الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

N° Réf :……

Centre Universitaire

ABD ELHAFID BOUSSOUF MILA

Institut Mathématiques et Informatique **Département Informatique**

Mémoire préparé en vue de l'obtention du diplôme de Master

En: Informatique

Spécialité : Sciences et Technologies de l'Information et de la Communication (STIC)

Un système Multi-Agent pour la gestion d'encombrement dans le trafic routier

Préparé par : Sedjal Raouia

Zemieche Nassira

Soutenue devant le jury

Président : Lalouci Ali, Professeur au centre universitaire **Abdelhafid Boussouf**

Examinateur : Attia Mourad, Professeur au centre universitaire **Abdelhafid Boussouf**

Encadrant : Hettab Abdelkamel, Professeur au centre universitaire **Abdelhafid Boussouf**

Année universitaire : 2023/2024

Remerciement

Nous remercions d'abord et avant tout Allah qui nous a donné le courage, la santé, la possibilité et la patience pour réaliser ce travail.

Un remerciement particulier à notre encadrant Monsieur <**Abdelkamel Hettab** > pour son soutient, son sérieux, sa disponibilité, ses précieux conseils et son aide tout au long de l'élaboration de ce travail.

Nous remercions également, les membres du jury d'avoir accepté d'examiner et d'évaluer notre travail.

Sans oublier tous les enseignants du département d'informatique pour la qualité de l'enseignement qu'ils ont bien voulu nous prodiguer durant nos études afin de nous fournir une formation efficiente.

Nous n'aurions garde d'oublier tous ceux qui ont contribué de près ou de loin à la réalisation de ce mémoire et à tous ceux qui ont partagé avec nous les moments les plus difficiles dans la réalisation de ce travail et tous ceux qui nous souhaite le bon courage.

Finalement, nous remercions très sincèrement tous nos familles pour leur encouragement sans limite.

Raouia et Nassira

الاهداء

<<وآخر دعواهم أن الحمد لله رب العالمين>>

الحمد لله حباً وشكراً وامتناناً على البدء والختام

لم تكن الرحلة قصيرة ولا ينبغي لها أن تكون، لم يكن الحلم قريباً ولا الطريق كان محفوفاً بالتسهيلات لكنني تخرجت فاللهم لك الحمد قبل أن ترضى ولك الحمد اذا رضيت ولك الحمد بعد الرضا لأنك وفقتني على إتمام هذا النجاح

أهدي هذا النجاح لنفسي أولا، وبكل حب أهدي ثمرة تخرجي

إلى من زين إسمي بأجمل الألقاب إلى من كلله الله بالهيبة والوقار إلى من أحمل اسمه بكل فخر إلى ذلك الرجل العظيم الذي بذل كل ما بوسعه ملاذي بعد الله فخري و اعتزازي، مأمني الوحيد وفرحتي الدائمة

والدي الحبيب **" زميش المختار "** متعه الله بالصحة والعافية

إلى من علمتني الأخلاق قبل الحروف نبراس أيامي و وهج حياتي إلى معنى الحب والتفاني إلى بسمة الحياة وسر الوجود إلى من كان دعاؤها سر نجاحي وحنانها بلسم جراحي وجهتي التي أستمد منها القوة

والدتي الحبيبة **"شريفي فاطمة** "متعها الله بالصحة والعافية

إلى من تحلت بالأمومة و وتميزت بالحنان والعطاء إلى التي ساندتني ، وأعطت بلا مقابل ، وقدمت لي الدعم لمواصلة طريقي قدوتي**" أختي لمياء"**

إلى الأعمدة الثابتة في الحياة الداعمين الساندين أرضي الصلبة وجداري المتين ،إلى من مدت أياديهم في أوقات ضعفي فارتخيت واقفين خلفي إلى من شد الله بهم عضدي فكانوا خير معين

أخواتي (**"سولاف" ،"عبلة" ،"صونيا" ،"شهرة" ،"رتيبة" ،"عفاف" ،"ملاك"**)

إخوتي(**"هارون" ،"منير" ،"يوسف" ،"مفدي"**)

إلى ركني العظيم في الحياة أختي التي لم تلدها أمي فكانت لي ينابيع أرتوي منها ،إلى حيرة أيامي وصفوتها إبنة عمي**" أحلام"**

إلى رفقاء السنين و أصحاب الشدائد إلى من أفاضوني بمشاعرهن ونصائحهن المخلصة إليكن

بنات عمي (**"وزينة" ، "دلال" ، "نادية" ، "صفاء"، "مروة" ، "لبنى"**)

إلى رفاق الدرب يا من جعلتن رحلة الدراسة ممتعة مليئة بالذكريات صديقاتي الغاليات كل واحدة باسمها

إلى كل من سعى معي لإتمام هذه المسيرة دمتم لي سنداً لا عمر له

نصيرة

الاهداء

من قال أنا لها ...نالها

وأنا لها وإن أبت رغما عنها أتيت بيها

لم تكن الرحلة قصيرة ولا ينبغي لها أن تكون، ولم يكن الحلم قريبا

ولا الطريق كان محفوفا بالتسهيلات لكنني فعلتها ونلتها.

شكرا موصولا لنفسي على الصبر والعزيمة والإصرار لذلك أهدي هذا النجاح لنفسي الطموحة أولا بدأت بطموح وانتهت بنجاح بعد أن قضيت وقت طويل أركض خوفا أن يفوتني شيء فقد أدركت ألا يفوتني شيء قد كتبه الله لي

كان فضل الله عظيما رجوت كريما ووثقت بصنعه

وما كان من يرجو الكريم يخيب

إلى داعمي الأول من شرفني بحمل اسمه وبذل الغالي والنفيس لأجلي واستمديت منه اكتفائي واعتزازي بذاتي

والدي العزيز (**الحاج الطاهر**)

إلى الجسر الصاعد بي إلى الجنة إلى اليد الخفية التي تحملت كل لحظة ألم مررت بها وساندتني عند ضعفي وهزلي

والدتي العزيزة **(بوعلي اسية)**

إلى صحابة الفضل العظيم سندي ومسندي واتكائي وضلعي الثابت الذي لا يميل أختي الغالية حفظها الله **(خلود)**

إلى الذين يبهجهم نجاحي هم سر قوتي ووهج حياتي إلى قرة عيني من تقاسموا معي مسيرتي بدءا من اول خطوة وانتهاء

أخر خطوة **(أخي وأخواتي)**

إلى من جمعتني بيهم الصدفة وأصبحوا أغلى ناسي من جعلوا رحلتي الدراسية ممتعة مليئة بالذكريات وكانوا لي بمثابة

العائلة الثانية (**صديقاتي العزيزات)**

أهديكم هذا الإنجاز وثمرة نجاحي لطالما تمنيته، ها أنا اليوم أتممت أول ثمراته بفضل الله عز وجل

فالحمد لله على ما وهبني، راجية من المولى ان ينفعني بما علمني

وأن يعينني ويجعلني مباركة أينما كنت.

راوية

Résumé

L’encombrement du trafic routier est un problème majeur dans de nombreuses régions, notamment dans les grandes villes. Plusieurs facteurs contribuent à cette situation, tels que des infrastructures routières inadéquates et les nombres élevé de véhicules, en particulier aux heures de pointe : le matin de 7h à 8h, le midi de 12h à 13h et de 15h à 17h

Notre objectif principal est de proposer une approche pratique et adaptable pour gérer l’encombrement en utilisant les capacités des systèmes multi-agents pour prendre des décisions rapides et pertinentes en temps réel. Nous visons à rendre la circulation plus fluide, à réduire la pollution et à rendre la conduite plus agréable et sûre pour tous les usagers de la route

Notre projet propose une simulation basée sur des agents pour gérer le trafic routier. Cette approche implique la coopération entre différents acteurs, tels que les conducteurs et les systèmes de gestion du trafic, afin d’optimiser la circulation et de réduire les embouteillages en ville. Le temps d’attente global des véhicules sur les routes est ensuite évalué en utilisant différentes heuristiques comme FIFO (First In First Out) et deux versions de l’heuristique glouton (Greedy), l’une simple et l’autre avancée. A la fin Une comparaison entre ces trois heuristiques montre que l’heuristique glouton avancée optimise mieux la gestion du trafic routier et diminue les temps d’attente des véhicules

Mots-clés : Gestion du trafic routier, Systèmes multi-agents, Heuristique FIFO, Heuristique glouton, Heuristique glouton avancée, Optimisation du trafic routier, Simulation basée sur des agents

Abstract

Traffic congestion is a major problem in many regions, especially in large cities. Several factors contribute to this situation, such as inadequate road infrastructure and the high number of vehicles, particularly during peak hours: in the morning from 7 a.m. to 8 a.m., at noon from 12 p.m. to 1 p.m., and from 3 p.m. to 5 p.m.

Our main objective is to propose a practical and adaptable approach to managing congestion by using the capabilities of multi-agent systems to make quick and relevant decisions in real time. We aim to make traffic flow smoother, reduce pollution, and make driving more pleasant and safe for all road users.

Our project proposes an agent-based simulation to manage road traffic. This approach involves the cooperation between different actors, such as drivers and traffic management systems, to optimize traffic flow and reduce congestion in cities. The overall waiting time of vehicles on the roads is then evaluated using different heuristics like FIFO (First In First Out) and two versions of the Greedy heuristic, one simple and the other advanced. In the end, a comparison between these three heuristics shows that the advanced Greedy heuristic better optimizes traffic management and reduces vehicle waiting times.

Keywords: Traffic management, Multi-agent systems, FIFO heuristic, Greedy heuristic, Advanced Greedy heuristic, Traffic optimization, Agent-based simulation

ملخص

ازدحام المرور مشكلة رئيسية في العديد من المناطق، لا سيما في المدن الكبيرة. تساهم عدة عوامل في هذه المشكلة، مثل البنية التحتية غير الكافية للطرق والعدد الكبير من المركبات، خاصة خلال ساعات الذروة: في الصباح من الساعة 7 إلى 8 صباحًا، وفي الظهر من الساعة 12 إلى 1 ظهرًا، ومن الساعة 3 إلى 5 مساءً.

هدفنا الرئيسي هو اقتراح نهج عملي وقابل للتكيف لإدارة الازدحام باستخدام قدرات الأنظمة متعددة الوكلاء لاتخاذ قرارات سريعة وملائمة في الوقت الفعلي. نحن نسعى لجعل حركة المرور أكثر سلاسة، وتقليل التلوث، وجعل القيادة أكثر متعة وأمانًا لجميع مستخدمي الطرق.

يقترح مشروعنا محاكاة تعتمد على الوكلاء لإدارة حركة المرور على الطرق. يتضمن هذا النهج التعاون بين مختلف الجهات الفاعلة، مثل السائقين وأنظمة إدارة المرور، من أجل تحسين تدفق حركة المرور وتقليل الازدحام في المدن. يتم بعد ذلك تقييم وقت الانتظار الإجمالي للمركبات على الطرق باستخدام تقنيات مختلفة مثل FIFO (First In First Out) ونسختين من خوارزمية Greedy ، واحدة بسيطة والأخرى متقدمة. في النهاية، تُظهر المقارنة بين هذه الخوارزميات الثلاثة أن خوارزمية Greedy المتقدمة تُحسّن إدارة المرور بشكل أفضل وتقلل من أوقات انتظار المركبات.

.

الكلمات المفتاحية: إدارة المرور، الأنظمة متعددة الوكلاء، خوارزمية FIFO ، خوارزمية الجشع، خوارزمية الجشع المتقدمة، تحسين حركة المرور، المحاكاة المعتمدة على الوكلاء

Table des matières

Chapitre03 : Analyse et conception

Chapitre 04 : Implémentation

Liste des tableaux

Table des figures

Introduction générale

Introduction générale

1. Contexte générale et objectif :

La gestion efficace du trafic routier représente un défi majeur dans les environnements urbains actuels, caractérisés par une densité croissante de véhicules et des infrastructures routières souvent saturées.

L'optimisation des déplacements urbains réduit les pertes de temps et d'énergie, apportant des bénéfices à la société. Avec les avancées informatiques du début du XXIe siècle, il est devenu possible de modéliser et simuler des phénomènes complexes comme le trafic routier. La simulation multi-agent, qui se concentre sur le comportement individuel des véhicules, est particulièrement prometteuse pour ces applications, car elle simule les interactions des véhicules dans des environnements complexes et dynamiques.

La gestion du trafic routier de nombreux véhicules dans une ville, atteignant des centaines ou milliers chaque seconde, nécessite une quantité importante de ressources de calcul et était perçue comme étant impossible jusqu'à récemment. Cependant, l'émergence d'architectures distribuées, telles que les systèmes multi-agents (SMA), a ouvert de nouvelles possibilités pour simuler un tel trafic avant de l'implémenter en réalité. Actuellement, plusieurs simulateurs et travaux ont été proposés **[1, 2, 3, 4, 5]** pour simuler et gérer le trafic routier. Cependant, une partie de ces simulateurs reste éloignée de la réalité, rendant difficile de mettre en place des stratégies de gestion du trafic en temps réel. L'objectif principal est donc de simuler le comportement du trafic routier en utilisant des SMA, en comparant plusieurs algorithmes basés sur différentes heuristiques pour choisir le meilleur. Ensuite, le SMA obtenu serait utilisé comme noyau d'un système réel intégrant des capteurs et actionneurs connectés, communiquant avec les agents du système.

Le principal objectif de ce mémoire est de proposer un système multi-agent agissant d'abord comme simulateur de trafic routier, avec une vision future de servir de base à un système de gestion de trafic réel intégrant des capteurs et des actionneurs interconnectés, communiquant avec les agents du noyau. Dans ce système, chaque agent, appelé conducteur, représente un véhicule qui communique avec un agent coordinateur appelé administrateur. Ce dernier gère le trafic et donne des instructions aux autres agents conducteurs en fonction de la situation globale des véhicules sur les routes, tout en fournissant des informations sur l'état global des routes en réponse aux requêtes des agents conducteurs. Enfin, une étude est menée pour évaluer l'efficacité des heuristiques FIFO (First-In, First-Out) et deux versions de l'heuristique

glouton (Greedy), dans la réduction du temps d'attente global des véhicules dans des scénarios de congestion simulés.

2. Organisation du mémoire :

Ce mémoire est structuré en quatre chapitres distincts :

Chapitre 1 : Ce chapitre présente une vue d'ensemble des enjeux du trafic routier, incluant des définitions, des problématiques, des comparaisons entre la réglementation traditionnelle et moderne de la circulation, ainsi que des solutions avancées proposées par la recherche.

Chapitre 2 : Ce chapitre présente les notions d'agents et de systèmes multi-agents en déterminant leurs propriétés et caractéristiques. Nous exposons également les différentes plateformes de développement des SMA ainsi que leurs principaux domaines d'application. Ensuite, nous introduisons quelques notions sur la simulation de trafic, les différents types de simulation, ainsi que l'utilisation des systèmes multi-agents (SMA) pour simuler des phénomènes complexes de trafic.

Chapitre 3 : Ce chapitre concerne la modélisation de notre système de gestion de trafic routier basé sur les systèmes multi-agents. Nous expliquons en détail toutes les phases de la conception de notre système en utilisant la méthodologie VOYELLE et le langage de modélisation AUML.

Chapitre 4 : Ce chapitre détaille l'implémentation de notre système, en décrivant l'heuristique FIFO (First-In, First-Out), l'heuristique glouton et l'heuristique glouton avancée. Il inclut également une présentation des différents outils et langages de développement utilisés.

Chapitre 01 :

Enjeux du Trafic routier

1. Introduction :

Le trafic routier, central dans nos sociétés modernes, pose d'importants défis dépassant les simples déplacements. L'augmentation rapide des véhicules et de l'urbanisation complique la gestion des flux. Sur le plan économique, les embouteillages coûtent cher en productivité perdue. Ils contribuent aussi à la pollution et aux émissions de gaz à effet de serre. Les problèmes sociaux comme le stress, les conflits et les inégalités d'accès sont manifestes, tout comme la sécurité routière reste préoccupante avec les risques d'accidents. La gestion du trafic est donc complexe, nécessitant des solutions intégrées pour des déplacements efficaces, durables et sûrs dans nos communautés en évolution.

Dans ce chapitre, nous allons donner un aperçu sur les concepts d'encombrement dans le trafic routier ainsi que des travaux connexes sur la gestion du trafic routier.

2. Les enjeux du Trafic routier :

2.1 Définitions :

Dans la littérature, il existe plusieurs définitions du concept de l'enjeux du trafic routier. Dans la suite de cette section nous allons présenter quelques définitions :

Définition 1 : Le trafic routier désigne la circulation routière avec une notion sous-jacente de volume de cette circulation **[6]**.

Il est important de connaître les données de trafic pour la sécurité routière, car elles permettent le calcul de certains indicateurs, tels que le taux d'accidents par type de route et/ou par catégorie de véhicule, l'indice d'accidentalité locale, et l'exposition aux risques de certaines catégories d'usagers **[6]**.

On distingue trois types de trafic routier **:**

Trafic de transit : concerne les véhicules qui traversent une zone sans y avoir leur point de départ ou leur destination finale **[7].**

Trafic d'échange : concerne les véhicules dont le point de départ est à l'intérieur de la zone étudiée et la destination à l'extérieur, ou vice versa **[7].**

Trafic local (ou interne) : concerne les véhicules qui se déplacent uniquement à l'intérieur de la zone étudiée **[7].**

Définition 2 : La gestion du trafic routier est un enjeu important pour réduire la congestion, le bruit, et la pollution liée au transport routier de marchandises en ville **[8]**.

Les pouvoirs publics encouragent des transports alternatifs et des systèmes innovants de gestion du trafic urbain pour répondre aux enjeux de la mobilité d'aujourd'hui et anticiper ceux de la future **[8]**.

2.2. Problématique du trafic automobile :

2.2.1. Comprendre les difficultés du trafic :

La compréhension des difficultés du trafic est un enjeu important pour y répondre efficacement. Selon les experts, la congestion urbaine se produit lorsque la demande de déplacements dépasse la capacité de la route. L'augmentation de la capacité d'une route sans modifier son coût d'utilisation peut attirer une nouvelle demande de déplacements, créant ainsi une demande induite par l'augmentation de l'offre viaire. Les problèmes de trafic peuvent également être accentués par une mauvaise hiérarchie des voies urbaines et un afflux de voitures en période de pointe. Les mesures d'apaisement de la circulation, telles que les rues complètes, peuvent aider à réduire les vitesses de circulation et à favoriser la convivialité de l'espace public. Les prévisions de difficultés de circulation sont publiées chaque année pour les autoroutes et les routes nationales ayant deux sens de circulation **[9, 10, 11]**.

2.2.2. Contextualisation des enjeux dans le trafic routier :

Les enjeux du trafic routier sont multiples et incluent des préoccupations sociales, environnementales et sanitaires. Le trafic routier est responsable d'une grande partie des particules fines qui polluent l'air extérieur, ce qui a un impact sur la santé publique **[12, 13]**. Sur le plan social, la croissance urbaine entraîne un besoin croissant de mobilité et une dépendance à l'automobile, ce qui nécessite une modification des comportements et une transition vers des modes de transport plus durables **[12]**. En outre, le secteur du transport routier est le principal émetteur de gaz à effet de serre, contribuant ainsi de manière significative au réchauffement climatique **[12]**. Par conséquent, la réduction de l'impact environnemental du trafic routier est un enjeu majeur. Des mesures telles que la réduction de la vitesse, les aménagements urbains adaptés et la promotion de modes de transport alternatifs sont essentielles pour faire face à ces enjeux **[14]**.

