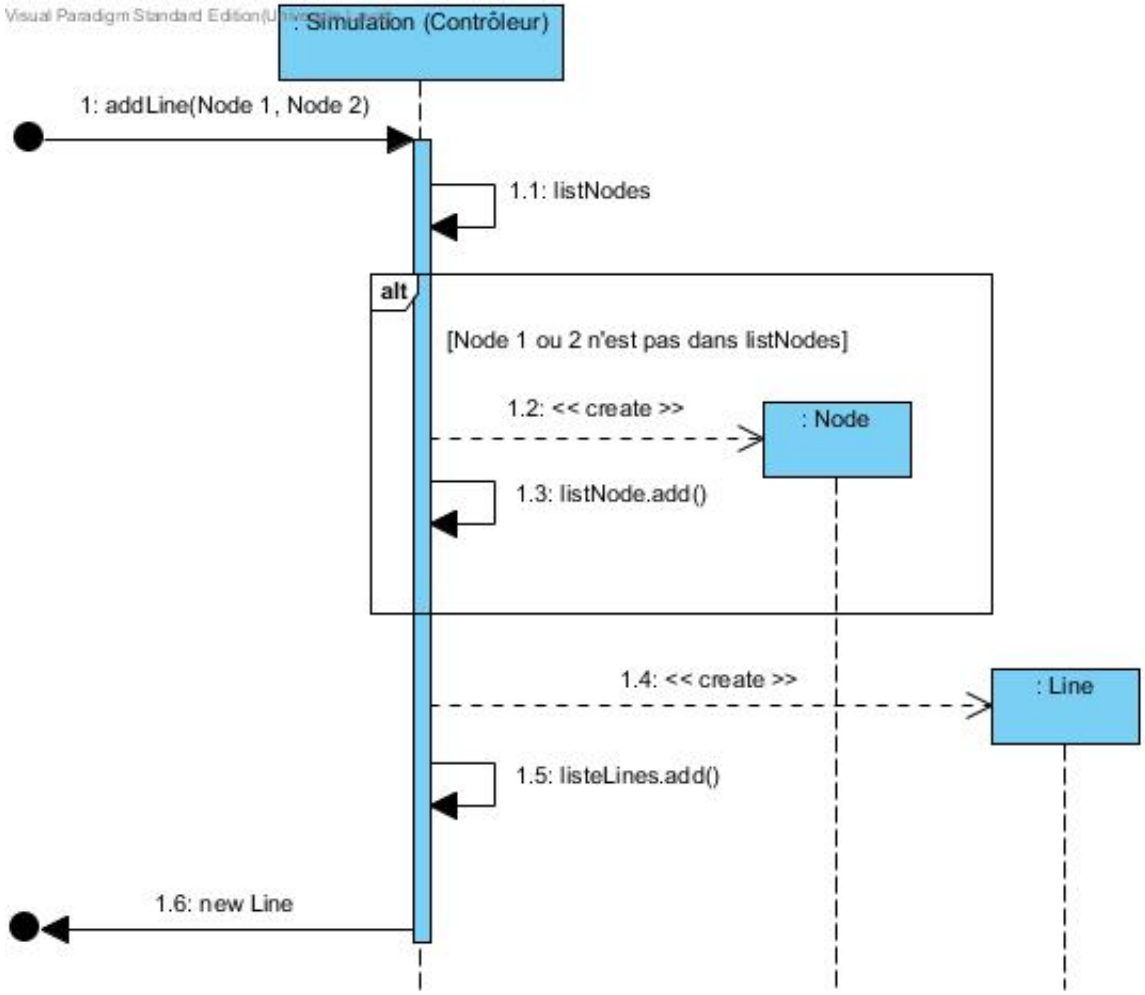


FIGURE 1.8 – Diagramme de séquence : ajout d'une arête

La fonction addLine() du contrôleur est appellée. Les deux noeuds passées en argument sont cherchées dans la liste de nodes. Si pour quelconque raison un des noeuds n'est pas trouvé dans la liste de noeuds, celui-ci est rajouté à la liste de noeuds du contrôleur. Par la suite, un objet Line est créé rejoignant les deux noeuds. Celui-ci est rajouté à la liste de noeuds.

### 1.3.3 Simuler

Le diagramme suivant montre la séquence du processus de démarrage d'une simulation par l'usager.

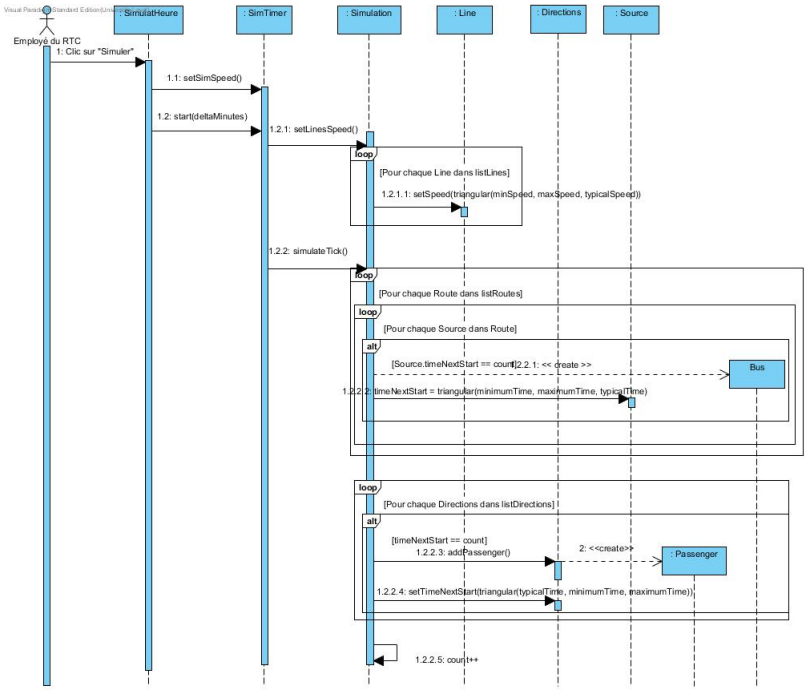


FIGURE 1.9 – Diagramme de séquence : démarrage d'une simulation

La fonction simulateTick() est utilisée comme un incrément de simulation. Dans le diagramme, la fonction est seulement appellée une seule fois puisque cela correspond au premier cycle de la simulation quand elle est démarrée. Durant cette ittération de simulation, les autobus et les passagers qui sont programmés pour apparaître au temps de départ font leur entrée. De plus, les prochains temps de création de bus et de passagers sont calculés à l'aide de la fonction triangular(). Ce n'est que dans les appelles suivant de simulateTick() dans une boucle correspondant au temps demandé de simulation que les bus vont commencer à se déplacer et où les passagers vont embarquer/débarquer des autobus.