Le tableau suivant compare les modes de circulation traditionnels et modernes, illustrant les différences entre les deux.

Caractéristiques	Circulation traditionnelle	Circulation moderne
Infrastructure	Routes étroites et sinueuses	Routes larges et droites
Vitesse	Limitée par les conditions de la route et la sécurité	Limitée par la réglementation et la sécurité
Véhicules	Véhicules lents et souvent encombrants	Véhicules rapides et plus efficaces
Règles de conduite	Informelles et basées sur la courtoisie	Strictes et réglementées
Sécurité	Dépend de la prudence des conducteurs	Dépend de la réglementation et de la technologie

Table 1.1 comparaison entre la réglementation de la circulation traditionnelle et moderne

2.3. Objectif de l'étude sur la gestion du trafic routier :

L'étude de la gestion du trafic routier influence le développement des entreprises et la qualité de vie des clients :

2.3.1. Pour les entreprises :

- ✓ Comprendre et résoudre les problèmes liés à l'encombrement des routes et des parkings, en particulier dans les zones urbaines.
- ✓ Les entreprises souhaitent optimiser l'utilisation de l'espace routier et des parkings pour améliorer l'efficacité des transports et la mobilité des usagers.
- ✓ Promouvoir la collaboration entre les entreprises, les autorités et les usagers.

2.3.2. Pour les clients :

- ✓ Trouver des solutions pour réduire les embouteillages et améliorer la fluidité du trafic routier
- ✓ Répondre aux attentes des clients en termes de temps de déplacement
- ✓ Réduire les coûts économiques et environnementaux associés aux embouteillages
- ✓ Mise en place de systèmes de transport intelligents.

3. Travaux de recherche sur la gestion du trafic routier :

Dans la littérature, de nombreux travaux explorent le domaine de la gestion du trafic routier. Certains chercheurs développent des systèmes réels de gestion du trafic en utilisant des capteurs pour surveiller le flux de véhicules, étudier les schémas de circulation et prendre des décisions d'intervention. En parallèle, d'autres chercheurs se concentrent sur la simulation du trafic routier, utilisant des algorithmes et des outils informatiques pour optimiser les flux avant de les appliquer dans la réalité. La section suivante présentera quelques-uns de ces travaux de recherche sur la gestion du trafic routier.

Le travail présenté dans **[40]**, se concentre sur l'amélioration des simulations de trafic multi-agent à grande échelle, essentielles pour la gestion territoriale moderne. Un simulateur de trafic générique, à la fois simple et représentatif des outils existants, est développé pour les expérimentations. Il inclut des modules permettant de simuler les déplacements en contexte microscopique et macroscopique, couvrant ainsi une large gamme de scénarios de simulation. Deux méthodes de distribution des simulations sont proposées : la première répartit les agents sur plusieurs cœurs de calcul, tandis que la seconde segmente l'environnement en différentes parties distribuées entre les hôtes. Ces méthodes sont appliquées et évaluées en fonction de leur efficacité dans différents contextes de simulation. Un algorithme d'équilibrage de charge dynamique est également développé pour optimiser les performances des simulations microscopiques, assurant une distribution efficace des charges pendant l'exécution. Les solutions proposées sont testées sur le réseau de Paris-Saclay, montrant une amélioration significative des performances. Les contributions de cette thèse sont généralisables et applicables à divers simulateurs existants, offrant des outils robustes pour la simulation de trafic à grande échelle et facilitant la planification et la gestion territoriale.

Les auteurs de **[1]** ont exploré l'amélioration des systèmes de transport urbain grâce aux nouvelles technologies de l'information, en mettant en avant l'utilisation des systèmes de transport intelligents pour réduire les embouteillages et améliorer la sécurité. Elle présente trois contributions majeures : d'abord, un modèle de graphe pour déployer des réseaux de capteurs sans fil dans plus de cinquante villes, révélant des caractéristiques structurales utiles pour l'évaluation de stratégies de routage. Ensuite, elle introduit les algorithmes TAPIOCA **[15]** et BASALTE **[1]** pour la gestion dynamique des feux de circulation, adaptés respectivement aux conditions de trafic normal et de saturation, dont les simulations montrent une efficacité supérieure aux systèmes traditionnels. Enfin, elle analyse la fiabilité des

communications sans fil, proposant un mécanisme d'interpolation pour compenser les pertes d'informations, validé par une co-simulation entre SUMO **[16]** (un simulateur transport) et OMNeT++ **[17]** (un simulateur réseau). Cette recherche démontre l'efficacité et la robustesse des systèmes de transport décentralisés et autonomes, offrant une meilleure tolérance aux pannes et une flexibilité accrue dans la gestion des infrastructures urbaines.

Les auteures de **[2]** ont mis l'accent sur la gestion intelligente du trafic urbain, essentielle pour les villes intelligentes. Birmingham City Council a ouvert l'accès à des ensembles de données en temps réel pour permettre des prévisions de trafic prédictives à code source ouvert. Pour répondre à ce défi, un système multi-agents (SMA) est proposé, intégrant des techniques de prévision et de classification. Les agents conçus ont pour tâches de prévoir les taux d'occupation des flux de trafic, des intersections routières et des parkings, de classifier les pannes et de contrôler et surveiller l'ensemble du processus. Les résultats expérimentaux démontrent que les prévisions basées sur les k-plus proches voisins et les arbres aléatoires sont hautement précises pour les données de trafic, tandis que les arbres de décision sont efficaces pour classifier les pannes. Ce système automatisé pourrait améliorer significativement la gestion du trafic à Birmingham et dans les Midlands de l'Ouest, offrant une solution proactive et réactive aux défis du trafic urbain.

Les auteurs de **[3]** ont proposés un système intelligent de contrôle et de déviation des feux de circulation appelé EITLCD (Efficient Intelligent Traffic Light Control and Deviation) basé sur un système multi-agent. Ce système utilise des capteurs et des algorithmes d'intelligence artificielle pour collecter des informations sur le trafic en temps réel et pour optimiser le contrôle des feux de circulation. Le système EITLCD est capable de dévier les véhicules avant qu'ils n'entrent dans des zones congestionnées, ce qui permet de réduire considérablement le temps de trajet et d'améliorer la fluidité du trafic. Les résultats de la simulation montrent que le système EITLCD est plus performant que les systèmes de gestion du trafic existants en termes de temps, de coût et de performance globale. Le système EITLCD est une solution prometteuse pour améliorer l'efficacité et la sécurité du trafic routier dans les villes intelligentes.

Les auteurs de **[4]** ont proposés une approche qui s'inspire du comportement des colonies de fourmis pour optimiser la circulation routière. Dans la nature, les fourmis déposent des phéromones sur leur chemin, ce qui permet aux autres fourmis de suivre le chemin le plus court vers la nourriture. Dans le cas de la gestion du trafic, les phéromones sont déposées

virtuellement sur les routes par un logiciel de qualité. Tout d'abord, le système collecte des données en temps réel sur le trafic, telles que la vitesse des véhicules, le nombre de véhicules et l'état des feux de signalisation. Puis, il calcul des phéromones sur la base des données collectées, le système calcule la quantité de phéromones à déposer sur chaque route. Les routes les moins encombrées reçoivent plus de phéromones, tandis que les routes les plus encombrées en reçoivent moins. Ensuite, il mise à jour des phéromones au fil du temps, les phéromones s'évaporent, ce qui signifie que leur quantité diminue. Cela permet de s'assurer que les routes les plus récemment empruntées par les véhicules ont le plus de phéromones, ce qui attire davantage de véhicules et optimise la circulation. Et finalement, le système dirige les véhicules qui sont équipés de capteurs spéciaux peuvent détecter la quantité de phéromones sur chaque route. Ils utilisent ces informations pour choisir la route la moins encombrée, ce qui permet de fluidifier le trafic et de réduire la congestion.

L'utilisation de systèmes de fourmis intelligentes permet à l'approche de s'adapter en temps réel aux conditions du trafic. Cela signifie que le système peut ajuster la quantité de phéromones déposée sur chaque route en fonction de la situation en temps réel, ce qui permet d'optimiser la circulation en permanence.

Les auteurs de **[5]** ont proposé un modèle de simulation du trafic urbain basé sur les automates cellulaires multi-agents (SMA) pour optimiser la coordination entre les véhicules et les feux de signalisation. Ce modèle combine les avantages des automates cellulaires, qui permettent de simuler efficacement les mouvements des véhicules, et des SMA, qui permettent de représenter les interactions complexes entre les agents du système. Le modèle prend en compte divers facteurs tels que le comportement des conducteurs, les caractéristiques des véhicules, l'état du trafic et les conditions environnementales. Il peut être utilisé pour simuler différents scénarios de trafic, y compris les intersections, les autoroutes et les zones urbaines denses. Les résultats de la simulation peuvent informer la conception et la gestion de systèmes de transport intelligents, contribuant ainsi à réduire la congestion, à améliorer l'efficacité du trafic et à minimiser l'impact environnemental des transports urbains.

4. Conclusion :

En conclusion, la gestion du trafic routier apparaît comme une problématique complexe et diversifiée. Les enjeux économiques, environnementaux, sociaux et de sécurité liée à la congestion routière nécessitent des solutions innovantes et une approche globale. La croissance rapide du nombre de véhicules et l'urbanisation posent des défis considérables,

impactant la productivité, l'environnement, la qualité de vie et la sécurité. Il est crucial de trouver des solutions efficaces, durables et sécuritaires pour répondre aux besoins évolutifs des communautés. Ce chapitre vise à explorer les concepts d'encombrement routier, en expliquant la problématique, les enjeux et les objectifs de la gestion du trafic routier, tout en proposant quelques travaux de recherche qui cherchent à améliorer la gestion du trafic routier en utilisant différentes méthodes et techniques de l'informatiques.

Chapitre 02 :
Système Multi-Agent

1. Introduction :

Les Systèmes Multi-Agents (SMA) émergent comme une branche essentielle de l'Intelligence Artificielle Distribuée (IAD). Fruit de la convergence entre l'intelligence artificielle et les systèmes distribués, les SMA ont vu le jour pour surmonter les limites de l'IA traditionnelle, offrant ainsi une solution aux problèmes répartis de manière inhérente. De nos jours, les SMA occupent une place prépondérante parmi les technologies dédiées au développement de systèmes complexes et distribués. Ce chapitre vise à présenter les concepts fondamentaux de la technologie des agents et des SMA, mettant en lumière leurs caractéristiques distinctives, tout en illustrant quelques exemples concrets de leurs domaines d'application diversifiés**.**

2. Notions d'agent

2.1. Définition d'un agent :

Il existe de nombreuses définitions d'agent, y compris cette définition générale, selon laquelle un agent est une entité autonome capable d'agir dans un environnement pour atteindre des objectifs. Il peut être une personne, un logiciel ou un système informatique. Les agents sont capables de percevoir leur environnement, de prendre des décisions et d'agir en conséquence. Ils peuvent également interagir avec d'autres agents pour atteindre des objectifs communs. Les agents sont utilisés dans divers domaines tels que les télécommunications, la simulation d'écosystèmes et le commerce électronique **[18, 19]**.

2.2. Structure générale d'un agent : La structure générale d'un agent comprend plusieurs composants clés **[20, 21, 22]** :

- **Perception** : L'agent doit être capable de percevoir son environnement à travers des capteurs.
- **Raisonnement** : L'agent doit être capable de raisonner pour interpréter les perceptions, résoudre les problèmes et prendre des décisions.
- **Action** : L'agent doit être capable d'agir sur son environnement à travers des actionneurs.
- **Environnement** : L'agent évolue dans un environnement qui peut être statique ou dynamique.
- **Objectif** : L'agent doit avoir un objectif à atteindre.
- **Communication** : L'agent doit être capable de communiquer avec d'autres agents pour atteindre ses objectifs.

La structure d'un agent peut varier en fonction de son type, tel que les agents à réflexe simple, les agents basés sur des modèles, les agents basés sur des buts, les agents basés sur l'utilité et les agents d'apprentissage.

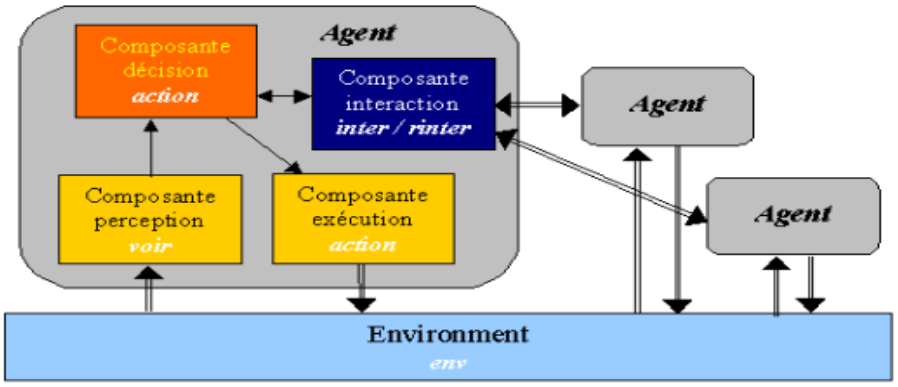

Figure 2.1 : Structure générale d'un agent qui interagit avec d'autres agents

2.3. Capacités d'agent :

Pour caractériser un agent intelligent, on peut identifier les capacités suivantes :

- **Autonomie** : Un agent intelligent est une entité autonome capable de percevoir son environnement grâce à des capteurs et d'agir sur celui-ci pour atteindre des objectifs **[23, 24]**.
- **Réactivité** : Les agents réactifs sont capables de réagir de manière rapide et appropriée à des changements dans leur environnement, sans nécessiter de modélisation interne complexe **[5]**.
- **Pro-activité** : Certains agents sont capables, sur leur propre initiative, de se fixer des buts pour atteindre leurs objectifs, ce qui est une forme de pro-activité **[25]**.
- **Apprentissage** : Les agents peuvent être dotés de capacités d'apprentissage, ce qui leur permet d'améliorer leurs performances au fil du temps en acquérant de nouvelles connaissances et en s'adaptant à des situations changeantes **[26, 24]**.
- **Adaptabilité** : Les agents peuvent être capables de s'adapter à des environnements changeants ou à des tâches variées, en ajustant leurs comportements ou leurs stratégies en fonction des circonstances **[26]**.
- **Aptitude sociale et coopération** : Certains agents sont capables d'interagir avec d'autres agents de façon coopérative ou compétitive pour atteindre des objectifs, ce qui implique une certaine aptitude sociale **[25]**.
- **Communication** : Les agents peuvent être amenés à communiquer avec d'autres agents pour échanger des informations, coordonner des actions, ou négocier dans des environnements multi-agents **[25]**.

- **Personnalisation** : Certains agents peuvent être personnalisés pour s'adapter aux préférences ou aux besoins spécifiques de leurs utilisateurs ou de leur environnement **[24]**.

Ces capacités permettent aux agents intelligents d'interagir de manière autonome et efficace avec leur environnement, et de réaliser des tâches complexes dans divers domaines d'application.

2.4. Types des agents et leurs applications :

Les agents se déclinent en plusieurs types, chacun avec des caractéristiques et des applications spécifiques. Voici des principaux types d'agents :

2.4.1. Agent réactif : Ce type d'agent réagit de manière rapide et appropriée à des changements dans son environnement, sans nécessiter de modélisation interne complexe. Il est utilisé dans des applications telles que les systèmes de contrôle de processus industriels ou les jeux vidéo **[26, 27]**.

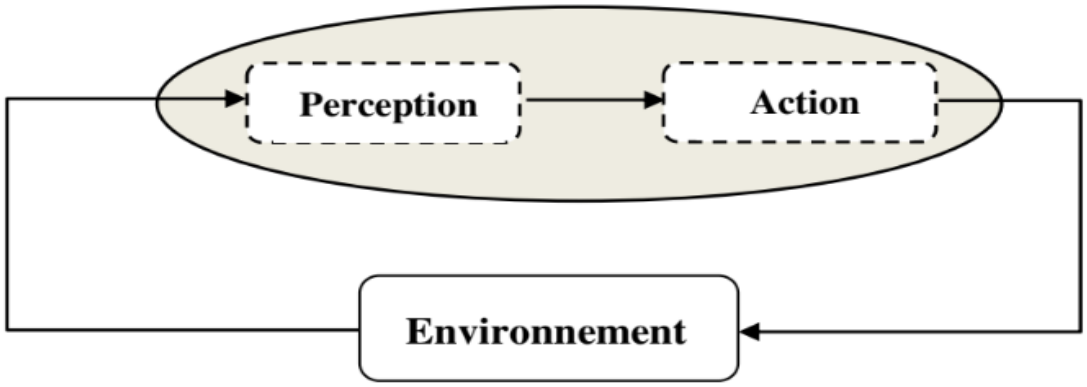

Figure 2.2 : Agent réactif

2.4.2. Agent cognitif : Ce type d'agent dispose d'une représentation interne de l'état de son environnement et utilise des modèles pour prendre des décisions. Il est capable de raisonner en s'appuyant sur une base de connaissances ainsi que sur des mécanismes sophistiqués d'interaction entre agents. Il est utilisé dans des applications telles que la planification de trajets pour les véhicules autonomes ou la prédiction de la demande pour les systèmes de gestion de la chaîne d'approvisionnement **[28, 29]**.

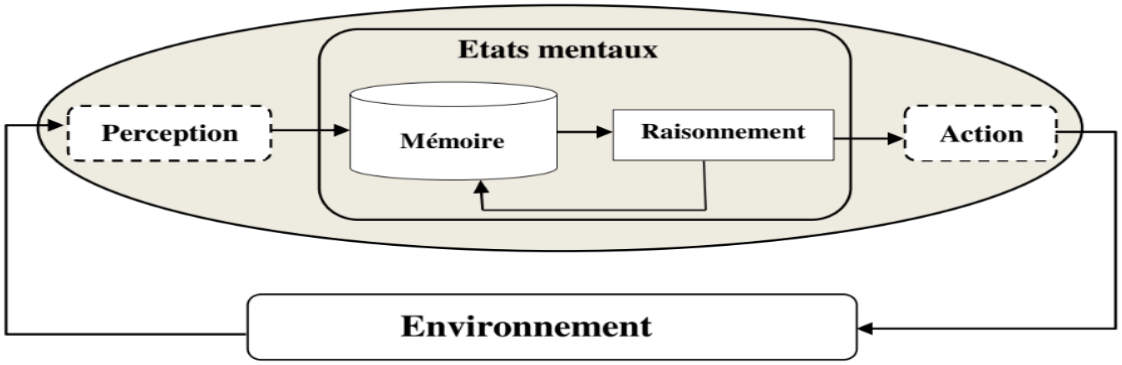

Figure 2.3: Agent cognitif

2.4.3. Agent hybride : Ce type d'agent combine à la fois des comportements réactifs et des comportements cognitifs. Il est utilisé dans des applications telles que la surveillance de la qualité de l'air ou la gestion de l'énergie **[27]**.

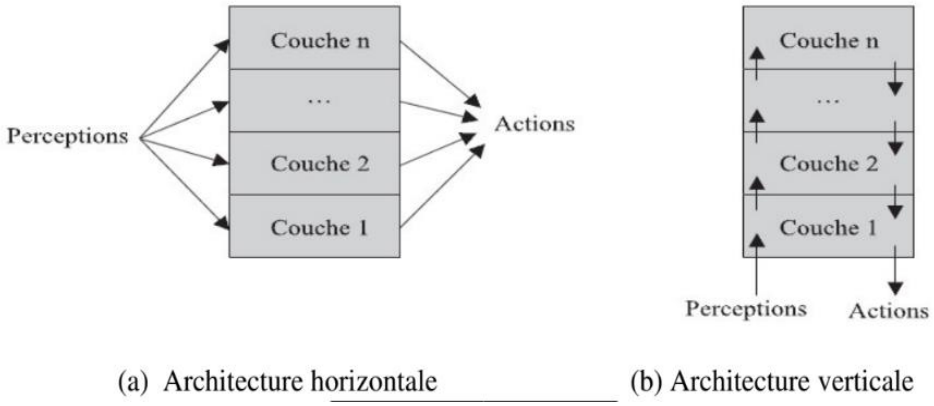

Figure 2.4 : Agent hybride

3. Système multi-agent :

3.1. Définition :

Un système multi-agent (SMA) est un système composé d'un ensemble d'agents autonomes, situés dans un environnement donné et interagissant selon certaines relations **.**Les agents peuvent être des processus, des robots, des êtres humains, etc. Les agents peuvent interagir pour gérer, communiquer, se coordonner, coopérer, négocier, etc. **[30,31]**

Les systèmes multi-agents sont utilisés dans divers domaines tels que la modélisation de sociétés, les sciences humaines, l'intelligence artificielle, etc. **[31, 32]**

Les SMA peuvent résoudre des problèmes difficiles ou impossibles à résoudre pour les systèmes traditionnels **[31]** et peuvent être vus comme la rencontre de divers domaines tels que l'intelligence artificielle, la théorie des systèmes, la théorie des jeux, la psychologie

cognitive, etc. **[30]**. Les SMA peuvent être utilisés pour modéliser des phénomènes complexes tels que la circulation routière, la gestion de l'énergie, la gestion de l'eau, etc. **[32]**

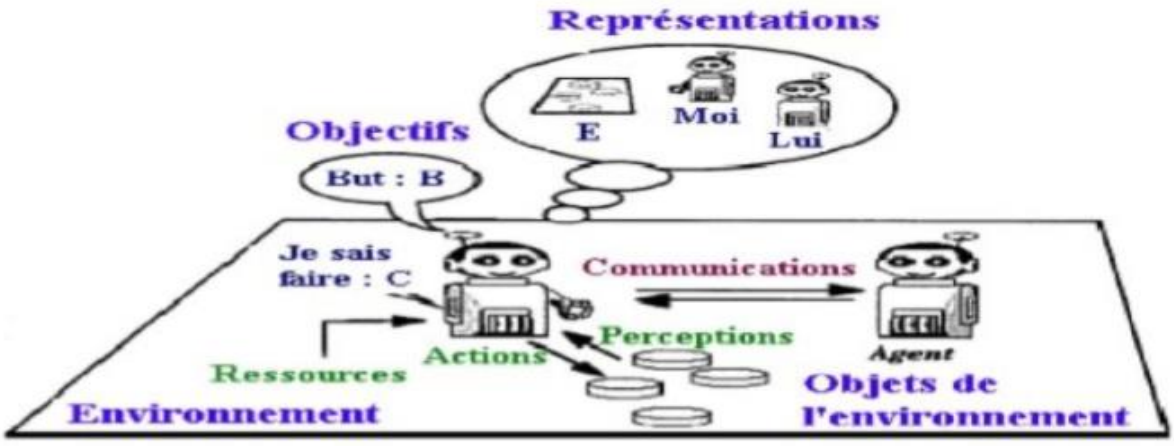

Figure 2.5 : Schéma d'un système multi agent [33]

3.2. Caractéristiques : Les caractéristiques d'un système multi-agent (SMA) incluent **[30, 25] :**

- La présence de plusieurs agents autonomes qui interagissent entre eux dans un environnement commun.
- Les agents peuvent être des processus, des robots, des êtres humains, etc.
- Les agents peuvent avoir des compétences simples ou avancées.
- Les interactions entre les agents peuvent être de nature contradictoire ou complémentaire.
- L'architecture peut être réactive ou délibérative.
- Les systèmes multi-agents peuvent être ouverts (les agents entrent et sortent librement) ou fermés (l'ensemble d'agents reste le même).
- Les agents peuvent être homogènes (construits sur le même modèle) ou hétérogènes (de modèles différents).
- Les agents sont autonomes et possèdent un point de vue partiel.
- Les données sont décentralisées et le calcul est asynchrone.
- Il n'y a pas de contrôle global du SMA

3.3. Les types des SMA:

Les systèmes multi-agents (SMA) peuvent être classés en différentes catégories en fonction de divers critères. Voici quelques-unes de ces classifications : **[30, 34]**

3.3.1. Ouvert vs Fermé :

- **Ouvert** : Dans un SMA ouvert, les agents peuvent entrer et sortir librement du système. Par exemple, l'utilisation d'un SMA pour développer une application de commerce électronique.

- **Fermé** : Dans un SMA fermé, l'ensemble d'agents reste constant, c'est-à-dire aucun agent n'entre ou ne sort du système.

3.3.2. Homogène vs Hétérogène :

-**Homogène** : Tous les agents dans un SMA homogène sont construits sur le même modèle.

-**Hétérogène** : Dans un SMA hétérogène, les agents sont de modèles différents, c'est-à-dire de granularités différentes.

Ces classifications permettent de mieux comprendre la nature et la complexité des SMA, et sont utiles pour concevoir et analyser ces systèmes.

3.4. Interaction entre agents :

3.4.1. Définition :

L'interaction entre les agents dans un système multi-agents se réfère à l'échange dynamique d'informations, de messages ou d'actions entre les entités autonomes. Ces interactions sont essentielles pour la coopération, la coordination et la résolution de problèmes collaboratifs au sein du système. Elles peuvent inclure la communication directe, la négociation, la coordination des actions, ou d'autres formes de coopération, permettant aux agents de travailler ensemble de manière efficace pour atteindre des objectifs communs ou individuels **[35]**.

3.4.2. Types d'interactions :

On distingue principalement quatre types d'interactions que les agents peuvent utiliser : la coordination, la coopération, la communication et la négociation **[35]**.

3.4.2.1. Coordination : La coordination implique l'ajustement des activités des agents pour atteindre un objectif partagé. Cela peut inclure la synchronisation des actions ou la répartition des tâches pour éviter les conflits et optimiser les résultats.

3.4.2.2. Coopération : La coopération se réfère au travail commun des agents pour atteindre un objectif partagé. Cela implique souvent le partage des ressources, l'assistance mutuelle et la prise de décisions qui bénéficient à l'ensemble du groupe.

3.4.2.3. Communication : La communication est l'échange d'informations entre les agents. Cela peut se faire par le biais de messages, de signaux ou d'autres formes de transmission d'informations. Elle est essentielle pour la coordination, la coopération et la prise de décision.

3.4.2.4. Négociation : La négociation implique la recherche d'un accord mutuellement bénéfique entre les agents. Cela peut inclure la discussion des objectifs, la résolution des conflits et la prise de décisions basée sur des compromis pour atteindre des résultats optimaux.

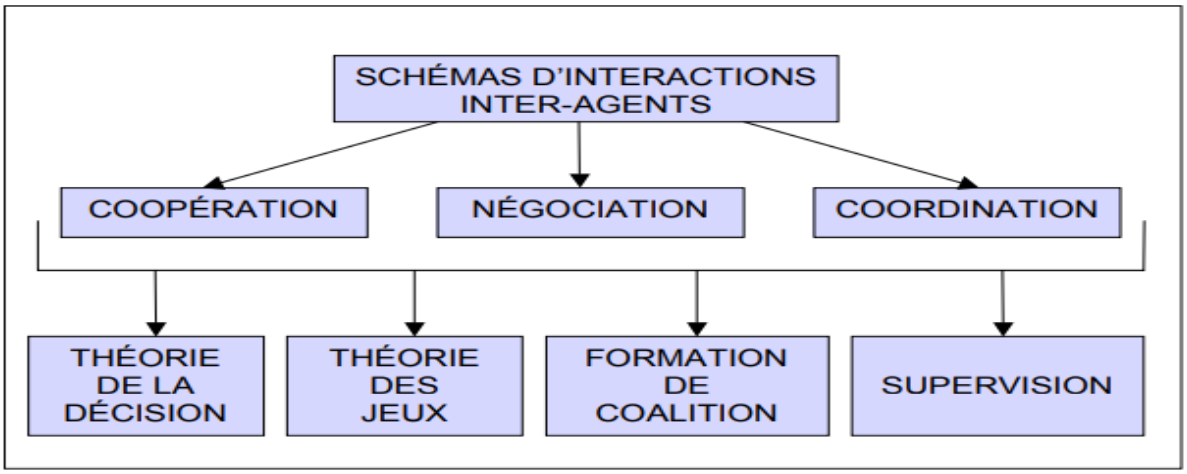

Figure 2.6: interactions inter-agents [36]

3.5. Plateformes SMA : Les environnements de développements des SMA, les plus connus sont **[37]** :

- **Jade** (Java Agent Développent Framework) : Jade **[38]** est un Framework de développement de systèmes multi agents, open-source et basé sur le langage Java. Il offre en particulier un support avancé de la norme FIPA-ACL, ainsi que des outils de validation syntaxique des messages entre agents basé sur les ontologies.
- **KQML** (Knowledge Query and Manipulation Language) : KQML **[39]** est un langage de communication entre agents ACL (Agent communication Language).

3.6. Domaines d'application des SMA : Les systèmes multi-agents (SMA) ont une large gamme de domaines d'application, notamment **[37]** :

- **Robotique :** Les SMA sont utilisés dans la robotique pour coordonner les mouvements et les actions de plusieurs robots autonomes. Chaque robot peut être considéré comme un agent capable de prendre des décisions en fonction de son environnement et de collaborer avec d'autres robots pour accomplir des tâches complexes.
- **Simulation:** Les SMA sont largement utilisés dans la simulation pour modéliser des systèmes complexes où de multiples entités interagissent. Par exemple, dans la simulation de trafic, chaque véhicule peut être modélisé comme un agent prenant des décisions basées sur son itinéraire, les conditions routières, etc. Les SMA permettent de reproduire des scénarios réalistes et de comprendre le comportement global du système.
- **Commerce électronique :** Dans le domaine du commerce électronique, les SMA sont utilisés pour la recommandation de produits. Les agents peuvent être configurés pour analyser le comportement d'achat des utilisateurs et recommander des produits en fonction de leurs préférences. Les interactions entre les agents contribuent à améliorer la précision des recommandations.
- **Surveillance** : Les SMA sont déployés dans des systèmes de surveillance pour la gestion et l'analyse des informations provenant de différentes sources. Les agents peuvent être programmés pour détecter des modèles anormaux, coordonner la surveillance de zones spécifiques, et collaborer pour traiter des situations de sécurité complexes.
- **Gestion du trafic :** Les SMA sont utilisés pour optimiser la gestion du trafic dans les systèmes de transport. Chaque véhicule peut être considéré comme un agent capable de prendre des décisions autonomes pour minimiser les congestions, améliorer le flux de circulation, et réagir aux changements dynamiques dans l'environnement routier.
- **Santé :** En santé, les SMA sont utilisés pour la gestion des dossiers médicaux électroniques, la planification des traitements et la coordination des soins. Les agents peuvent représenter des dispositifs médicaux, des professionnels de la santé et même les patients. Ils peuvent collaborer pour optimiser les horaires des rendez-vous, partager des informations médicales pertinentes de manière sécurisée, et contribuer à la prise de décision médicale.

4. Simulation de trafic routier :

La simulation de trafic routier est une méthode informatique complexe qui utilise des modèles mathématiques et des algorithmes pour reproduire de manière réaliste le comportement des véhicules dans un environnement routier virtuel. Elle permet d'observer comment les véhicules se déplacent, où se forment les embouteillages, et comment organiser les routes de manière plus efficace pour éviter les problèmes de circulation. Elle est particulièrement utile pour les urbanistes et les ingénieurs, car elle les aide à planifier des villes plus efficaces et à améliorer la sécurité sur les routes **[40]**.

Par la suite, nous présenterons le concept de simulation informatique en abordant différents types de simulateurs, en mettant particulièrement l'accent sur la simulation multi-agent, qui constitue le sujet central de notre mémoire.

4.2 Type de simulateurs :

Les modèles de simulation se déclinent en trois catégories principales :

4.2.1 La simulation macroscopique : se concentre principalement sur les volumes agrégés de flux de trafic sans considérer les éléments individuels tels que les véhicules ou les usagers.

4.2.2 La simulation mésoscopique : utilise des agents individuels, mais leur comportement est influencé par des caractéristiques agrégées du trafic comme la densité et la vitesse moyenne. Les interactions directes entre les agents se produisent généralement aux intersections.

4.2.3 La simulation microscopique : modélise le comportement de chaque agent individuel dans son environnement immédiat, prenant en compte les distances et les variations de vitesse par rapport aux autres agents proches. Les mouvements sont continus dans le temps et l'espace, avec des décisions rapides sur la vitesse et la direction en étapes très courtes, souvent inférieures à une seconde. L'état global du trafic routier découle des actions individuelles des agents.

En note, une approche hybride combine des aspects de simulation mésoscopique et microscopique, souvent utilisée dans des sections spécifiques du réseau routier pour une représentation plus précise et détaillée **[40]**.

4.3 Les systèmes multi-agents au service de la simulation :

Les systèmes multi-agents (SMA) sont des architectures informatiques où plusieurs entités autonomes, appelées agents, interagissent pour atteindre des objectifs communs dans un environnement donné. Ces systèmes sont utilisés pour simuler des phénomènes complexes en modélisant les interactions dynamiques entre les agents et leur environnement. Par exemple, dans une simulation de trafic routier, chaque véhicule peut être représenté par un agent qui prend des décisions en temps réel en fonction de son environnement, des règles de circulation et des autres véhicules.

L'un des points forts des SMA est leur capacité à générer des comportements collectifs à partir des interactions individuelles des agents. Cela permet d'observer des phénomènes émergents tels que la formation de files d'attente, la coordination du trafic, ou encore des schémas de comportement sociaux complexes.

Les SMA sont utilisés dans de nombreux domaines, notamment la gestion de flottes de véhicules, la modélisation des interactions sociales, la simulation de systèmes distribués, la robotique, les jeux vidéo, la finance et même la biologie pour simuler des comportements d'organismes complexes.

En résumé, les systèmes multi-agents sont des outils puissants pour la simulation de phénomènes complexes grâce à leur capacité à modéliser des interactions dynamiques et générer des comportements collectifs à partir d'agents autonomes **[41]**.

4.4 Gestion des Files d'Attente dans les Simulateurs Informatiques

Dans les simulateurs informatiques, la gestion des files d'attente est cruciale pour reproduire de manière réaliste les conditions d'attente rencontrées dans divers systèmes, tels que les réseaux de transport, les centres de service, ou les systèmes de traitement de données.

4.4.1 Principaux Algorithmes

Parmi les algorithmes les plus couramment utilisés, on trouve le principe FIFO (First In, First Out) **[42]** où les entités sont traitées dans l'ordre d'arrivée, et le modèle de priorité, où certaines entités ont la priorité sur d'autres en fonction de critères prédéfinis. Des exemples d'algorithmes de priorité incluent le SJF (Shortest Job First) **[42]**, où les entités avec le plus petit temps de traitement sont servies en premier, et le LIFO (Last In, First Out), où la dernière entité arrivée est servie en premier.

4.4.2 Approches Avancées

D'autres algorithmes incluent les méthodes de répartition de charges telles que l'algorithme du tourniquet (Round Robin) **[43]** qui répartit équitablement les ressources entre les entités en attente, et les stratégies de gestion du temps d'attente, comme l'algorithme de gestion de file MFQ (Multi-Level Feedback Queue) **[44]** qui ajuste dynamiquement les priorités en fonction du temps passé en attente. L'algorithme de gestion de file MLFQ peut également être combiné avec des stratégies de rétrogradation pour éviter les situations de famine où certaines entités n'obtiennent jamais de traitement.

4.4.3 Modèles Sophistiqués

En outre, des approches plus avancées telles que les files d'attente à plusieurs niveaux ou les files d'attente à seuil peuvent être utilisées pour modéliser des systèmes complexes avec des niveaux de priorité différents ou des capacités de traitement variables. Par exemple, le modèle MLQ (Multi-Level Queue) **[45]** classe les entités en plusieurs niveaux de priorité, tandis que le modèle M/M/1/K introduit une capacité maximale (K) pour la file d'attente, simulant ainsi des situations où les ressources sont limitées.

En combinant judicieusement ces algorithmes, les simulateurs informatiques peuvent fournir des outils puissants pour l'analyse et l'optimisation des systèmes réels soumis à des files d'attente.

5. Conclusion :

En conclusion, ce chapitre nous a permis d'explorer de manière approfondie les concepts fondamentaux des agents et des systèmes multi-agents (SMA), ainsi que de nous familiariser avec quelques plateformes de développement et exemples concrets d'applications. Les SMA se présentent comme des ensembles d'agents organisés en sociétés, interagissant, communiquant et coopérant pour atteindre des objectifs spécifiques. À la lumière de leurs caractéristiques intrinsèques, il est manifeste que les agents et les SMA offrent une approche prometteuse pour le développement d'applications complexes, en particulier dans le contexte de la gestion d’encombrement dans le trafic routier. Leur capacité à prendre des décisions décentralisées, à s'adapter aux changements dynamiques et à collaborer de manière autonome fait des SMA une solution pertinente pour relever les défis complexes liés à la gestion du trafic, ouvrant ainsi la voie à des solutions innovantes et efficaces.

Chapitre 03 :
Analyse et conception

1. Introduction :

Après avoir abordé les concepts des systèmes multi-agents et des simulations dans les chapitres précédents, ce chapitre se concentre sur les phases d'analyse et de conception de notre système. La première section propose une description fonctionnelle détaillée du système à développer. Ensuite, nous introduisons la méthode voyelle et le langage de modélisation AUML (Agent Unified Modeling Language) que nous avons employés pour la modélisation. La troisième section identifie les agents et explore les diverses interactions entre eux. Enfin, la partie conception présente différents diagrammes décrivant notre système.

2. Description du système réalisé :

Dans le cadre de ce projet de fin d'études, notre objectif est de développer un système de gestion du trafic routier visant à réduire l'encombrement. Ce système s'appuie sur une approche multi-agents et utilise une simulation avec des algorithmes basés sur différentes heuristiques.

Dans cette approche, les utilisateurs du système, qu'ils soient conducteurs sur la route ou administrateurs responsables de la surveillance générale, seront représentés par des agents logiciels spécialement conçus. Ces agents auront la capacité d'accomplir diverses tâches de manière autonome.

Le but principal est de faciliter la circulation routière et de diminuer les embouteillages pour rendre la conduite plus agréable.

3. Choix méthodologique :

3.1. AUML :

3.1.1. Définition :

AUML (Agent Unifiez Modeling Language) **[46,47]** est un langage de modélisation graphique qui a été standardisé par la comité technique de modélisation FIPA (Foundation for Intelligent Physical Agents). Il a été proposé comme une extension du langage de modélisation unifié (UML) pour la modélisation des systèmes multi-agents. Il utilise les caractéristiques de décomposition, d'abstraction et d'organisation pour réduire la complexité

de développement de logiciel. AUML décompose le système en petites parties d'objets, de modèles, de cas d'utilisation ou de classes, et d'actions opérationnelles.