### 1.3.4 Distribution triangulaire

Le diagramme suivant montre comment la fonction triangular() est utilisé afin de déterminer la vitesse d'un segment.

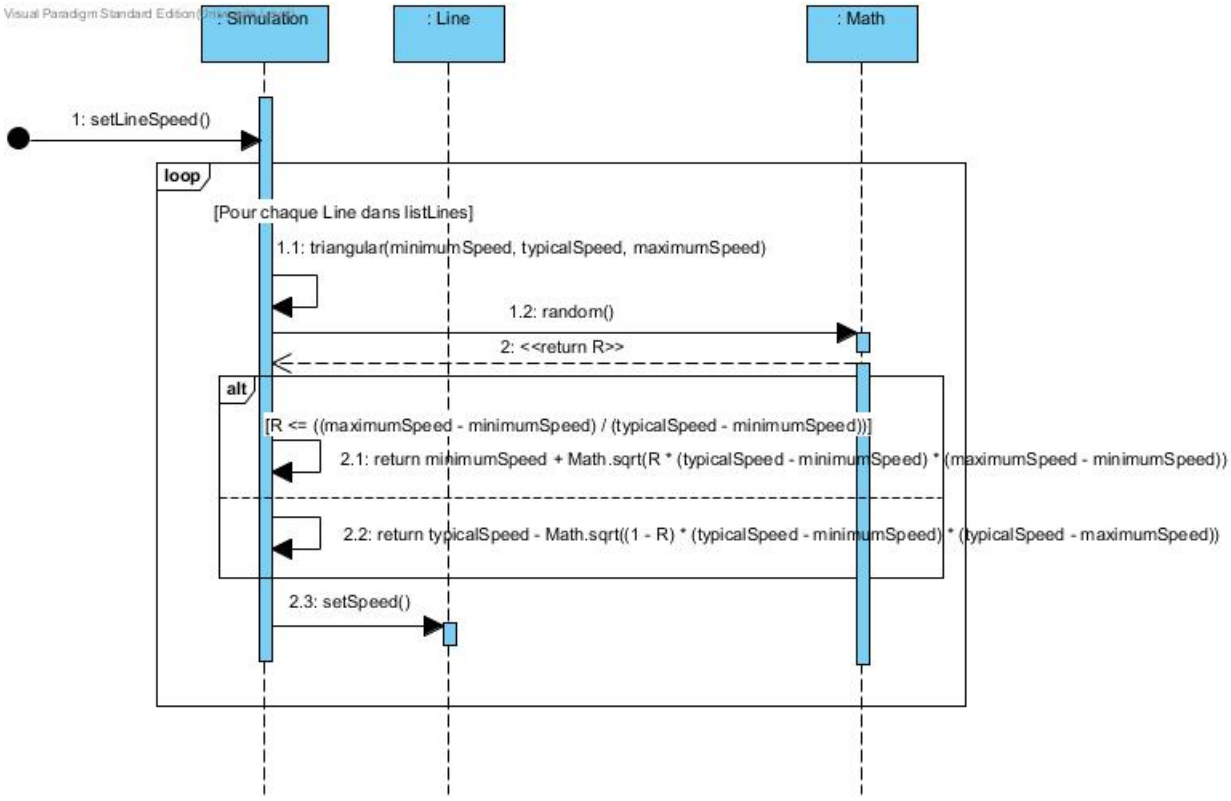


FIGURE 1.10 – Diagramme de séquence : distribution triangulaire de l'assignation de la vitesse à un segment

La fonction triangular prend la vitesse minimum, typique et maximum configurée pour le segment et calcul alors une valeur aléatoire à partir de la distribution triangulaire afin de déterminer la vitesse des autobus sur le segment lors de la simulation.

### 1.3.5 Position d'un autobus

Le diagramme suivant montre comment sélectionner un autobus sur l'interface et récupérer la position d'un autobus.