La partie principale d'AUML est les mécanismes de modélisation des protocoles d'interaction des systèmes multi-agents. Cela est réalisé par l'introduction d'une nouvelle classe de diagramme à UML : 'diagramme de Protocole d'interactions'. Ces diagrammes étendent les diagrammes de séquence en incluant : 'les rôles d'agents', 'comportements d'agents' , 'interactions d'agents', 'protocole Template', etc..**[48]**

3.1.2 Diagrammes AUML :

Comme précisé dans la définition, AUML étend UML pour intégrer des concepts liés aux agents qui ne sont pas présents dans UML. Il hérite des différentes représentations proposées par UML, tout en introduisant des extensions pour modéliser les concepts de la technologie des agents. La liste des principaux diagrammes d'AUML comprend les diagrammes usuelles d'UML plus les diagrammes spécialement conçus pour AUML **[49]**. Dans la suite nous présentons la liste des diagrammes d'AUML en expliquant les deux diagrammes : Le diagramme de classes d'agent et le diagramme de Protocole d'interactions.

a. **Diagramme de Cas d'Utilisation :** Représente les fonctionnalités du système du point de vue des utilisateurs.
b. **Diagramme de Classes :** Montre les classes du système, leurs attributs, méthodes et les relations entre elles.
c. **Diagramme d'Objets :** Illustre des instances spécifiques de classes et les relations entre ces instances.
d. **Diagramme de Séquence :** Modélise la séquence d'interactions entre les objets au fil du temps.
e. **Diagramme de Collaboration :** Met en évidence les interactions entre les objets et leurs relations.
f. **Diagramme d'État :** Représente les différents états d'un objet et les transitions entre ces états.
g. **Diagramme d'Activités :** Montre le flux de contrôle ou de travail dans un processus.
h. **Diagramme de Déploiement** : Représente la configuration matérielle du système et la manière dont les composants logiciels y sont déployés.
i. **Diagramme de Composants :** Montre les composants logiciels et leurs dépendances.

j. **Diagramme de Package :** Organise les éléments du modèle en paquets et montre les dépendances entre les paquets.

k. **Diagramme de classes d'agent :** Une classe d'agent représente un agent ou un groupe d'agents capable de jouer un rôle ou d'adopter un comportement spécifique. Elle inclut une description de la classe d'agent et des rôles, une description de l'état interne, des actions, des méthodes et services fournis, ainsi que des messages échangés (voir Figure 3.1).

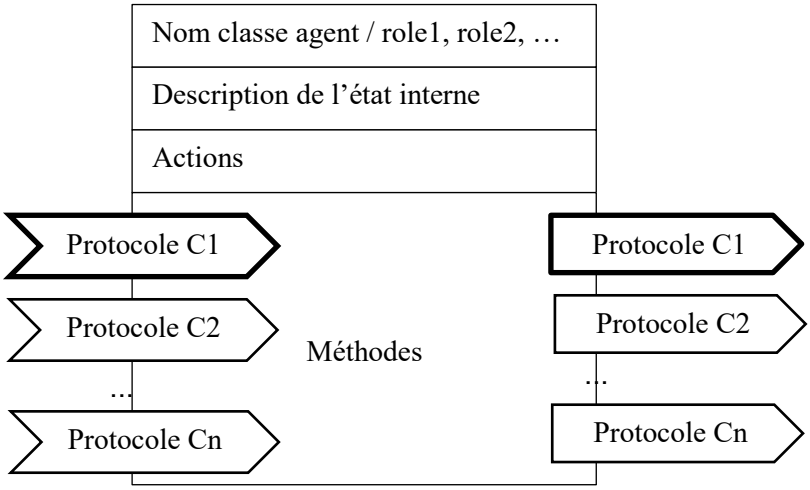

Figure 3.1 : Diagramme de classe d'agent

- **Nom de la classe d'agent/Rôles** : Un agent appartenant à une classe donnée peut assumer plusieurs rôles (par exemple, un détaillant peut être à la fois acheteur et vendeur).
- **Description des états** : Cette section définit les variables d'instance qui reflètent l'état actuel de l'agent.
- **Actions (Plans)** : Deux types d'actions peuvent être spécifiés :
  - Actions proactives : Déclenchées par l'agent lui-même lorsque certaines pré conditions sont vérifiées.
  - Actions réactives : Déclenchées en réponse à un message reçu d'un autre agent.

  En d'autres termes, les actions sont les plans disponibles à un agent.
- **Méthodes** : Elles sont définies selon les conventions d'UML et peuvent être accompagnées de pré conditions, de post conditions ou d'invariants.
- **Envoi et réception de messages :** Cette partie décrit les messages émis et reçus par l'agent, en spécifiant les protocoles associés.

- **Automate :** Il représente les changements d'état induits par les échanges de messages.

l. **Diagramme de Protocole d'interactions :** Un protocole d'interaction entre agents (AIP : Agent Interaction Protocol) comprend un schéma de communication précis. Ce schéma définit une séquence autorisée de messages échangés entre des agents ayant divers rôles, tout en imposant des contraintes sur le contenu de ces messages. De plus, le protocole établit une sémantique alignée avec les actes de communication au sein du schéma, garantissant ainsi une cohérence dans les échanges entre les agents.

AUML (Agent UML) permet de représenter la communication simultanée des actes envoyés de l'expéditeur au destinataire. Dans la suite, nous introduirons quelques actes de communication entre les agents :

- Les actes de communication simultanée ou concurrentes de CA-1 à CA-n sont envoyés en parallèle (voir Figure 3.2 (a)).

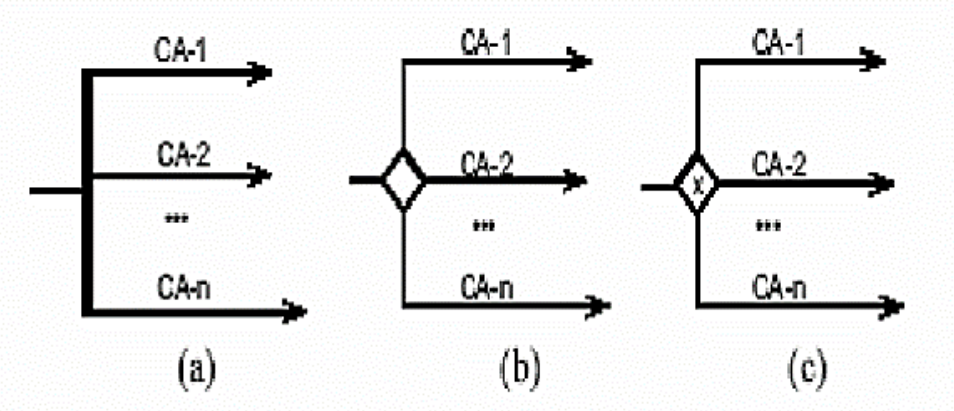

Figure 3.2 différents types d'interaction entre les agents

- Une sélection de zéro ou plusieurs actes est envoyée simultanément, incluant la possibilité qu'un seul acte CA soit envoyé parmi eux (voir Figure 3.2 (b)). Cela signifie que l'envoi peut inclure aucun acte, un seul acte, ou plusieurs actes à la fois.
- Choix exclusif : un seul des actes de communication est envoyé à la fois (voir Figure 3.2 (c)).

3.2. Méthodologie Voyelle : La méthodologie Voyelle est une approche de décomposition des systèmes en quatre dimensions ou lettres : Agent, Environnement, Interaction et Organisation. Cette méthodologie s'avère précieuse pour la compréhension et la conception de systèmes basés sur des agents **[50]**. Les quatre dimensions sont définies comme suit :

1. **Agent :** Cette dimension se concentre sur la modélisation des agents, qui sont des entités autonomes capables de prendre des décisions et d'interagir avec leur environnement.
2. **Environnement :** Il s'agit de la dimension qui concerne la modélisation de l'environnement dans lequel les agents opèrent. Cela peut inclure des facteurs externes, des ressources et d'autres entités.
3. **Interaction :** Cette dimension vise à représenter les interactions entre les agents et avec leur environnement. Cela peut inclure les échanges d'informations, les transactions ou d'autres formes de communication.
4. **Organisation :** La dimension Organisation se concentre sur la structuration des agents et de leurs interactions au sein d'une organisation plus large. Cela peut inclure des hiérarchies, des règles, des rôles, etc.

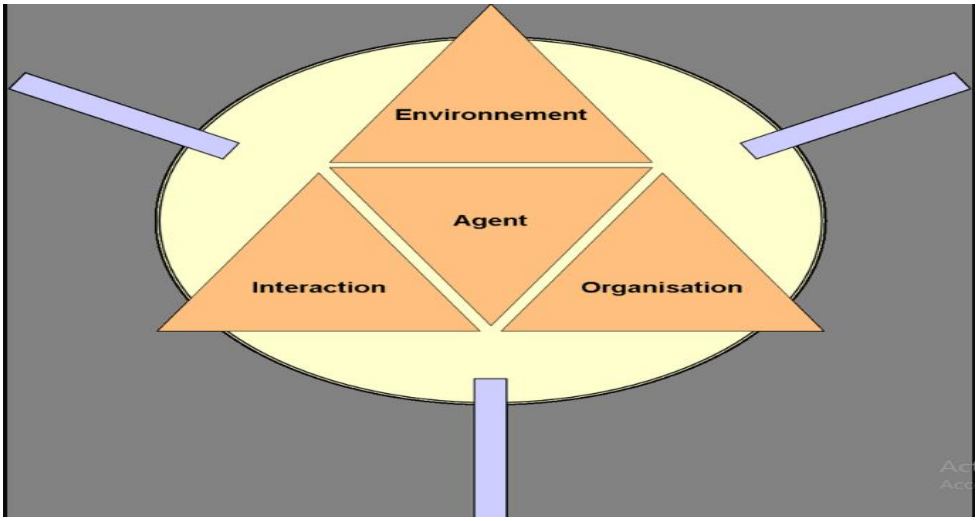

Figure 3.3 : Représentation de méthode voyelle

Le processus de développement avec la méthode Voyelle (ou AE/O) comprend trois étapes essentielles :

Analyse : Durant cette étape, l'objectif est d'identifier les cinq éléments clés, à savoir l'Agent, l'Environnement, les Interactions, l'Organisation (O), et les Utilisateurs (U). **Conception :** Il s'agit de choisir les modèles opérationnels des composantes. **Implémentation :** Cette étape consiste en l'instanciation des modèles en utilisant des plateformes et des langages choisis.

4. Analyse :

4.1. Identification des utilisateurs :

Dans notre système, nous avons identifié deux types d'utilisateurs ; le conducteur et le système. Chaque utilisateur joue un rôle spécifique.

- **Conducteur** Il s'agit des personnes qui conduisent les véhicules. Parmi les actions principales effectuées par les conducteurs au sein du système, on peut citer :
  - L'envoi de la position et la direction.
  - La conduite.
  - L'envoie des requêtes à l'administrateur pour avoir des informations sur la route.
  - La consultation des notifications qu'ils reçoivent de l'agent administrateur.
- **Administrateur :** C'est le responsable de la gestion du trafic routier. Les principales tâches de l'administrateur dans le système incluent :
  - Gérer le trafic routier.
  - Notifier et donner des ordres aux conducteurs.
  - Répondre aux requêtes des conducteurs.

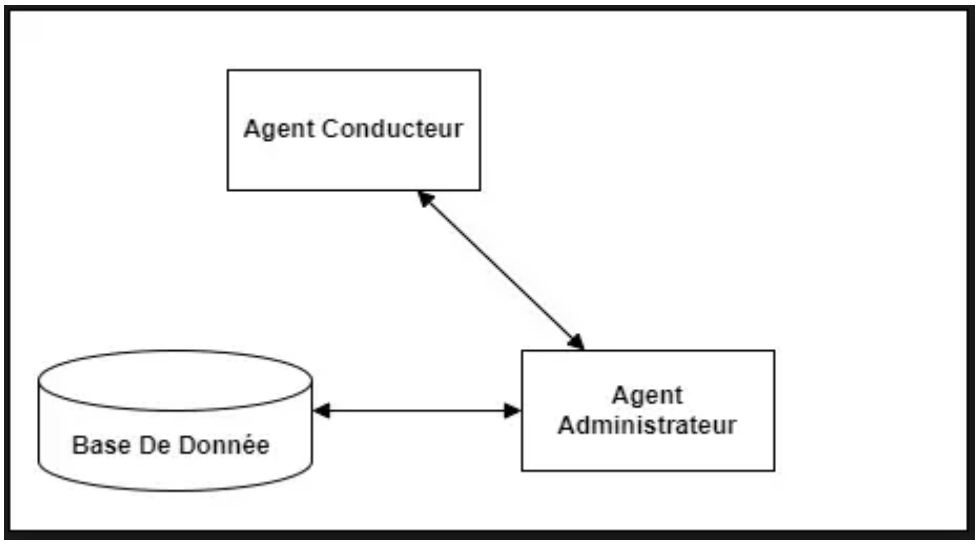

Figure 3.4. Architecture générale du système

4.2. Identification des agents :

La **figure 3.4** illustre l'architecture générale de notre système. Celui-ci est constitué de plusieurs agents, chacun remplissant un rôle bien défini. Dans la suite de cette section, nous présenterons les différents agents de notre système ainsi que les rôles qu'ils jouent.

- **Agent conducteur :** Il représente virtuellement un conducteur dans le système de gestion du trafic routier. Cet agent est capable d'exécuter diverses tâches, comme décrit dans la section précédente. Sa communication avec les autres agents du système garantit une conduite sûre et efficace. Il reçoit des informations de l'agent administrateur.
- **Agent Administrateur :** Responsable de la gestion du trafic routier, cet agent est également chargé de créer et de gérer d'autres entités dans le système comme les agents conducteur.

4.3. Identification des interactions :

Après avoir identifié les différents agents composant notre système, nous entamons cette étape en identifiant les interactions entre ces agents. Dans notre système, une interaction spécifique se distingue : celle entre l'agent administrateur et l'agent conducteur :

- **Interaction Agent Conducteur / Agent Administrateur :** chaque agent conducteur envoie continuellement sa position et sa direction à l'agent administrateur. Ce dernier gère le trafic routier en donnant des ordres de stop ou de passage aux agents conducteurs se trouvant à une intersection. L'agent administrateur fournit également l'état de la route aux agents conducteurs sur demande. Ces informations sont ensuite utilisées par les agents conducteurs pour naviguer de manière autonome.

4.4. L'environnement : Dans le contexte des Systèmes Multi-Agents (SMAs), les agents fonctionnent au sein d'un environnement partagé avec d'autres agents et entités, physiques ou logiques, avec lesquels ils interagissent. Ce cadre implique que les agents sont des entités autonomes capables de percevoir, raisonner, décider et agir de manière indépendante pour influencer leur environnement tout en collaborant avec d'autres agents. Le tableau ci-dessous présente l'environnement spécifique de chaque agent de notre système :

Agent	Environnement
Agent Conducteur	Agent Administrateur
Agent Administrateur	Agent Conducteur

Table 3.1 : L'environnement de chaque agent de notre système.

4.5. L'organisation : L'organisation représente la manière dont les agents interagissent et sont structurés pour collaborer ou rivaliser en vue d'atteindre des objectifs définis au sein de leur environnement commun. La structure organisationnelle de notre système est la suivante :

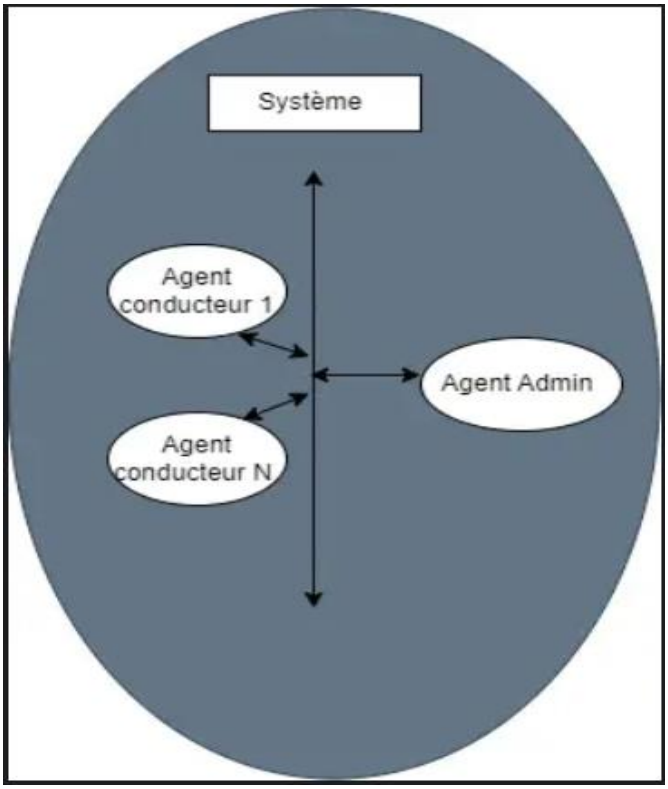

Figure 3.5. Organisation du système.

5. Conception :

L'approche VOYELLE ne nécessite pas de notation spécifique pour la modélisation de systèmes. Nous avons donc décidé d'utiliser le langage AUML pour ce faire. Notre conception se déroulera comme suit :

- Identification et description des cas d'utilisation, permettant de préciser les interactions des utilisateurs avec notre système.

- Modélisation des interactions entre agents par l'élaboration de diagrammes protocolaires d'interaction.
- Élaboration de diagrammes de classes.

5.1. Identification et description des cas d'utilisations :

5.1.1. Le diagramme de cas d'utilisation :

La figure 3.6 illustre le diagramme de cas d'utilisation global de notre système. Dans ce diagramme, nous avons un seul acteur principal qui est le conducteur.

Le conducteur envoie continuellement sa position et sa direction à l'administrateur, qui représente le système qui gère le système de trafic routier. Il demande également à l'administrateur des informations sur l'état des routes pour naviguer de manière autonome et éviter les accidents.

L'administrateur donne des ordres de passage ou d'arrêt aux conducteurs se trouvant aux intersections et répond aussi aux requêtes des conducteurs.