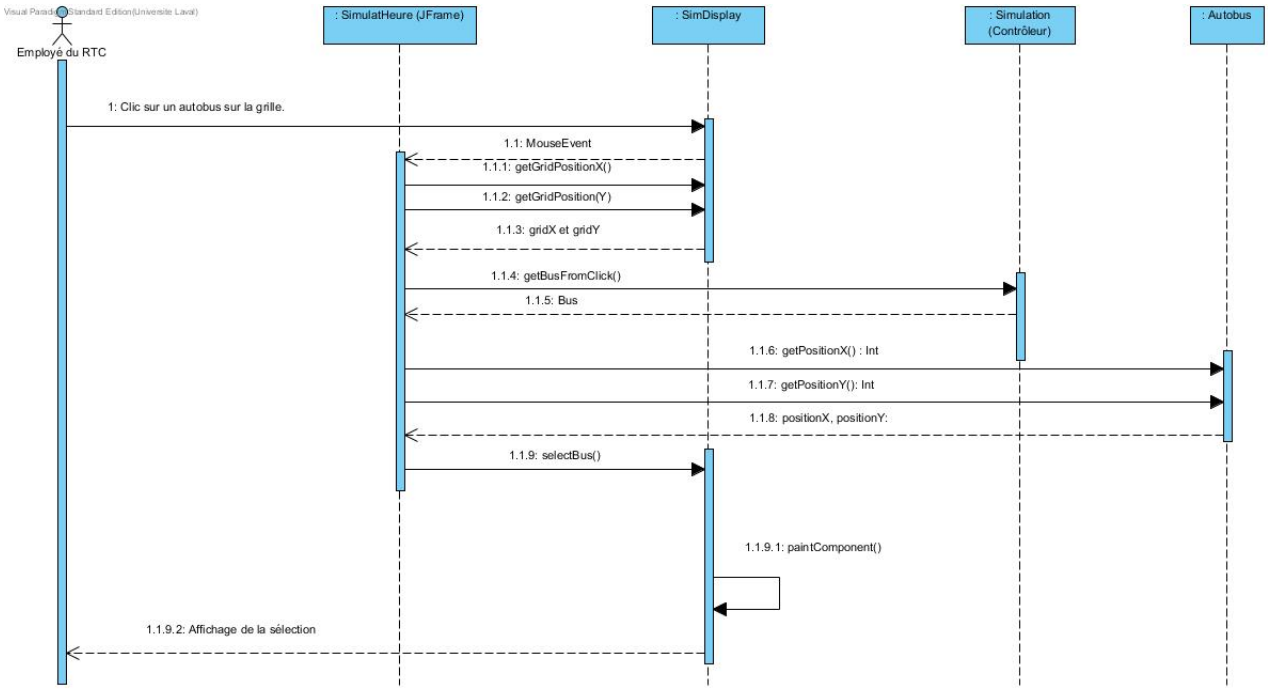


FIGURE 1.11 – Diagramme de séquence : Position d'un autobus

L'utilisateur clique sur un autobus dans la fenêtre de simulation alors que cette dernière est en cours. Lors du clic, un MouseEvent est déclenché. Les positions X et Y sont obtenues avec les fonctions getGridPositionX() et getGridPositionY() de l'affichage. Comme le bus est prioritaire (en premier plan) la fonction de sélection appelle getBusFromClick() qui retourne l'objet bus approprié selon la position dans la liste de bus du contrôleur de simulation.

La valeur de position centrale de l'autobus peut-être obtenu à l'aide des fonctions getPositionX() et getPositionY() de l'autobus sélectionné, cet appel peut être fait continuellement pour mettre à jour la position de l'autobus sur l'interface. La fonction selectBus() et displaySim() se charge conjointement du changement de couleur de l'autobus dans l'interface lors de la sélection.

## 1.4 Diagramme d'états

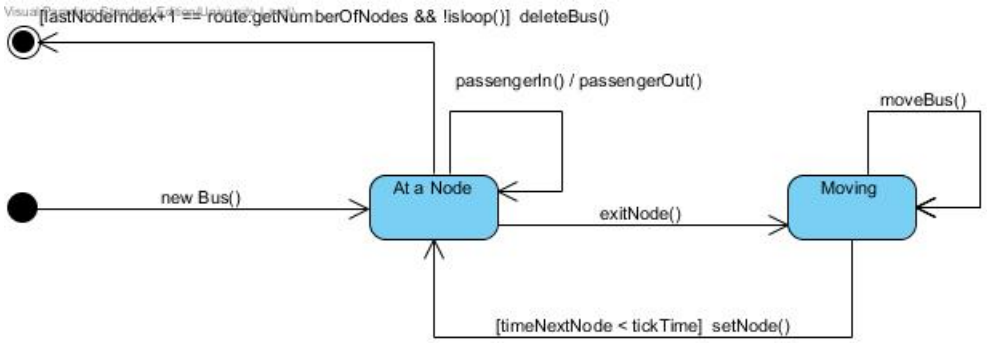


FIGURE 1.12 – Diagramme d'état de SimulatHEURE

Le diagramme présenté dans la figure 1.12 montre le fonctionnement d'une simulation dans laquelle un bus se déplace sur son circuit. Tout d'abord, un objet *Bus* est nécessairement créé avec une position initiale correspondant à celle d'un *Noeud*. Le bus ainsi créé se trouve alors à l'état nommé *At a Node*. En fonction du nombre de *Passagers* ayant un besoin en transport incluant le circuit du *Bus*, celui-ci peut immédiatement se remplir de *Passagers*. À l'itération suivante du programme, le bus entre dans l'état *Moving*, où il reste jusqu'à ce la condition ("temps avant prochain *Noeud*" < "Temps d'un *Tick*") soit remplie. On considère alors que le bus se trouve de nouveau à l'état *At a Node*.

À l'instant (*Tick*) où la position d'un bus correspond à celle d'une station, plusieurs situations peuvent survenir. Dans le cas où le bus contient un ou des passagers dont le besoin en transport (destination) correspond au n ?ud actuel, ceux-ci "sortent" du bus, décrémentant son nombre de passagers actuels. Inversement, tout comme au *Node* précédent, d'autres passagers peuvent "entrer" dans le bus. Finalement, si la condition ("Est dernier noeud du présent circuit" AND "Circuit n'est pas une *Loop*") est remplie, l'objet bus est supprimé à l'itération suivante. (La vie d'aucun passager n'est en danger, bien sûr !) La condition "Loop" détermine si le circuit est circulaire, c'est à dire que son n ?ud de départ correspond à son n ?ud d'arrivé et que les bus doivent y continuer leur parcours plutôt que d'être supprimés.