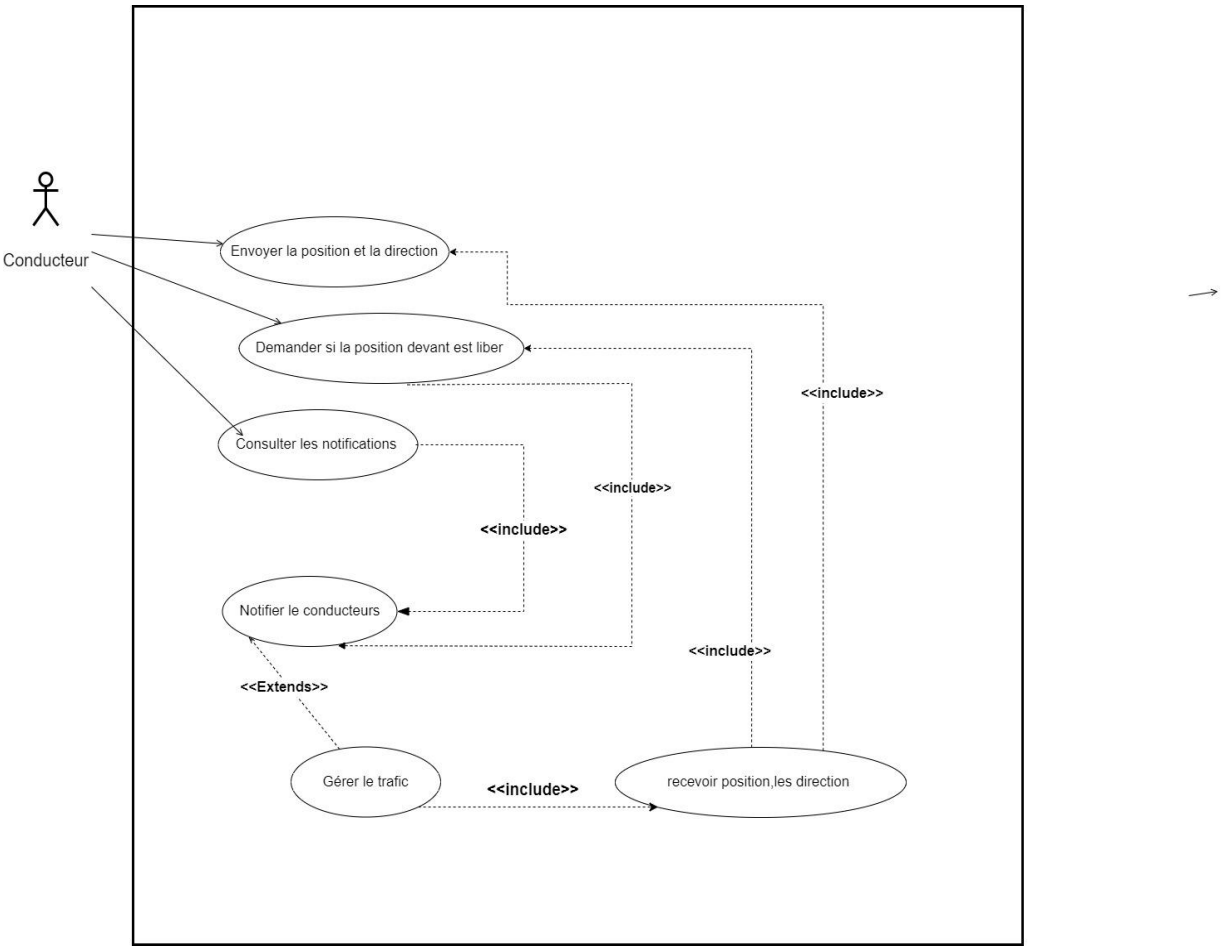

Figure 3.6. Le diagramme de cas d'utilisation

5.1.2. Description des cas d'utilisation par les diagrammes de séquence système :

Dans la prochaine section, nous présenterons des diagrammes de séquences système qui illustrent le déroulement des processus dans notre système.

a. Diagramme de séquence « Envoyer la position et la direction » :

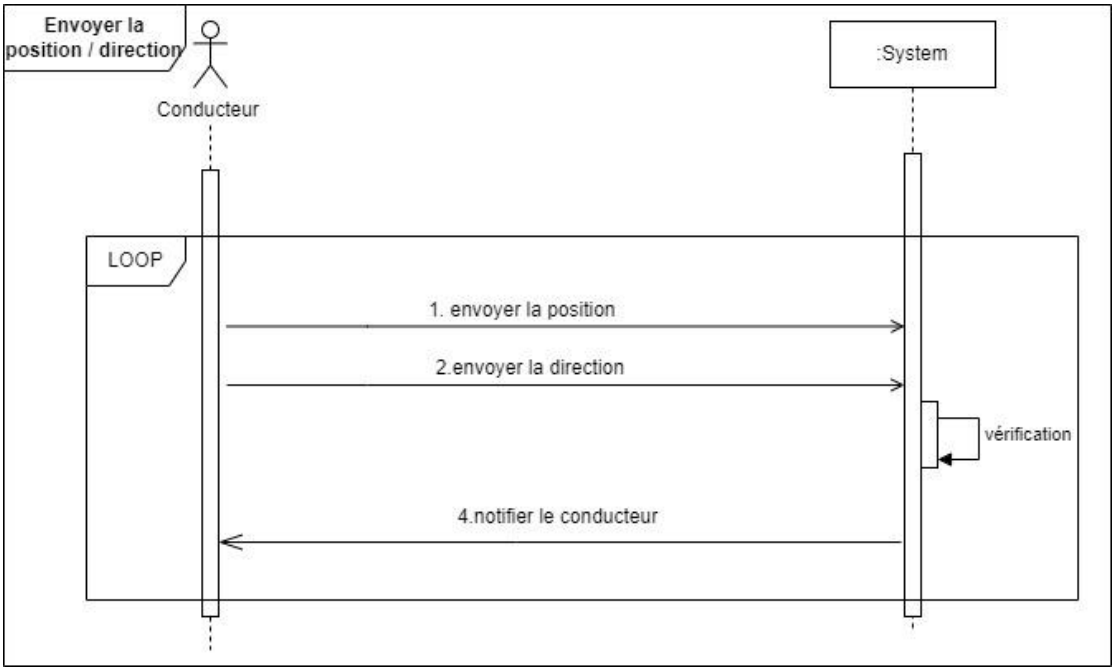

Figure 3.7. Diagramme de séquence « Envoyer la position et la direction »

b. Diagramme de séquence « Demander si la position devant est libre » :

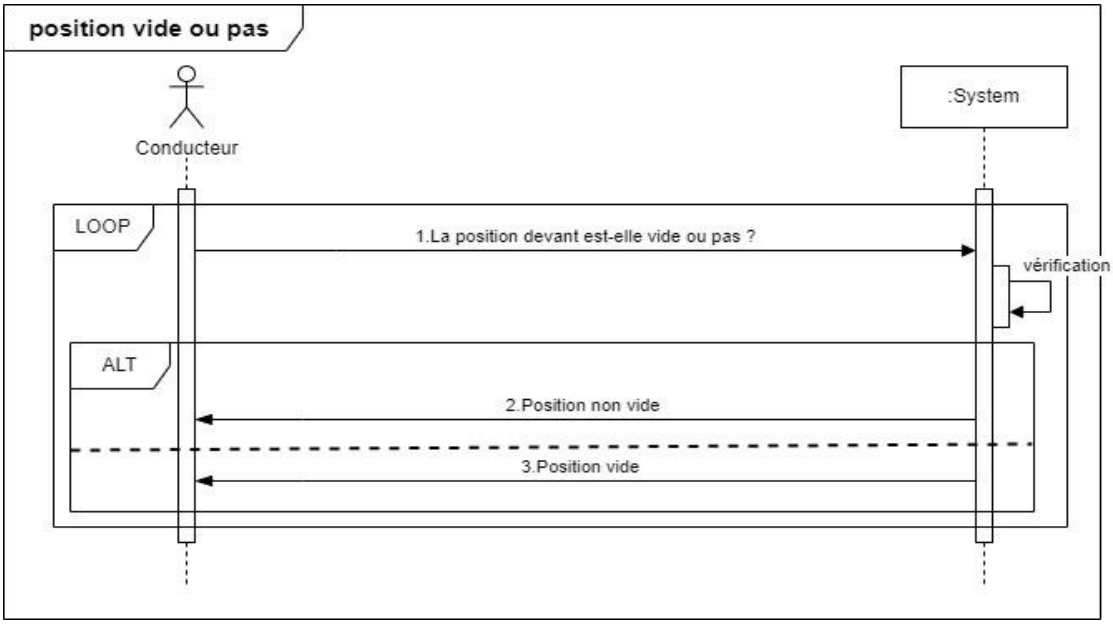

Figure 3.8. Diagramme de séquence « Demander si la position devant est libre »

c. Diagramme de séquence « Gérer le trafic » :

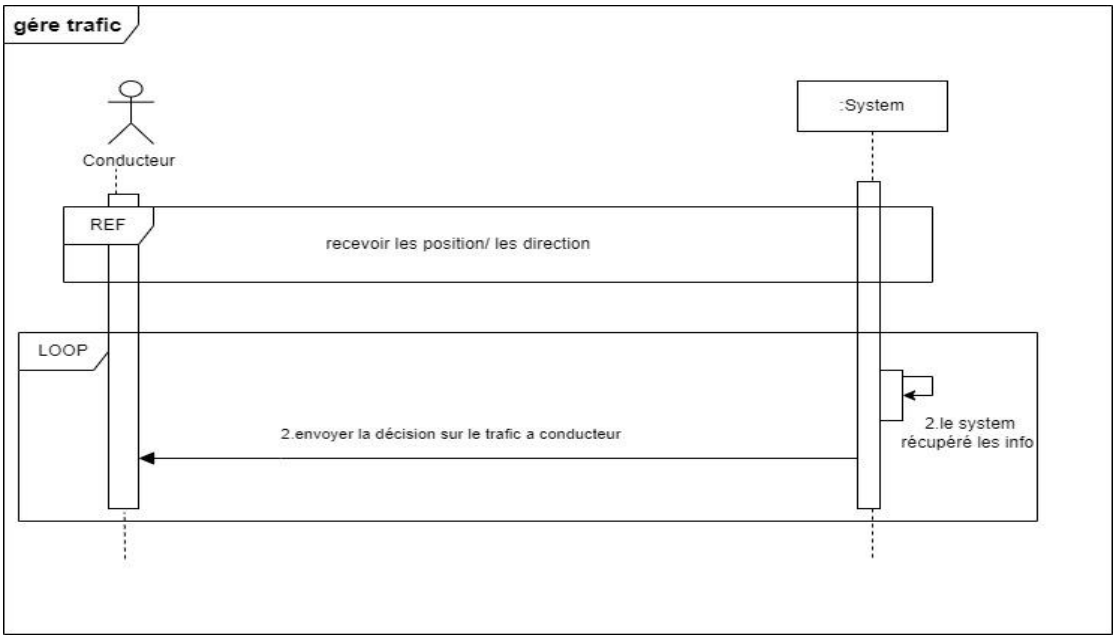

Figure 3.9. Diagramme de séquence « Gérer trafic ».

d. Diagramme de séquence « Notifier le conducteur »

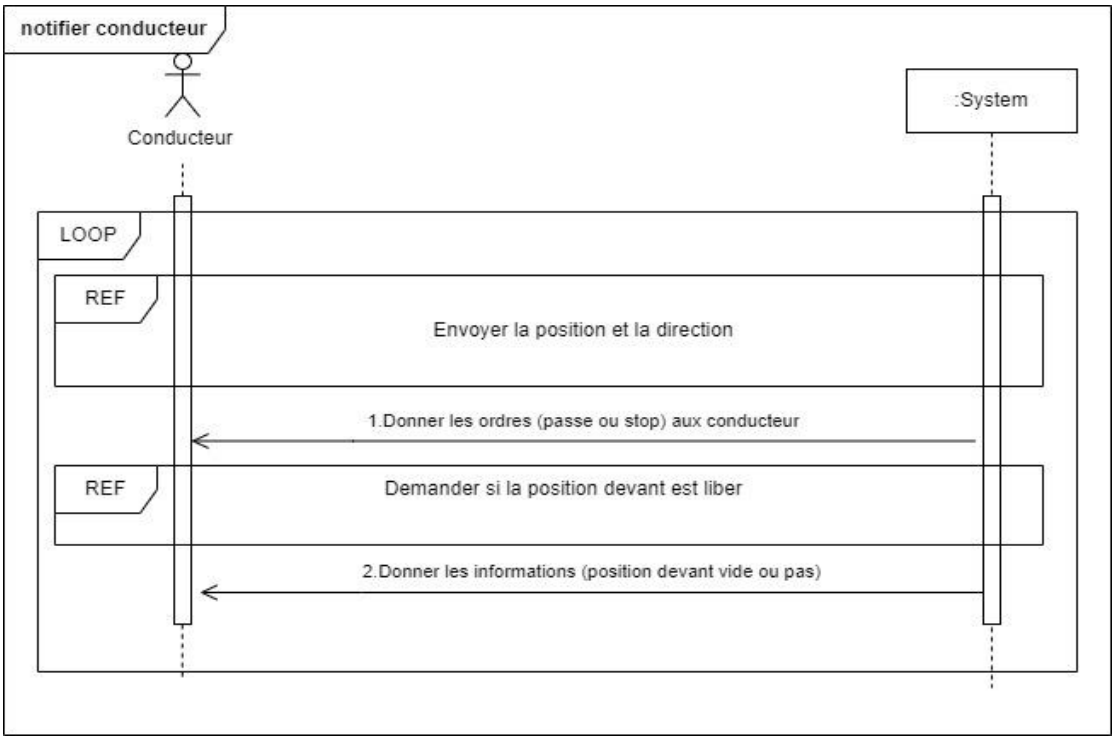

Figure 3.10. Diagramme de séquence « Notifier le conducteur »

5.1.3. Diagramme de protocole d'interactions

Dans la prochaine section, nous présenterons des diagrammes de protocole d'interactions illustrant l'échange de messages entre les différents agents du système.

a. Diagramme de protocole d'interactions « Gérer trafic » :

Dans ce diagramme, l'agent administrateur envoie des requêtes contenant des décisions aux agents conducteurs situés à l'intersection. L'agent conducteur qui reçoit la requête suit les instructions et envoie un message de type INFORM.

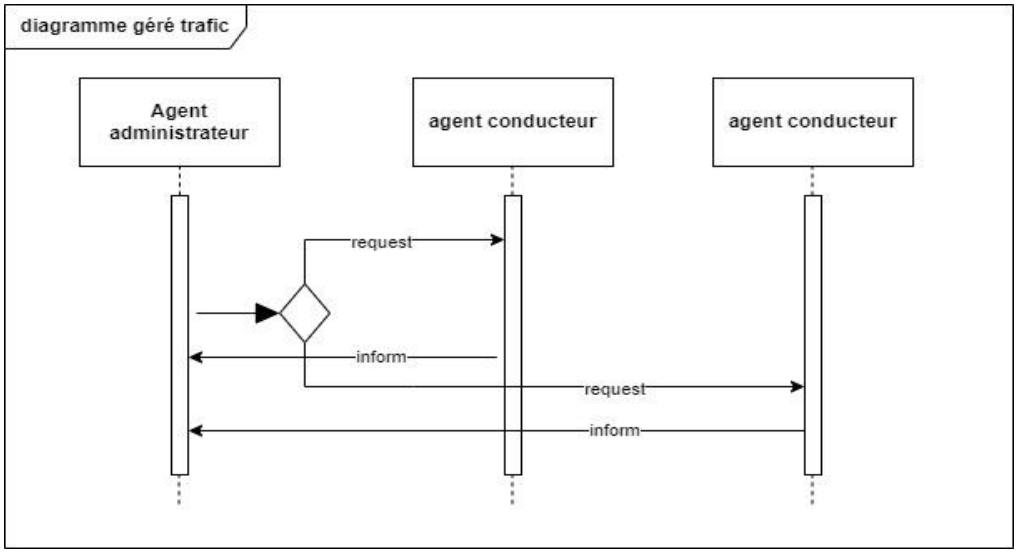

Figure 3.11. Diagramme de protocole d'interactions « Gérer trafic »

b. Diagramme de protocole d'interactions « Envoyer la position et la direction » :

Dans ce diagramme, chaque agent conducteur envoie une requête (position/direction) à l'agent administrateur. L'agent administrateur reçoit et traite la requête. En fonction de la situation, il prend une décision et envoie un message de type INFORM.

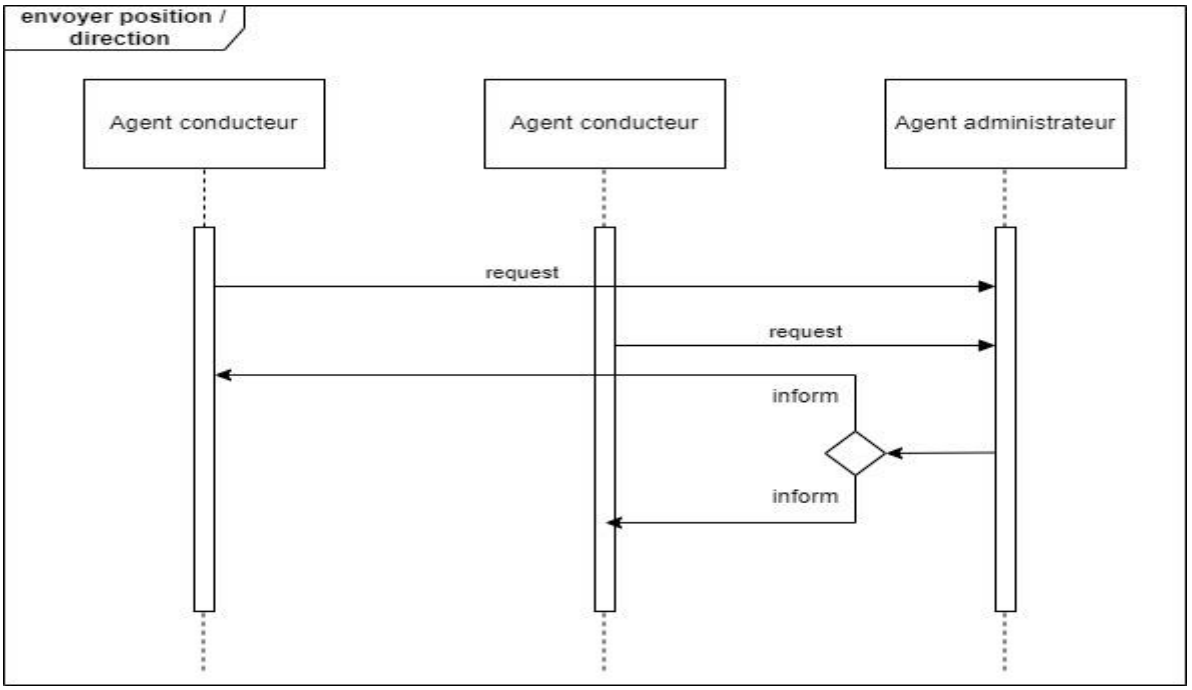

Figure 3.12. Diagramme de protocole d'interactions « Envoyer position et la direction »

c. Diagramme de protocole d'interactions « Demander si la position devant est libre » :

Dans ce diagramme, chaque agent conducteur envoie une requête à l'agent administrateur pour obtenir des informations sur l'état de la position devant eux, afin de savoir si elle est libre ou occupée. L'agent administrateur reçoit la requête et, en fonction de la situation actuelle de la route, répond par "oui" si la position est occupée et "non" si elle est libre.

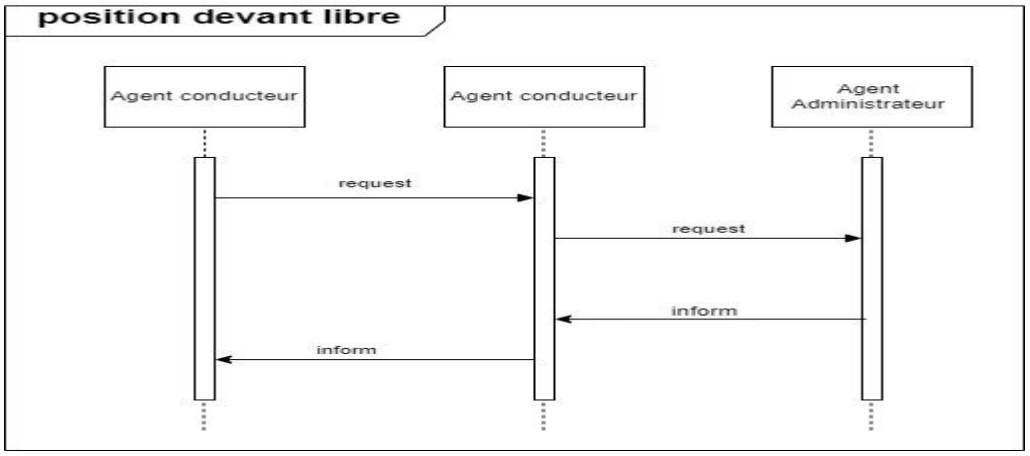

Figure 3.13. Diagramme de protocole d'interactions « Demander si la position devant est libre »

5.1.4. Diagramme de classes :

Dans cette section, nous allons dessiner des diagrammes pour illustrer l'organisation de notre système. Tout d'abord, nous créerons un diagramme représentant les différentes composantes de notre système. Ensuite, nous réaliserons un autre schéma montrant les différentes classes d'agents et leurs interactions au sein de notre système.

5.1.4.1. Diagramme de classes du system :

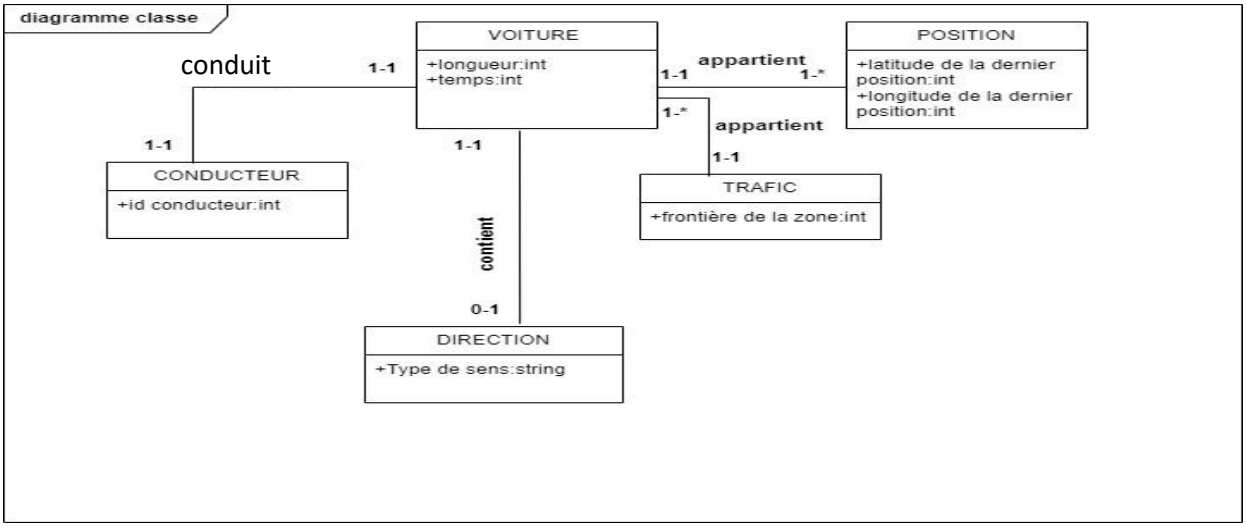

Figure 3.13. Diagramme de classes du système.

5.1.4.2. Diagramme de classes Agent :

Voici le diagramme de classes des agents de notre système. Ce diagramme montre deux classes d'agents qui héritent de la classe principale "Agent".

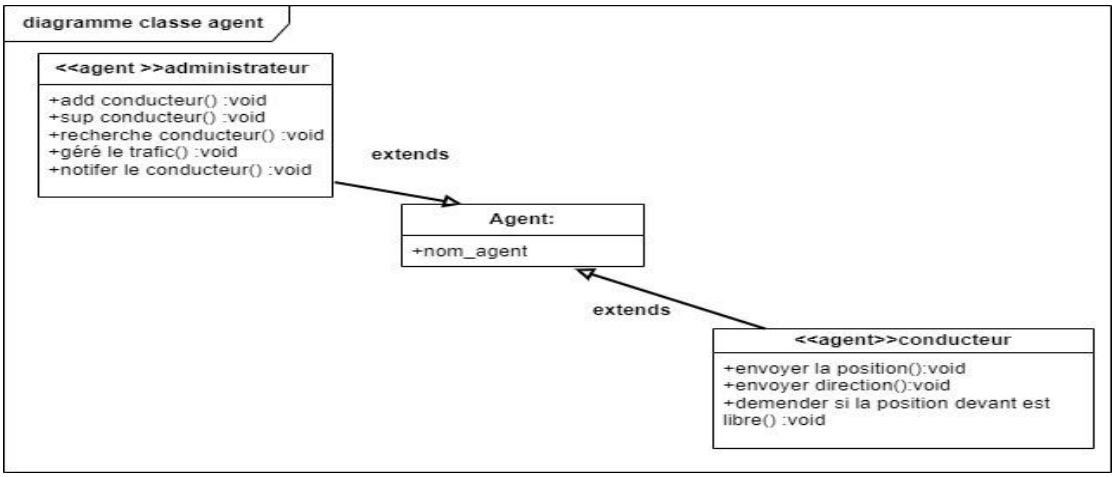

Figure 3.14. Diagramme de classes Agent

Les figures 3.15 et 3.16 fournissent une représentation détaillée des différentes classes d'agents dans notre système.

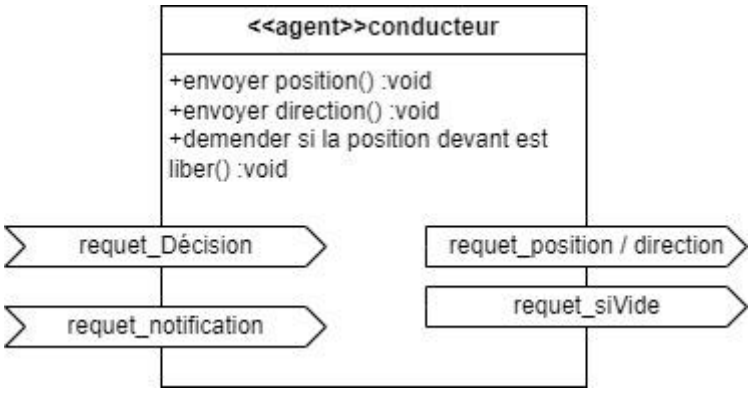

Figure 3.15. Classes Agent « Agent conducteur »

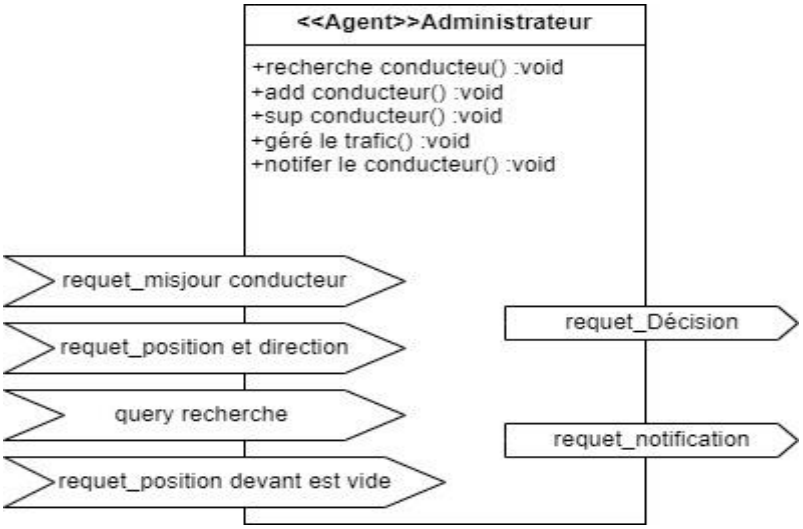

Figure 3.16. Classes Agent « Agent administrateur »

6. Conclusion :

Dans ce chapitre, nous avons expliqué comment nous avons planifié et conçu notre système en utilisant la méthodologie Voyelle et le langage AUML. Tout d'abord, en utilisant la méthode Voyelle, nous avons identifié les agents interagissant dans le système, son architecture globale, ainsi que l'environnement de chaque agent. Ensuite, après avoir identifié les différentes parties du système, nous avons créé des diagrammes AUML pour le représenter. Ces diagrammes comprennent le diagramme de cas d'utilisation, le diagramme de séquence et le diagramme de protocole d'interactions, illustrant comment les parties du

système communiquent entre elles. Nous avons également présenté le diagramme de classe d'agent, modélisant l'aspect statique de notre système.

Chapitre 04 :

Implémentation

1 .Introduction

Dans le chapitre précédent, nous avons proposé une architecture basée sur des agents pour la création d'un système de gestion du trafic routier. Dans ce chapitre, nous allons passer à l'étape cruciale de la réalisation. Nous commencerons par présenter les différents langages et outils utilisés pour développer notre système. Ensuite, nous expliquerons brièvement la manière dont nous avons implémenté notre système avec la plateforme d'agents JADE. Puis, nous présenterons quelques interfaces et la spécification des différents agents du système, en utilisant les outils de JADE. Enfin, nous évaluerons le temps d'attente global des véhicules en utilisant notre système de gestion du trafic routier, en comparant trois heuristiques différentes.

2. Langages et outils de développement

Dans cette section, nous présentons les différents langages et outils utilisés pour le développement de notre application.

2.1 Langage Java

Java est un langage de programmation et une plateforme informatique lancés par Sun Microsystems pour la première fois en 1995. Il s'agit d'un langage orienté objet basé sur des classes, conçu pour être portable, ce qui signifie que le code Java peut fonctionner sur une grande variété de matériels et de systèmes d'exploitation. Java est largement utilisée pour le développement d'applications d'entreprise, d'applications mobiles, de jeux vidéo et d'autres types de logiciels. De plus, Java dispose d'un écosystème important et actif, avec une multitude de bibliothèques et de Framework pour les développeurs **[51].**

2.2 Javafx

JavaFX est une plateforme de développement logiciel et une bibliothèque graphique utilisée pour créer des applications riches en fonctionnalités avec des interfaces utilisateur modernes et interactives. Elle est intégrée à Java depuis la version 8 et offre un ensemble d'outils et de composants pour la création d'applications graphiques **[52].**

2.3 Plateforme Jade

JADE (Java Agent Development Framework) est une plateforme de développement de systèmes multi-agents (SMA) conçue pour faciliter la création et la gestion d'agents intelligents dans des environnements distribués. Il est basé sur les spécifications de FIPA (Foundation for Intelligent Physical Agents) et fournit une structure pour les agents, les

services et les conteneurs. Les agents dans JADE peuvent interagir entre eux et avec d'autres systèmes via des protocoles de communication standardisés **[53]**. Dans la suite, nous présenterons les principaux composants de base de JADE.

2.3.1 Composants de Base : Les composants de base de JADE incluent **[54]** :

***Agents :** Les agents sont les unités de base de JADE. Ils peuvent être créés, déployés et gérés par la plateforme. Les agents peuvent interagir entre eux et avec d'autres systèmes via des protocoles de communication standardisés.

***Services** : Les services sont des composants qui offrent des fonctionnalités spécifiques aux agents. Les services peuvent inclure des mécanismes de gestion de messages, de gestion des événements, etc.

***Conteneurs :** Les conteneurs sont des entités qui gèrent les agents et les services. Ils peuvent être utilisés pour déployer et gérer les agents dans des environnements distribués.

***RMA (Runtime Management Agent)** : Le RMA est un agent qui gère les autres agents dans un conteneur. Il est responsable de la gestion des agents, des services et des événements.

2.3.2 Pourquoi Choisissons-Nous JADE ?

Nous avons choisi JADE (Java Agent Developpement Framework) pour les raisons suivantes :

- Facilité d'installation.

-Documentation détaillée.

-Utilise un langage puissant et stable (JAVA).

-Intégration avec d'autres outils de développement.

- Licence libre.

- Il a été étudié au parcours scolaire.

2.4 IDE Eclipse

Eclipse est un environnement de développement intégré (IDE) libre et extensible, également désignant le projet correspondant lancé par IBM. Il est universel et polyvalent, permettant

potentiellement de créer des projets de développement utilisant n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java, avec l'utilisation de la bibliothèque graphique SWT d'IBM. De plus, ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions **[55]**.

Figure 4.1 : Interface d'Eclipse

2.5. SceneBuilder

SceneBuilder est un outil de conception graphique utilisé pour créer des interfaces utilisateur graphiques (GUI) dans des applications JavaFX. Il permet aux développeurs de créer et de personnaliser visuellement des interfaces utilisateur en utilisant une approche de type glisser-déposer pour ajouter des composants graphiques tels que des boutons, des zones de texte, des menus, etc. SceneBuilder génère ensuite le code FXML correspondant, qui est un langage de balisage utilisé pour décrire la structure de l'interface utilisateur dans JavaFX. Ce code FXML peut être ensuite intégré dans des projets JavaFX pour construire des applications avec des interfaces utilisateur interactives et esthétiques **[56].**

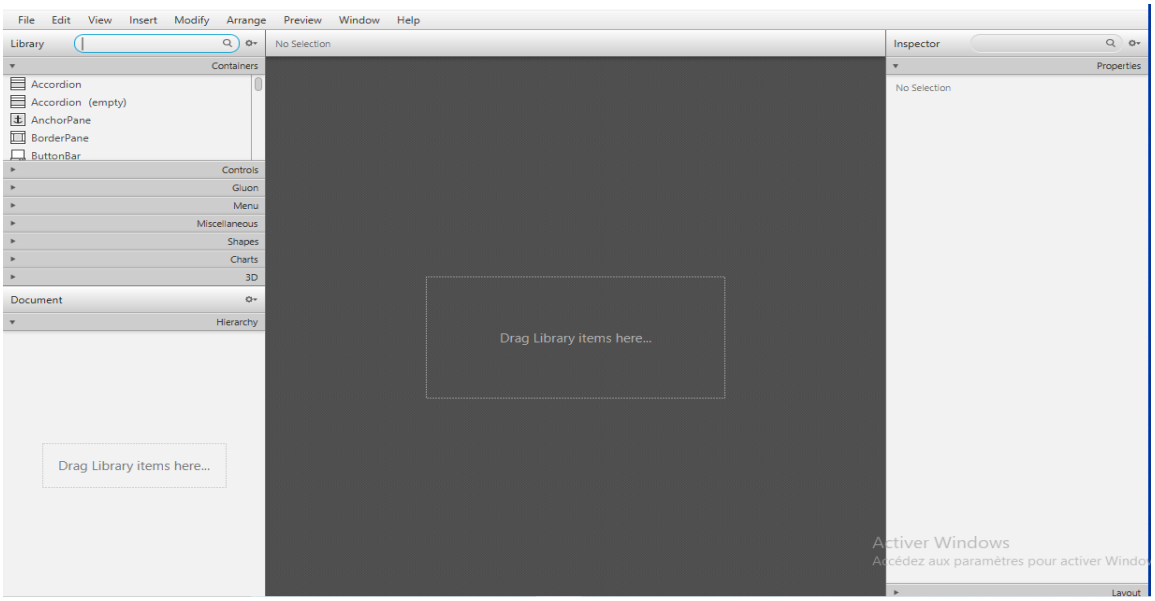

Figure 4.2 : Interface SceneBuilder

3. Implémentation des agents avec Jade :

JADE, un middleware facilitant le développement des systèmes multi-agents (SMA), utilise un modèle où les agents sont créés et gérés dans des conteneurs. Chaque conteneur peut héberger plusieurs agents, et l'ensemble des conteneurs forme une plateforme. Au cœur de cette architecture se trouve le conteneur principal appelé "main-container", qui joue un rôle central en offrant trois modules principaux **[57].**

- Le DF, ou Directory Facilitator, offre un service similaire aux "pages jaunes" pour la plate-forme.
- L'ACC, ou Agent Communication Channel, assure la gestion des communications entre les agents.
- L'AMS, ou Agent Management System, supervise l'enregistrement des agents, leur authentification, ainsi que leur accès et utilisation du système.

L'implémentation d'agents avec JADE (Java Agent Development Framework) implique plusieurs étapes, notamment la configuration de l'environnement JADE, la création d'agents, la définition des comportements et l'interaction entre les agents. Dans la suite nous présenterons les étapes nécessaires pour utiliser JADE **[58].**

3.1. Configuration de l'environnement JADE :

Pour configurer JADE, on suit les étapes suivantes :

-Téléchargez JADE depuis le site officiel et configurez-le dans votre IDE (comme Eclipse ou IntelliJ IDEA).

-Ajoutez les bibliothèques JADE à votre projet Java**.**

3.2. Création d'un Agent :

Pour créer un agent sur JADE, on suit les étapes suivantes :

-Créez une classe Java pour votre agent en étendant jade.core.Agent.

-Implémentez la méthode setup() pour initialiser votre agent et définir ses comportements initiaux.

3.3. Définition des Comportements :

La définition du comportement pour un agent s'effectue comme suit :

-Utilisez les classes de comportement fournies par JADE, comme Behaviour, pour définir les actions de votre agent.

-Ajoutez des comportements à votre agent en utilisant la méthode addBehaviour () dans setup ().

3.4. Interaction entre les Agents :

L’interaction entre les agents de JADE s'effectue comme suit :

-Utilisez le système de communication de JADE pour permettre aux agents de communiquer entre eux.

-Utilisez les méthodes send () et receive () pour envoyer et recevoir des messages entre les agents.

4. Description de notre système :

Dans notre système, nous utilisons un agent administrateur qui agit comme coordinateur pour la gestion du trafic routier, ainsi que plusieurs agents conducteurs, chacun représentant un véhicule.

4.1 Variables et structure de données utilisés :

Dans la suite, nous présentons les variables utilisées dans notre système :

Chaque agent conducteur possède les variables suivantes :

- Une variable pour calculer le temps d'attente de cet agent.
- Une variable qui représente la position de l'agent sur la route.
- Une variable qui représente la direction de l'agent. Cette variable peut prendre les valeurs suivantes : gauche, droite ou directe.

Dans l'agent administrateur, nous définissons les variables suivantes :

- Une variable qui représente le temps d'attente global des véhicules sur les routes, qui correspond à la somme de tous les temps d'attente des agents conducteurs.
- Quatre files d'attente, chacune représentant une partie de l'intersection (V1, V2, H1, H2). Chaque file d'attente contient les informations suivantes : le nom de l'agent conducteur, sa position actuelle, sa direction et son temps d'attente. La création d'un agent conducteur sur une partie de l'intersection consiste à ajouter une case dans la file d'attente appropriée.

De plus, nous utilisons des variables globales pour évaluer la vitesse des véhicules et le temps de déplacement d'une position de la route à une autre.

4.2 Fonctionnements du système :

Notre système de gestion du trafic routier fonctionne comme suit :

Après avoir défini la vitesse des véhicules et le temps de déplacement entre les positions sur les routes, la création d'un nouvel agent conducteur consiste à initialiser sa variable de temps d'attente à 0.

À chaque unité de temps de déplacement, chaque agent conducteur envoie à l'agent administrateur les informations suivantes : sa position actuelle, sa direction et son temps d'attente, qui augmente en ajoutant la valeur du temps de déplacement si l'agent conducteur ne bouge pas. Si l'agent se déplace vers l'avant (si la position en avant est vide), son temps d'attente reste inchangé. L'agent conducteur demande également à l'agent administrateur si la position en avant est vide ou non, et il se déplace si elle est vide.

De son côté, à chaque unité de temps de déplacement, l'agent administrateur reçoit toutes les positions des agents conducteurs et décide quel agent doit passer par l'intersection parmi ceux qui la rejoignent, en utilisant un algorithme prédéfini (voir la section suivante). L'agent administrateur répond également aux requêtes des agents conducteurs concernant l'occupation ou non de la position en avant.