## 1.5 Diagramme d'activité

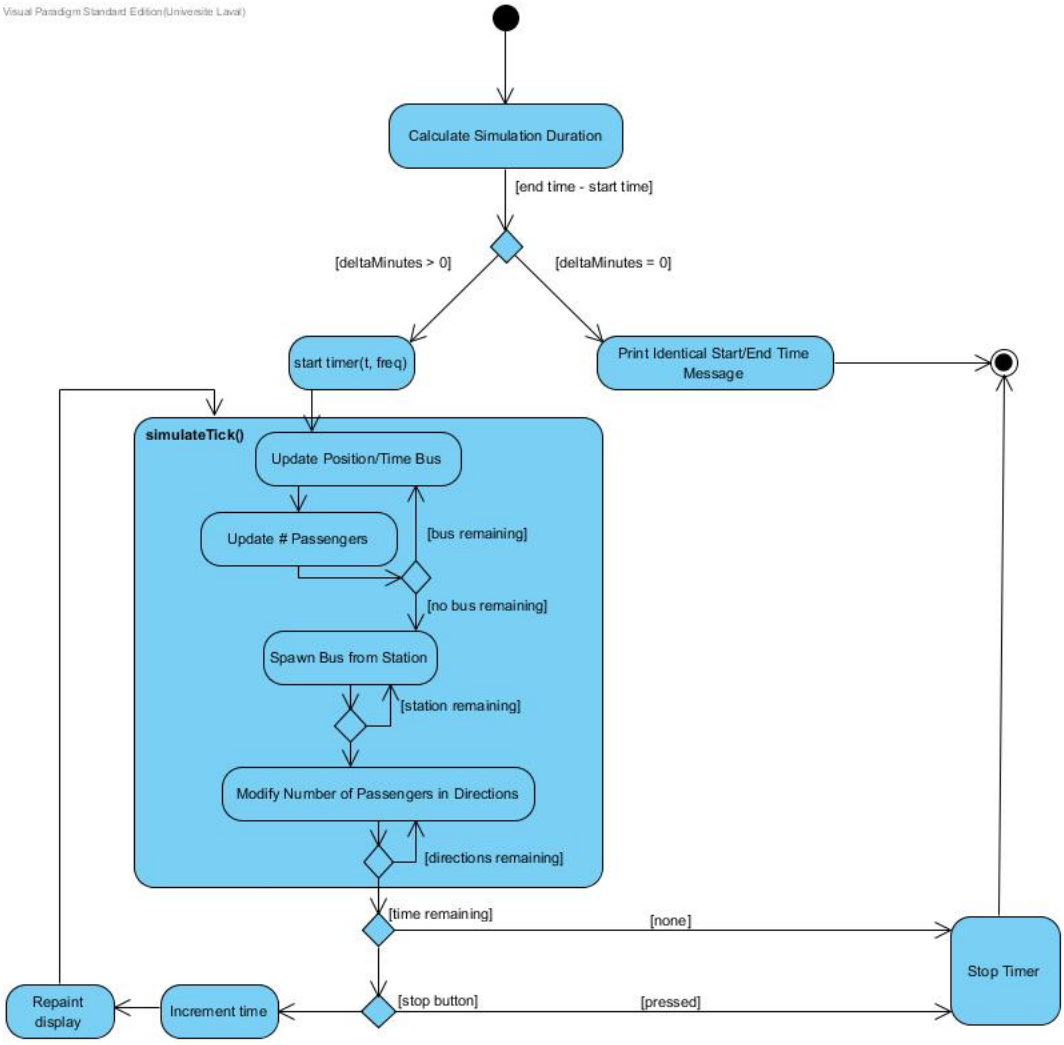


FIGURE 1.13 – Diagramme d'activité de SimulatHEURE

Lors du démarrage de la simulation, les heures de départ et de fin sont sélectionnées. Si celles-ci sont égales, un message d'erreur est affiché, sinon la simulation débute. Le timer est démarré avec un durée (t) et une fréquences (freq) afin de déterminer la longueur d'un tick. À chaque "tick", les positions de chaque bus et la durée de leur trajet sont actualisés. Dans chaque station, les bus apparaîssent à chaque station selon la fréquence de chacune. Par la suite, pour chaque trajet, le nombre de passagers est modifié. Si le temps actuel est égal au temps de fin, ou si le bouton stop est appuyé, la simulation arrête. Sinon, le temps est incrémenté, l'affichage remis à jour et l'exécution continue.

# Chapitre 2

# Architecture logique

## 2.1 Diagramme de package

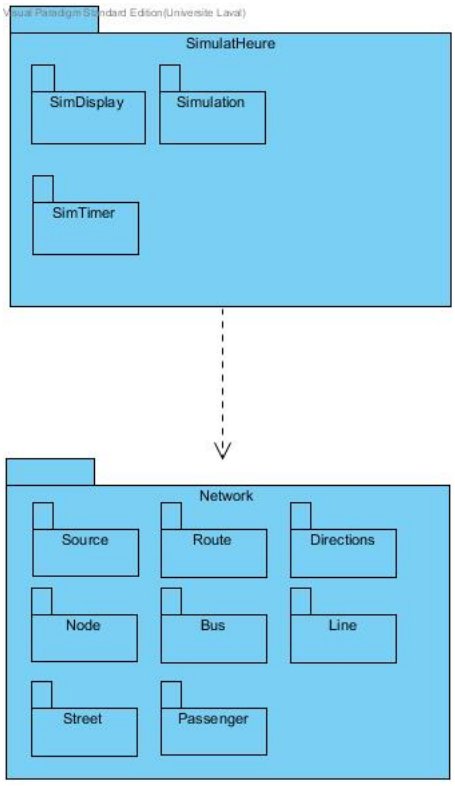


FIGURE 2.1 – Diagramme de package de SimulatHEURE

La figure 2.1 présente la structure logique des modules du programme. Le module principal, nommé SimulatHeure, est composé des classes SimDisplay et SimTimer et Simulation, dont le fonctionnement est expliqué au paragraphe suivant. Le Module SimulatHeure appelle et contrôle un second module gérant le fonctionnement du réseau routier, d'où son nom

"Network". Ce dernier est constitué de tout les éléments composant un réseau d'autobus, incluant *Node*, *Line*, *Source*, *Directions*, *Route*, *Bus* et *Passenger*. L'interaction des classes dans Network est présenté en détail plus loin.

La classe *Simulation* est le cerveau de la simulation. C'est elle qui contient les listes des différentes composantes du réseau routier. Elle permet d'ajouter, de supprimer et de faire bouger les différentes composantes du réseau routier. La classe *SimDisplay*, incorporée dans le module SimulatHeure, prend en charge l'affichage en temps réel de la simulation des transports. Son champs d'actions couvre entre autres le dessin des éléments du réseau sur la fenêtre de simulation, la gestion des contrôles usagers avec la souris (déplacement de la vue, zoom, dezoom, selection, etc.) ainsi que l'affichage de la grille d'alignement dynamique. La classe *SimTimer* permet d'appeller périodiquement la fonction simulateTick() de *Simulation*, ce qui met en marche la simulation.

Les classes dans *Network* sont toutes des classes purement logique, c'est à dire qu'elle n'ont aucun lien avec l'interface utilisateur. Les divers éléments tels les noeuds, les segments où les bus possèdent des attributs spatiaux, correspondant à leur position sur une grille virtuelle, qui doit être configurée de façon externe, ici par un contrôlleur et un panneau d'affichage graphique. C'est le contrôlleur qui permet de faire intéragir les différentes classes ensemble par leurs fonctions.

# Chapitre 3

# Annexe - Vision

## 3.1 Introduction

Ce projet a pour objectif de développer l'application SimulatHEURE, commanditée par le réseau de transport de la capitale de la ville de Québec (RTC). Celle-ci permet à un utilisateur de créer, modifier et simuler un modèle de réseau de transport en commun de façon conviviale et efficace, afin d'analyser ses performances.

## 3.2 Opportunité commerciale

En plus de permettre l'optimisation et l'amélioration du RTC pour ses clients, l'application permet potentiellement à n'importe quel réseau de transport de simuler le développement de son réseau. En milieu urbain où la densité de population ne cesse de croitre, les réseaux de transport en commun sont vitaux au bon fonctionnement du système de transport dans son ensemble. Cette croissance démographique entraîne de façon naturelle le développement de réseaux de transport partout à travers le monde, ce qui offre un marché vaste et en pleine croissance à l'application SimulatHEURE.