4.3 Algorithmes utilisés :

Afin d'évaluer l'efficacité de notre système de gestion du trafic routier, nous commençons par définir une fonction objective, qui est de minimiser le temps d'attente global des véhicules sur les routes. Ensuite, nous définissons et appliquons trois heuristiques différentes : une heuristique FIFO et deux versions de l'heuristique Greedy, l'une simple et l'autre avancée. Dans la suite, nous expliquons ces trois heuristiques proposées.

4.3.1. L'algorithme FIFO (First In, First Out) : Comme définition, FIFO est une méthode d'ordonnancement des processus qui traite les tâches dans l'ordre chronologique de leur arrivée. Les processus sont placés dans une file d'attente selon leur ordre d'arrivée. Lorsque le processeur est libre, il est attribué au processus en tête de cette file d'attente. Le processus actif libère le processeur lorsqu'il se termine ou lorsqu'il demande une opération d'entrée-sortie. Cet algorithme est simple et efficace pour gérer les processus arrivants successivement et nécessitant un traitement dans l'ordre de leur arrivée.

Dans notre projet de gestion du trafic, nous rencontrons des intersections, et nous utilisons un algorithme FIFO pour réguler le mouvement des véhicules. L'algorithme FIFO assure que les véhicules sont autorisés à traverser l'intersection dans l'ordre où ils arrivent, traitant chaque voiture successivement. Lorsque l'accès à l'intersection est autorisé, le premier véhicule de la file FIFO est déplacé en priorité. Si deux ou plusieurs véhicules rejoignent l'intersection simultanément, notre algorithme FIFO sélectionne un véhicule au hasard. Cela permet de gérer efficacement le flux de trafic, de minimiser les temps d'attente, tout en garantissant un

traitement équitable de chaque véhicule arrivant à l'intersection.

FIFO
Si un seul véhicule dans l'intersection **alors** Donner autorisation de passage à ce véhicule **Sinon** // 2 ou plus véhicules arrive en même temps alors Choix **aléatoire** d'un véhicule **Fin si** Mise à jour (Temps d'attente pour chaque agent) Mise à jour (Temps d'attente global)

4.3.2. L'algorithme Greedy : Comme définition, un algorithme glouton (ou greedy algorithm) est une méthode d'optimisation qui suit le principe de faire des choix locaux optimums pour espérer obtenir un résultat optimum global. Il construit une solution incrémentalement, en maximisant à chaque étape un critère local sans revenir sur les décisions prises précédemment. Bien que non garantissant l'optimalité, il est simple et efficace pour résoudre des problèmes d'optimisation.

Dans notre projet, nous utilisons un algorithme glouton (greedy) pour choisir les véhicules qui passent l'intersection, de manière à minimiser le temps d'attente global de tous les véhicules. Cet algorithme fonctionne comme suit :

- Il calcule d'abord le temps d'attente des voitures sur chaque branche de l'intersection (V1, V2, H1, H2).
- Ensuite, il évalue le temps d'attente global de tous les véhicules à l'intersection en simulant le passage d'un véhicule de chaque branche (V1, V2, H1, H2).
- Enfin, il compare les temps d'attente globaux obtenus lors de l'étape précédente et choisit le véhicule en tête de la branche qui a le temps d'attente le plus faible. En cas d'égalité entre deux ou plusieurs branches, un véhicule est choisi aléatoirement parmi celles-ci.

Ainsi, cet algorithme permet de fluidifier la circulation à l'intersection tout en minimisant le temps d'attente global des véhicules.

Gloutonne:

Si un seul véhicule dans l'intersection **alors**

Donner autorisation de passage à ce véhicule

Sinon // 2 ou plus véhicules arrive en même temps **alors**

Pour chaque véhicule à l'intersection **faire**

Calculer (Temps d'attente global) si on va choisir ce véhicule

Fin pour

Si on obtient un temps global minimal **alors**

Donner autorisation de passage au véhicule approprié

Sinon // égalité entre 2 ou plus temps d'attente global **alors**

Choix **aléatoire** d'un véhicule

Fin si

Mise à jour (Temps d'attente pour chaque agent)

Mise à jour (Temps d'attente global)

4.3.3. L'algorithme glouton avancé : Cet algorithme fonctionne de manière similaire à l'algorithme glouton de base, avec l'ajout de la possibilité de faire passer deux véhicules simultanément. Cet algorithme évalue les véhicules en attente dans chaque direction, identifie les paires de véhicules pouvant traverser sans collision, et sélectionne la paire qui minimise le temps d'attente global.

Gloutonne avancée:

Si un seul véhicule dans l'intersection alors
Donner autorisation de passage à ce véhicule
Sinon // 2 ou plus véhicules arrive en même temps alors
Pour chaque véhicule à l'intersection faire
Calculer (Temps d'attente global) si on va choisir ce véhicule
Pour chaque autre véhicule à l'intersection faire
Calculer (Temps d'attente global) si on va choisir ce véhicule avec l'autre véhicule si la situation le permet

Fin pour
Fin pour
Si on obtient un temps global minimal alors
Donner autorisation de passage à/aux véhicules appropriés
Sinon // 2 ou plus véhicules ont le même temps d'attente global alors
Choix aléatoire d'un véhicule ou deux véhicules selon la situation
Fin si
Fin si
Mise à jour (Temps d'attente pour chaque agent)
Mise à jour (Temps d'attente global)

La figure 4.3 illustre les scénarios de trafic où deux véhicules traversent l'intersection en même temps.

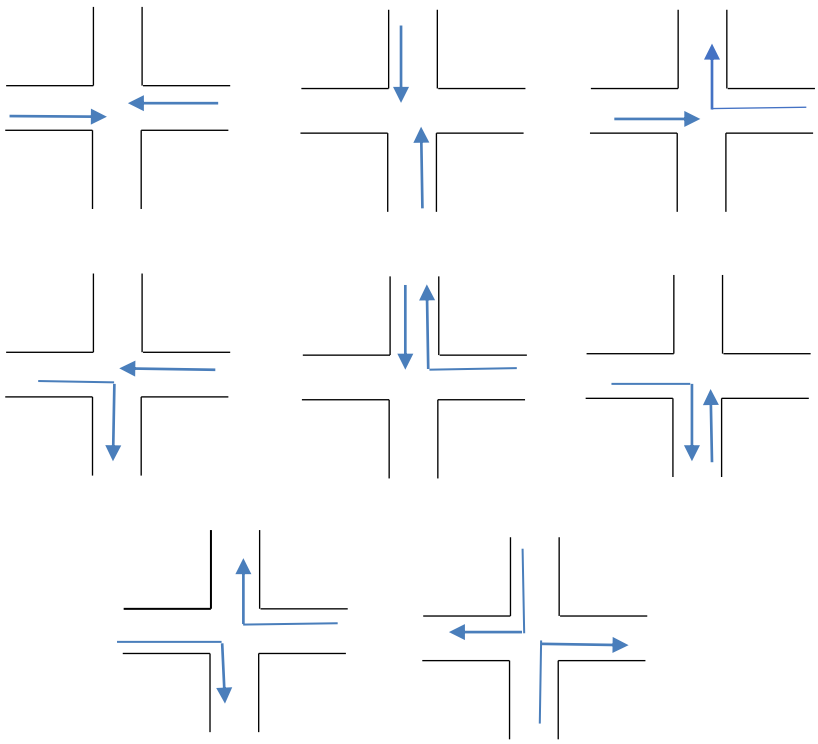

Figure4.3 Les cas de trafic qui traverse deux véhicules simultanément

5. Interfaces principales du système :

Cette section donne une description du système réalisé. Nous allons d'abord présenter les différentes interfaces du système développé, puis nous simulerons l'exécution d'un scénario en utilisant les fonctionnalités de Jade.

Cette section présente quelques interfaces de notre système.

5.1. Interface d'initialisation des agents :

La figure 4.4 illustre l'interface utilisée pour initialiser les agents conducteurs. Nous y spécifions le nombre d'agents, leur répartition sur les quatre parties de l'intersection, le temps de déplacement des agents d'une position à l'autre, ainsi que l'espace entre les agents (véhicules). Ensuite, pour chaque agent, nous indiquons sa position, sa direction et son temps d'attente. En cliquant sur le bouton « Visualiser », le système commence à fonctionner de manière autonome avec la situation initialement créée.

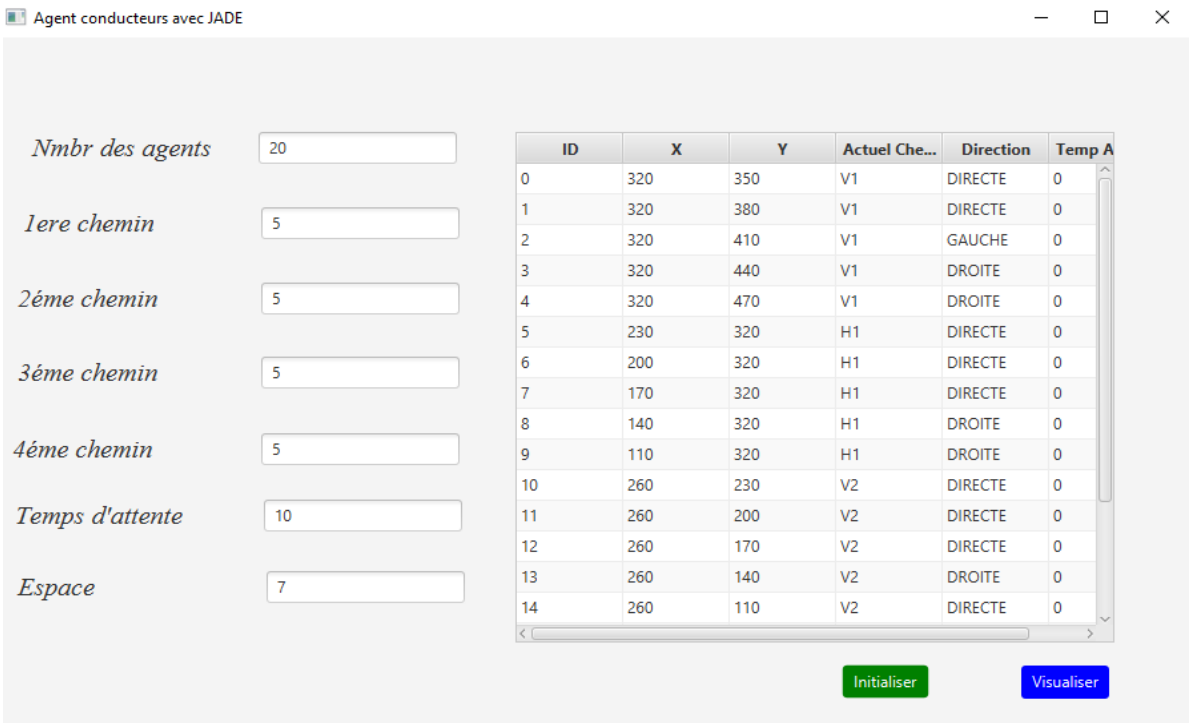

Figure 4.4 Interface d'initialisation des agents

5.2 Interface visualisation

La figure 4.5 illustre l'interface de visualisation de notre système, dans laquelle on peut observer la circulation des véhicules dans l'intersection selon l'algorithme choisi.

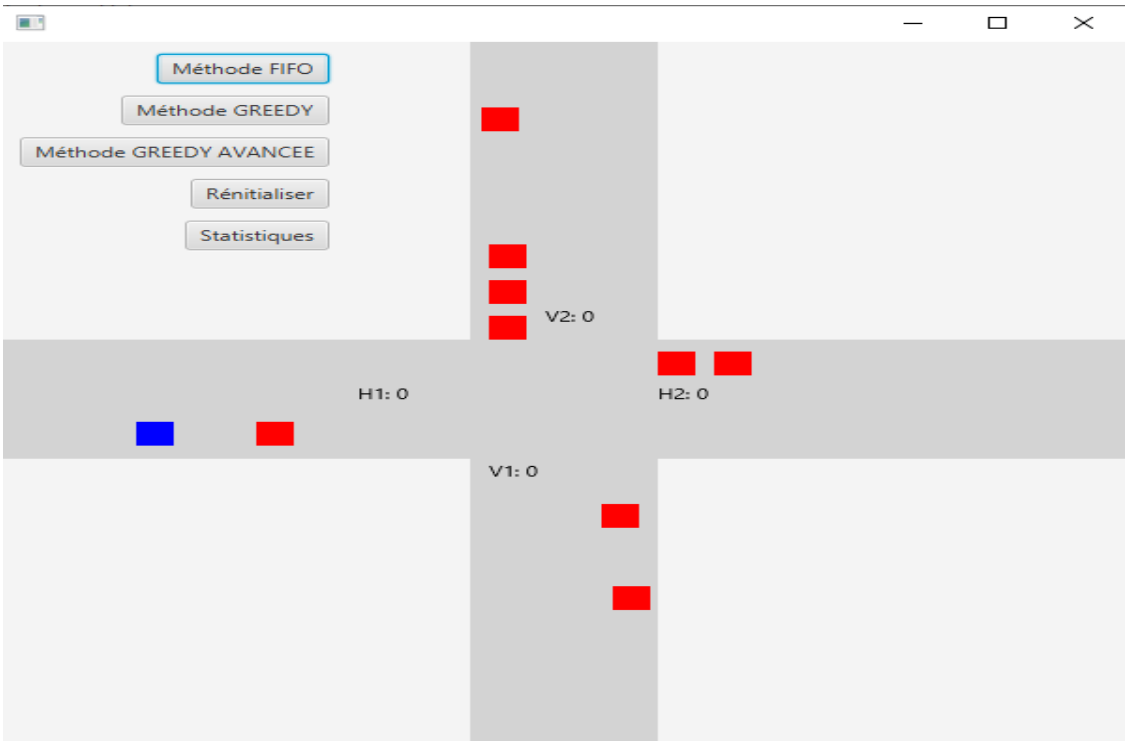

Figure4.5 Interface visualisation

5.3 Interface RMA

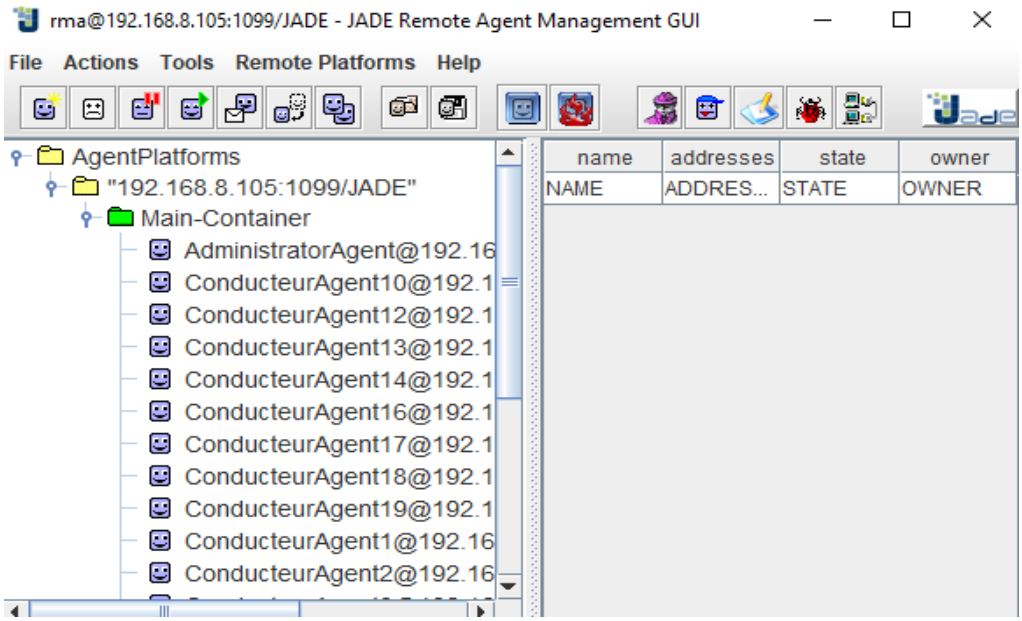

Figure4.6 Interface RMA

5.4 Interface sniffer agent

La figure 4.7 illustre l'interface "sniffer" de notre système. Cette interface permet de visualiser graphiquement les différentes interactions entre les agents de notre système.

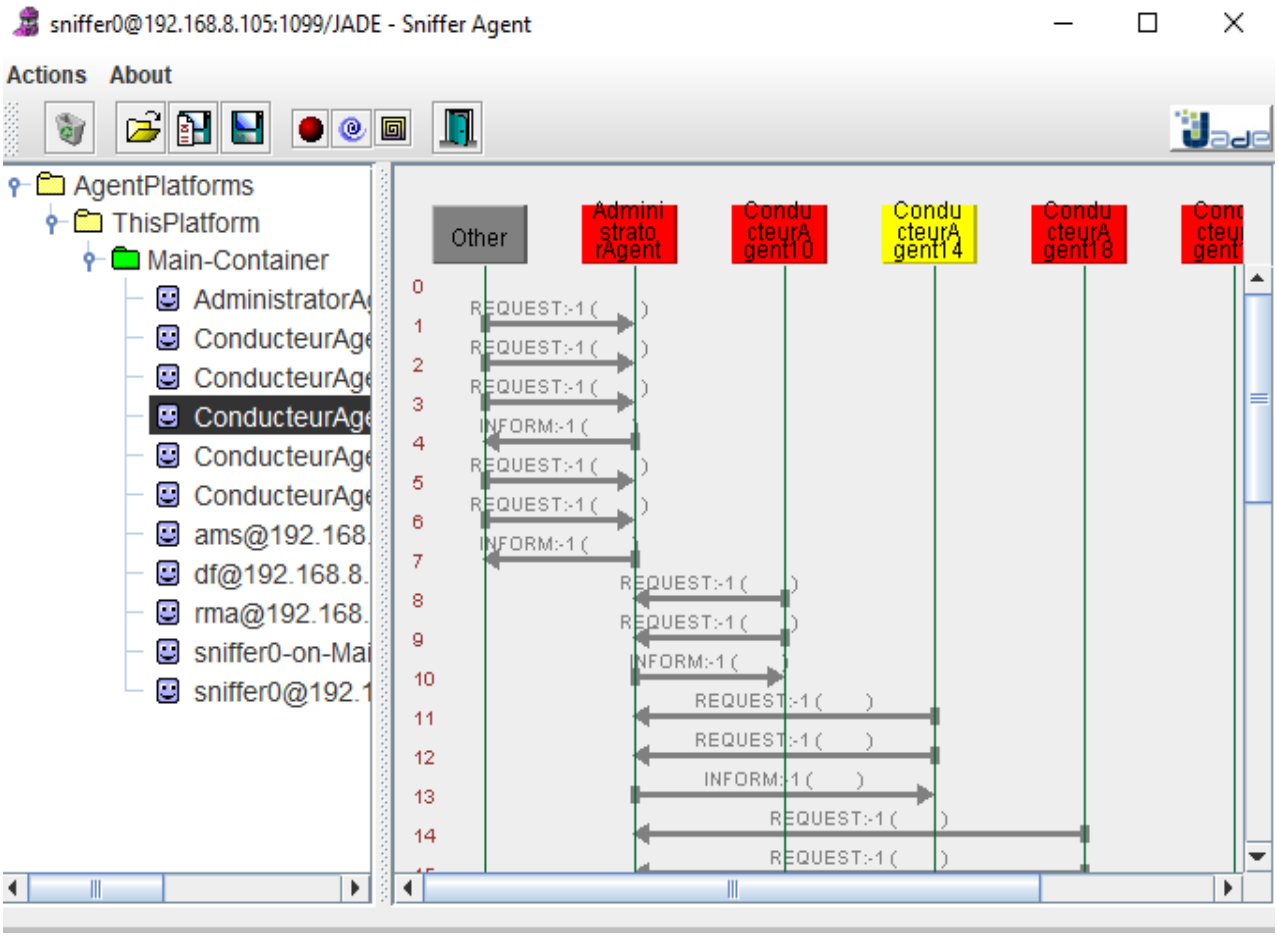

Figure4.7 sniffer agent

5.5 Interface de comparaison entre les trois algorithmes

La figure 4.8 illustre l'interface utilisée pour comparer les résultats obtenus de l'exécution des trois algorithmes sur la même situation initiale en termes de temps d'attente global.