## 3.3 Énoncé du problème

Les réseaux de transport en commun se complexifient de plus en plus, ce qui rend difficile leur gestion ainsi que l'analyse de leur efficacité. Il peut être difficile de déterminer les améliorations et les correctifs à apporter, entrainant des coûts sans preuve convaincante du succès de ces modifications. Le simulateur vise la résolution de ce problème.

| Objectifs de haut niveau | Priorité | Problème |
|---|---|---|
| Création d'un modèle de réseau de transport réaliste et simulable | Haute | Difficulté, complexité et temps nécessaire à l'analyse d'un réseau de transport sur le terrain |
| Simuler avec des paramètres variables le modèle du réseau | Haute | Identifier les modifications qui seront bénéfiques au réseau de transport |
| Générer des statistiques et des données de simulation (temps de transit, nombre de véhicules en circulation...) | Haute | Comparer objectivement et rapidement les différentes configurations possibles d'un réseau |
| Offrir une interface utilisateur simple et rapide | Moyenne | Rendre accessible à n'importe quel utilisateur la prise en charge du logiciel et de ses fonctionnalités |

TABLEAU 3.1 – Objectifs de haut niveau

## 3.4 Fonctionnalités

**Création et modifications**

- Création d'un réseau de transport en commun composé de passagers, stations, segments, véhicules et circuits.
- Chaque éléments du réseau est paramétrable.
- Interface graphique avec entrées clavier/souris, permettant la création d'un réseau avec coordonnées géographique sur un canvas vide.
- Sauvegarde et chargement d'un réseau et des résultats de simulation.

**Simulation**

- Affichage graphique du réseau et de ses composantes lors d'une simulation.
- Contrôle temporel de la simulation (Démarrer, pause, ralentir, accélérer).
- Génération de temps aléatoires pour le temps d'arrivée des passagers, le temps entre les autobus et le temps pour franchir un segment.
- Création de statistiques sur le temps minimum, moyen et maximum d'un trajet donné.

# Chapitre 4

# Annexe - Besoins

Le projet SimulatHEURE est conçu dans l'optique de permettre à son utilisateur de simuler un réseau de transport. Afin de satisfaire cette exigence, il est nécessaire de le concevoir en fonction des besoins du client. Dans cette section, divers besoins (*requirements*) sont présentés et détaillés.

## 4.1 Modèle d'utilisation

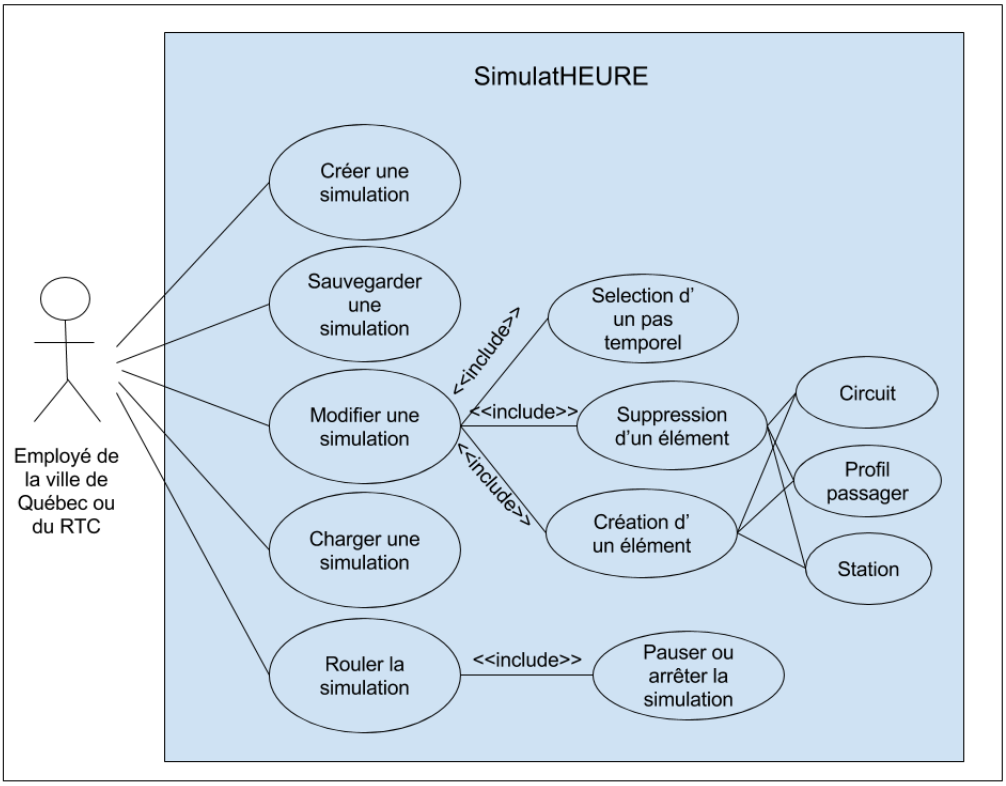


FIGURE 4.1 – Diagramme du modèle d'utilisation

SimulatHEURE doit contenir une interface utilisateur permettant au client de modifier, utiliser, sauvegarder et charger les simulations. Un modèle d'utilisation décrivant l'interaction entre le client et le simulateur est fourni dans cette section. Celui-ci inclut l'acteur principal, les garanties de succès, l'utilisation classique ainsi que les extensions et exigences spéciales devant être supportées.

### 4.1.1 Acteur principal

Employé de la ville de Québec ou du réseau de transport de la capitale (RTC).

### 4.1.2 Parties prenantes

Le client veut un simulateur dont la vitesse peut être personnalisée (y compris l'arrêt ou le redémarrage), et qui respecte les conditions suivantes :

1. Les passagers et véhicules apparaissent au bon moment.
2. Il est possible d'observer le nombre de passagers dans un véhicule en tout temps.

3. Les véhicules doivent se déplacer visuellement durant la simulation, disparaissant arrivé à destination (sauf en cas de boucle).

### 4.1.3 Cas #1 : Création et sauvegarde d'une simulation

#### 4.1.3.1 Garantie de succès

En aucun cas il est possible d'avoir un circuit ou un profil de passager sans point de départ, de fin ou d'intersection.

#### 4.1.3.2 Scénario usuel

1. Le client débute le logiciel.
2. Le client place à la souris une série de points correspondant à des intersections et/ou arrêts d'autobus.
3. Le client définit des circuits en sélection un point d'origine puis une série de points à franchir de manière consécutive.
4. Le client identifie des profils de passagers avec un point d'origine, un point de destination, ainsi que les segments empruntés lors du trajet.
5. Le client enregistre la simulation en appuyant sur le bouton de sauvegarde.

#### 4.1.3.3 Extensions

1. N'importe quand, le client :
   (a) Retire un arrêt d'un circuit avec le menu contextuel, les intersections sont alors retirées.
   (b) Retire une intersection ou ajoute un arrêt ou une intersection avec le menu contextuel.
   (c) Tente de fermer le logiciel. Le logiciel demande alors si le client veut enregistrer la simulation avant l'arrêt du programme.
      i. Le client choisit oui : Les paramètres de simulation actuels sont enregistrés.
      ii. Le client choisit non : Le logiciel ferme immédiatement.
      iii. Le client choisit d'annuler : Le scénario usuel reprend court.

#### 4.1.3.4 Exigences spéciales

Toutes les manipulations des éléments visuels doivent pouvoir être faites avec la souris.

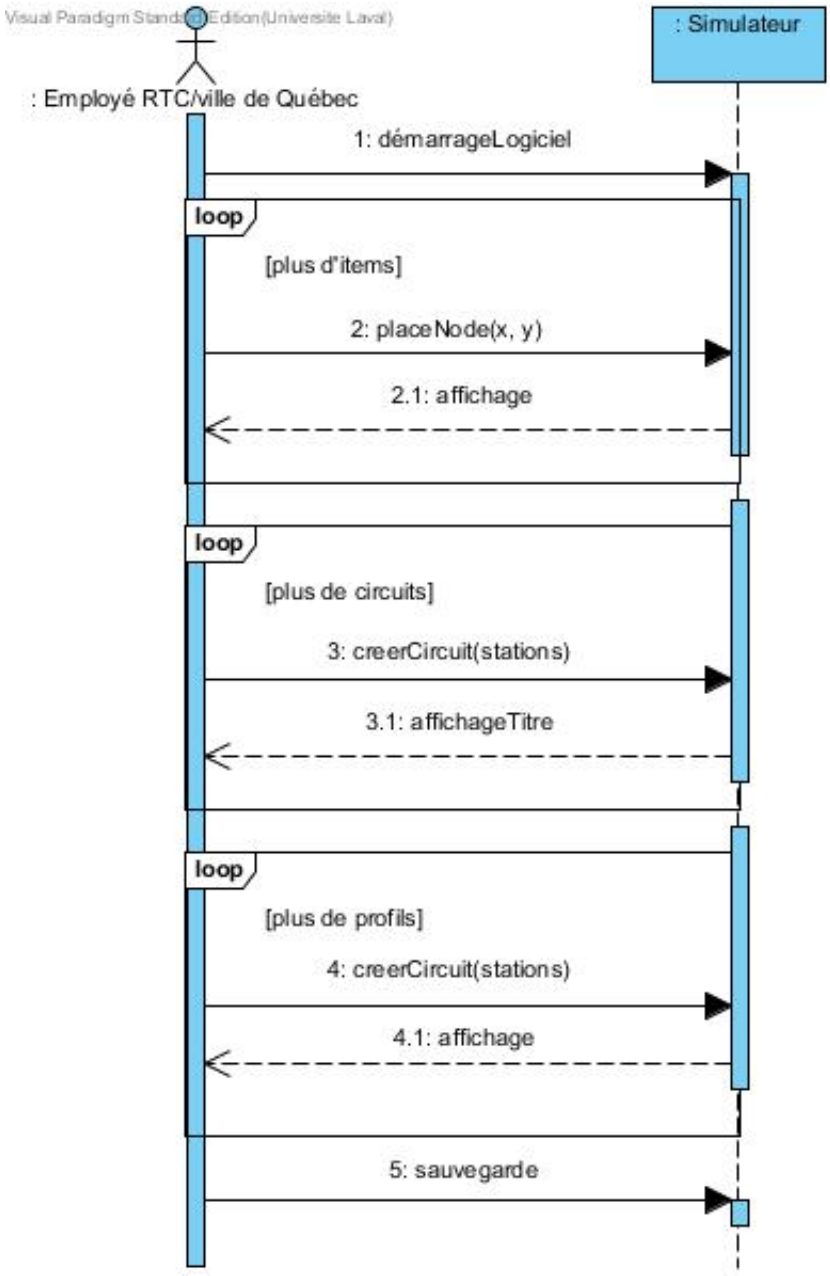


FIGURE 4.2 – Séquence de création et sauvegarde d'une simulation.

### 4.1.4 Cas #2 : Charge d'une simulation

#### 4.1.4.1 Garantie de succès

Les circuits et les profils de passager chargés doivent n'avoir aucun circuit sans point de départ, de fin ou d'intersection.

#### 4.1.4.2 Scénario usuel

1. Le client débute le logiciel.
2. Le client clique sur le bouton pour charger une simulation.
3. Les paramètres de simulations enregistrés sont extraites et appliquées au simulateur.

#### 4.1.4.3 Extensions

1. Lors de la sélection de la simulation enregistrée, le client choisit d'annuler : Le scénario de création d'une simulation suit alors.

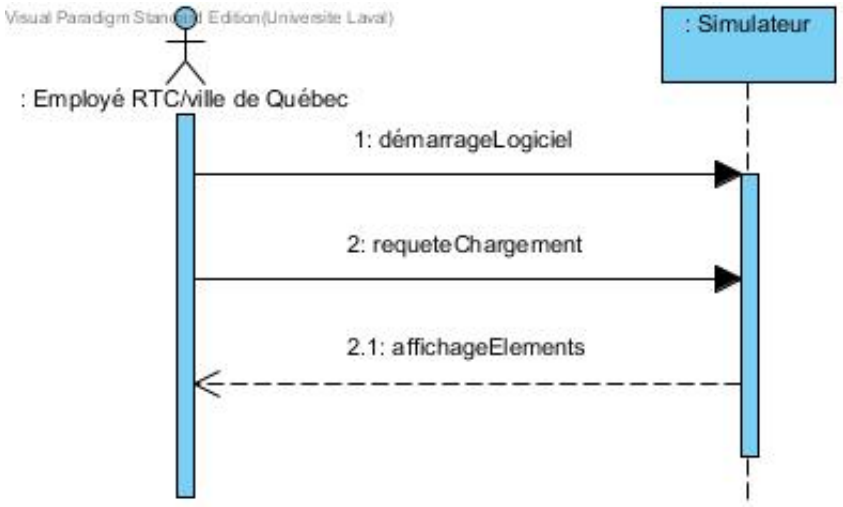


FIGURE 4.3 – Séquence de chargement d'une simulation.

### 4.1.5 Cas #3 : Rouler une simulation

#### 4.1.5.1 Précondition

La simulation contient au minimum un circuit et un profil de passager valides.

#### 4.1.5.2 Garantie de succès

Temps minimal, temps maximal et temps moyen pour franchir une distance sont correctement mesurés et sauvés.

#### 4.1.5.3 Scénario usuel

1. L'employé sélectionne une heure de début et de fin ainsi qu'un pas temporel.
2. L'employé appuie sur le bouton pour démarrer la simulation.
3. Les temps associés à chaque segment du réseau sont sélectionné avec distribution triangulaire.

4. La position des véhicules et leur nombre de passagers se mettent à jour en suivant les trajets.
5. Pour chaque profil de passager, le temps minimal, maximal et moyen durant la simulation est calculé.
6. La simulation prend fin par elle-même lorsque l'heure de fin est atteinte.

#### 4.1.5.4 Extensions

1. N'importe quand durant la simulation, l'employé :
   (a) Clique sur un véhicule afin d'obtenir le nombre de passagers à l'intérieur.
   (b) Déplace sa souris sur la carte, affichant les coordonnées géographiques associées dans la barre d'état.
   (c) Pause ou arrête la simulation.
   (d) Zoom/Dézoom la carte.

#### 4.1.5.5 Exigences spéciales

Toutes les manipulations des éléments visuels doivent pouvoir être faites avec la souris.

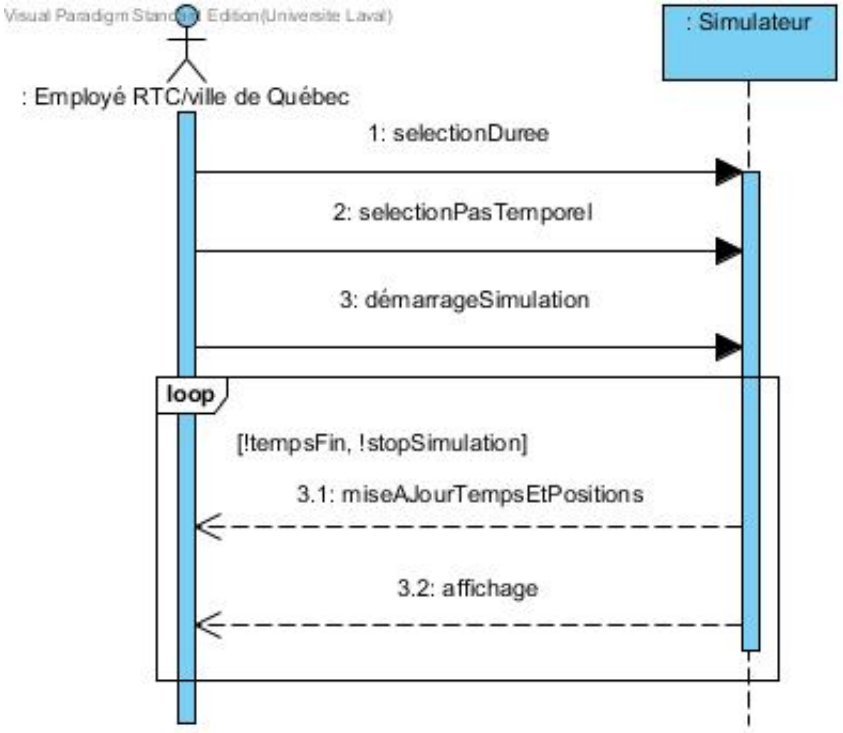


FIGURE 4.4 – Séquence d'exécution d'une simulation.

### 4.1.6 Cas #4 : Suppression d'un élément

#### 4.1.6.1 Garantie de succès

Les circuits et les profils de passager chargés doivent n'avoir aucun circuit sans point de départ, de fin ou d'intersection.

#### 4.1.6.2 Scénario usuel

1. Le client charge une simulation existante.
2. Le client sélectionne un élément en appuyant sur le bouton sélection puis sur un élément.
3. Le client appuie sur le bouton supprimer.
4. L'élément est retiré de la simulation.

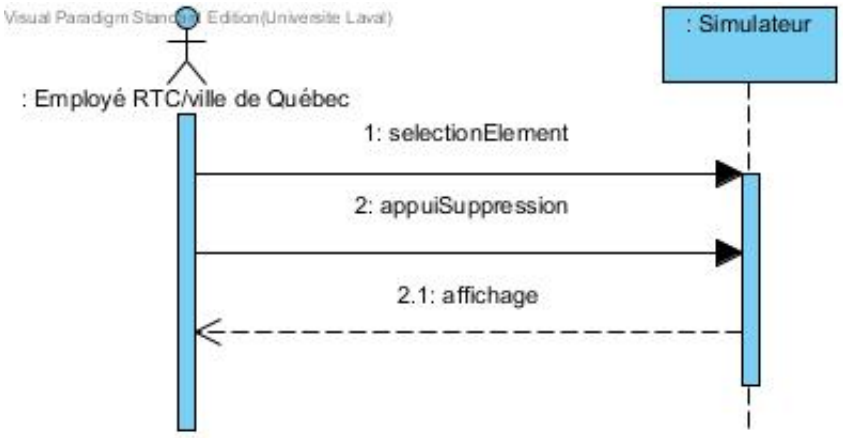


FIGURE 4.5 – Séquence de suppression d'un élément.

### 4.1.7 Cas #5 : Création d'un élément

#### 4.1.7.1 Garantie de succès

Les circuits et les profils de passager chargés doivent n'avoir aucun circuit sans point de départ, de fin ou d'intersection.

#### 4.1.7.2 Scénario usuel

1. Le client charge une simulation existente.
2. Le client sélectionne un élément dans l'interface.
3. Le client clique sur un point dans le plan pour rajouter l'élément.
4. L'élément est inséré dans la simulation.

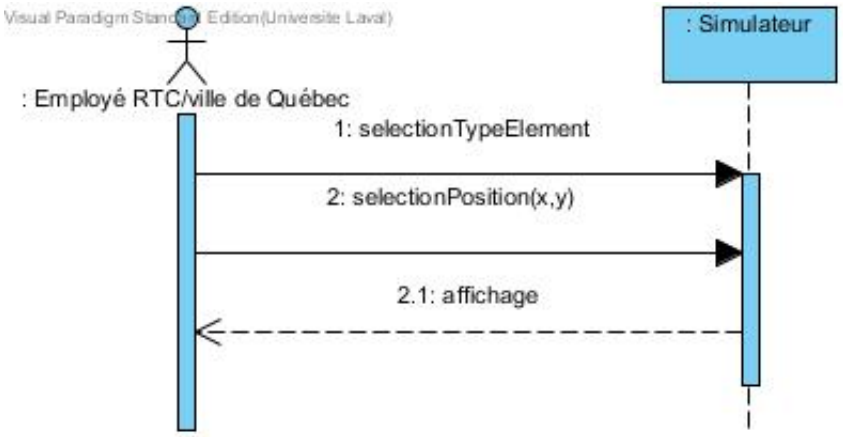


FIGURE 4.6 – Séquence de création d'un élément.

## 4.2 Spécifications supplémentaires

Au delà des besoins du client spécifiées à la section 4.1, certaines spécifications moins évidentes mais néanmoins essentielles au projet sont présentées dans cette section. Celles-ci sont introduites en ordre décroissant d'importance vis-à-vis du projet.

### 4.2.1 Fiabilité et précision

En assumant que l'utilisateur final maîtrise l'outil de simulation, celui-ci ne prendra pas probablement pas la peine de vérifier les résultats obtenus à la suite d'une simulation. Ainsi, il est essentiel, considérant l'utilité principale du simulateur, que celui-ci soit parfaitement calibré afin de fournir des résultats exacts de façon consistante.

### 4.2.2 Fonctionnalité

La pertinence d'un simulateur, outre l'expérimentation virtuelle de situations réelles et complexes, réside entre autre dans sa capacité à tester la performance d'un modèle. Dans le cas présent, le simulateur routier doit offrir à l'utilisateur tout les outils nécessaires à l'élaboration, aux tests et à l'évaluation de la performance de son modèle de conception.

### 4.2.3 Convivialité (Usabilité)

Tout produit informatique, quel qu'il soit, se démarque de ses compétiteurs en grande partie par sa facilité d'utilisation. Dans le cas d'un programme "end-user", soit avec une interface graphique (*GUI*), celle-ci devient parfois l'élément déterminant du succès ou de l'échec commercial du produit. Il est donc essentiel que le temps et les ressources appropriées soient assignées au design et au test des fonctionnalités disponibles sur l'interface, avec le plus grand soucis de simplicité et d'esthétique.

### 4.2.4 Versatilité d'application

L'utilisateur principal du simulateur étant un employé du RTC, il est néanmoins important de considérer qu'un tel outil de simulation est d'utilité générale. Ainsi, son développement doit s'orienter afin de cibler un auditoire large, employé municipal ou non et peu importe ses besoins spécifiques. Certaines fonctionnalités de SimulatHEURE, bien que non-essentielles à une personne souhaitant simuler un réseau de transport spécifique, sont donc implémentées à cet effet. À titre d'exemple, on peut citer la capacité de visualiser nombre d'information supplémentaires aux résultats d'une simulation.

## 4.3 Glossaire

**Segment :** Segment reliant deux stations dans le réseau

**Circuit :** Ensemble de stations et de segments formant une boucle ou un aller simple dans le réseau

**Besoin en transport :** Ensemble des besoins pour le déplacement d'un passager

**GUI :** Interface graphique utilisateur du programme

# Chapitre 5

# Annexe - Modèle du domaine

## 5.1 Diagramme du modèle du domaine

Le modèle du domaine permet d'identifier les classes conceptuelles du projet SimulatHEURE ainsi que ses principaux attributs crées lors de la création d'une simulation dans le programme. Le diagramme du modèle du domaine est représenté à la figure 5.1

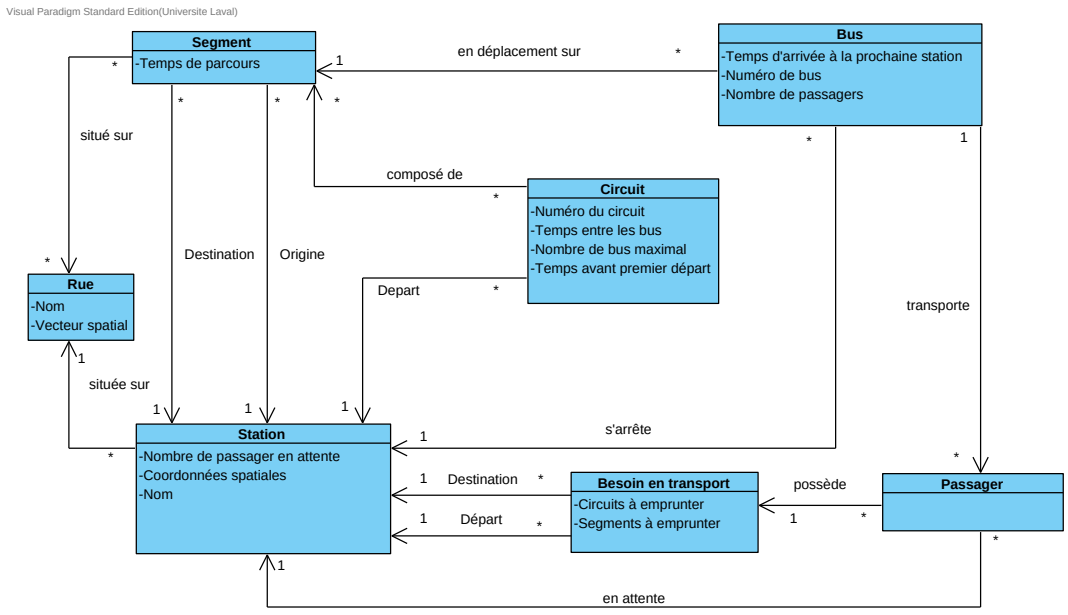


FIGURE 5.1 – Modèle du domaine SimulatHEURE