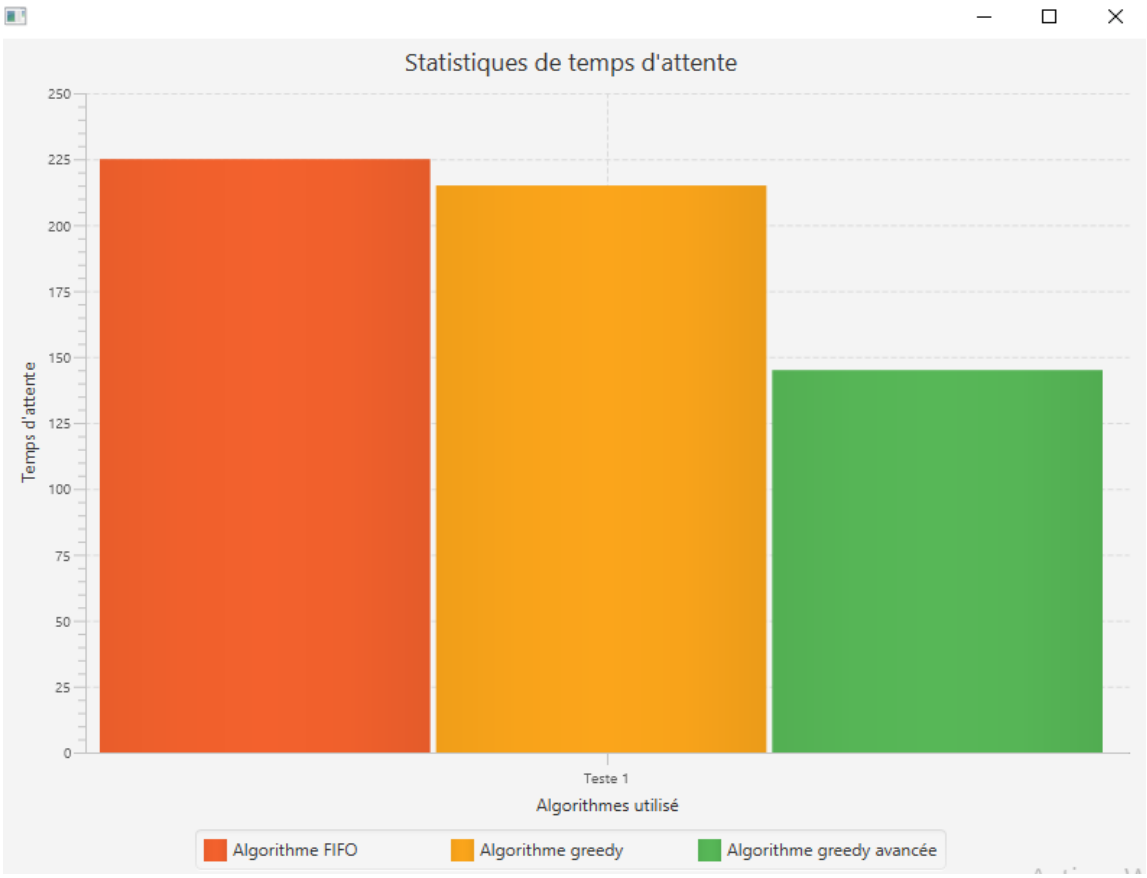

Figure 4.8 Interface de comparaison entre les trois algorithmes

6. Evaluation du système :

Dans cette section, nous évaluons notre système de gestion du trafic routier en l'exécutant avec différentes situations initiales. Pour chaque situation initiale, nous appliquons les trois algorithmes : FIFO, glouton (greedy) et glouton avancé. Ensuite, nous utilisons une interface de comparaison des résultats des trois algorithmes pour présenter les temps d'attente globaux obtenus. Nous détaillons ensuite les situations et les résultats obtenus pour chacune d'elles.

6.1 : Première situation : Dans cette configuration, nous plaçons un nombre égal d'agents dans chaque partie de l'intersection, par exemple 10 agents par partie, avec des directions variées dans quatre parties de l'intersection. La figure 4.9 illustre les résultats obtenus.

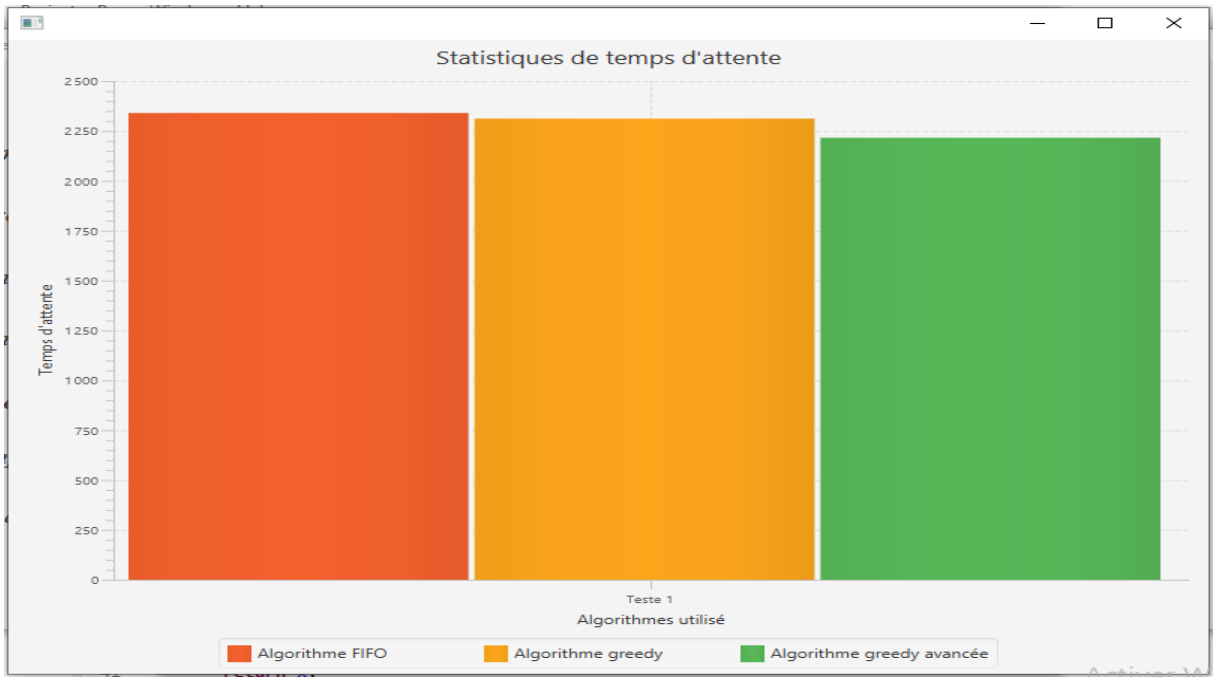

Figure 4.9 Résultats obtenus pour la première situation

6.2 : Deuxième situation : Dans cette situation, nous plaçons un nombre variable d'agents dans chaque partie de l'intersection : par exemple, 10 agents dans la partie V1, 8 agents dans la partie V2, 6 agents dans la partie H1, et 4 agents dans la partie H2. avec des directions variées. La figure 4.10 illustre les résultats obtenus.

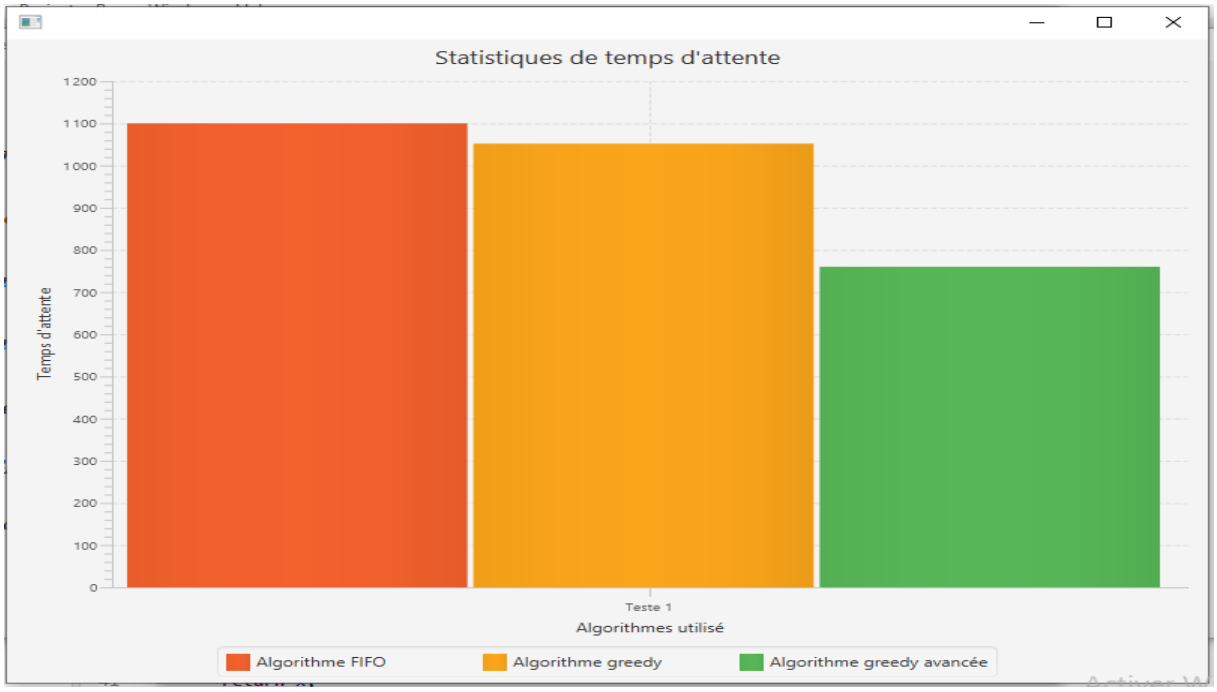

Figure 4.10 Résultats obtenus pour la deuxième situation

6.4 : troisième situation : Dans cette situation, nous plaçons un nombre variable d'agents avec des directions variées dans trois parties de l'intersection, tandis que la quatrième partie reste vide. La figure 4.12 illustre les résultats obtenus.

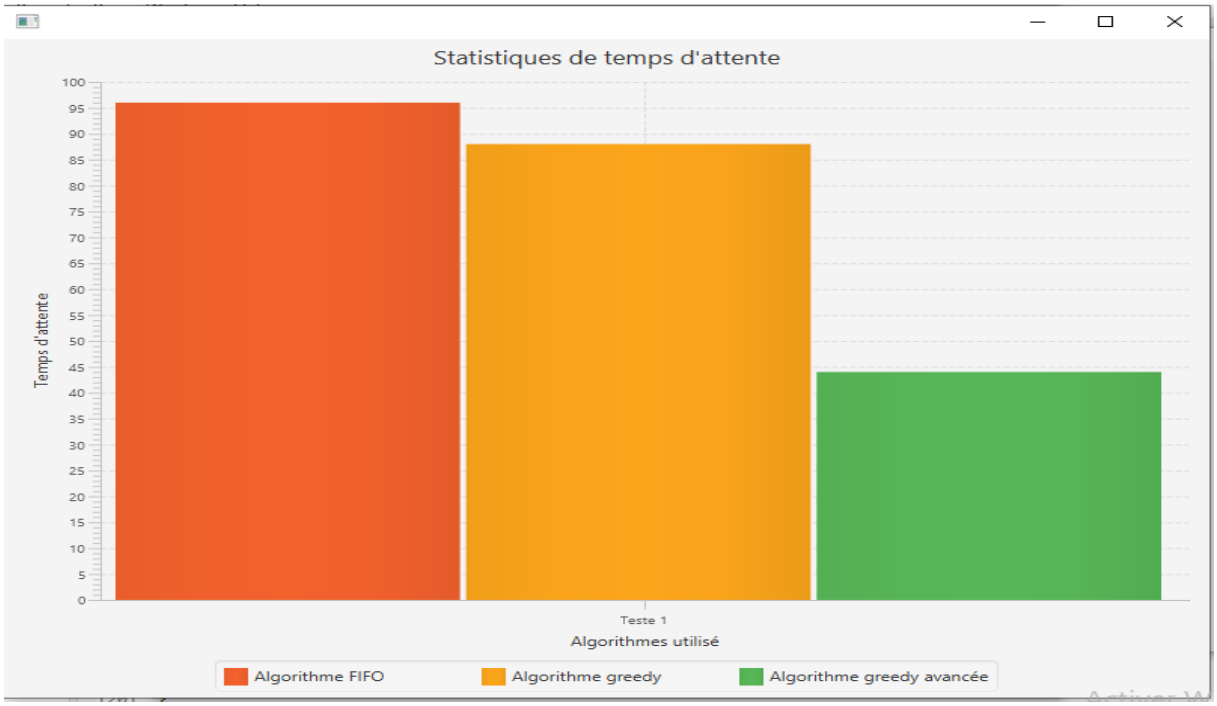

Figure 4.12 Résultats obtenus pour la troisième situation

6.5 : quatrième situation : Dans cette situation, nous plaçons un nombre variable d'agents avec des directions variées dans deux parties de l'intersection, tandis que les deux autres parties restent vides. La figure 4.13 illustre les résultats obtenus.

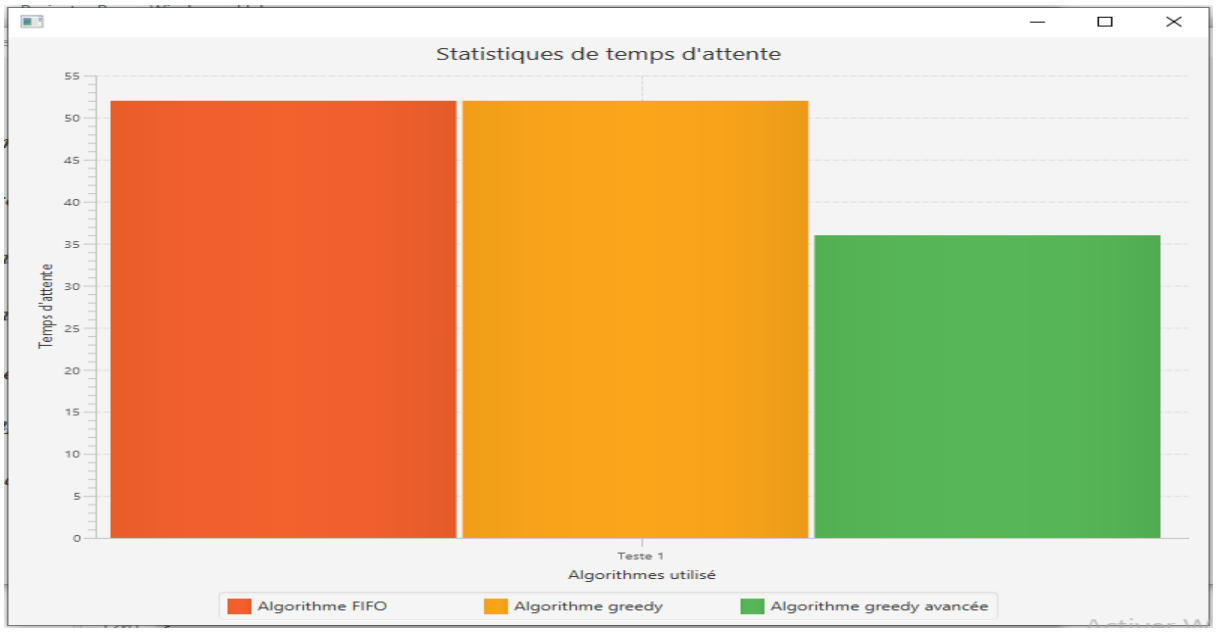

Figure 4.13 Résultats obtenus pour la quatrième situation

6.8 Discussion des résultats : Les figures 4.9 à 4.13 montrent la comparaison entre les trois algorithmes FIFO, glouton et glouton avancé en termes de temps d'attente global des véhicules dans l'intersection. Les résultats obtenus montrent clairement que l'algorithme glouton avancé offre le temps d'attente global le plus faible, suivi par l'algorithme glouton, tandis que l'algorithme FIFO arrive en dernière position. Ainsi, l'utilisation de l'heuristique glouton plutôt que FIFO réduit significativement le temps d'attente global des véhicules et améliore par conséquent la gestion du trafic routier. De plus, l'utilisation de l'heuristique glouton avancé diminue encore plus le temps d'attente global des véhicules par rapport aux autres algorithmes, en optimisant le passage de deux véhicules simultanément dans l'intersection. En conclusion, l'heuristique glouton avancé se révèle être la meilleure solution pour gérer efficacement le trafic routier et satisfaire les conducteurs.

7. Conclusion :

Ce chapitre pose les bases de notre travail pratique en démontrant comment les concepts théoriques précédemment présentés sont mis en œuvre pour créer un système fonctionnel et efficace de gestion du trafic routier. Nous avons d'abord introduit les langages, environnements, outils et frameworks utilisés pour implémenter notre système. Ensuite, nous avons décrit notre système en détaillant les variables, les structures de données utilisées et la spécification des différents algorithmes. Nous avons également présenté et expliqué les interfaces de notre application. Enfin, nous avons évalué le système et discuté des résultats obtenus. Et pour chaque heuristique nous calculant le temps d'attente global des véhicules en exécutons le système avec des situations déférentes des routes. Enfin, nous comparent les résultats obtenus pour choisir la meilleure heuristique.

Conclusion générale

Conclusion générale

La gestion efficace du trafic routier demeure un défi complexe mais crucial dans nos environnements urbains en constante évolution. Les solutions traditionnelles, souvent limitées par des contraintes technologiques et une approche centralisée, peinent à s'adapter aux dynamiques de trafic en temps réel. Cependant, ce mémoire a démontré qu'il est possible de relever ce défi grâce à une approche innovante basée sur les systèmes multi-agents.

Nous avons proposé un système multi-agents agissant d'abord comme simulateur de trafic routier, avec une vision future de servir de base à un système de gestion de trafic réel intégrant des capteurs et des actionneurs interconnectés, communiquant avec les agents du noyau. Un chapitre supplémentaire (voir l'annexe A) illustre comment notre simulateur peut être utilisé dans un système réel de gestion du trafic routier. Dans ce système, chaque véhicule sur la route est représenté par un agent appelé conducteur, qui se comporte comme un véhicule réel ; il se déplace de manière autonome et suit les ordres de l'agent de gestion du trafic, appelé administrateur. Pour choisir la meilleure façon de gérer le trafic, nous avons intégré des algorithmes tels que FIFO (First In First Out) et deux versions de l'heuristique glouton (Greedy), l'une simple et l'autre avancée. Nous avons ensuite comparé ces trois heuristiques en termes de réduction du temps d'attente global, en collectant et en discutant les résultats obtenus.

Pour réaliser ce travail, nous avons adapté la méthode de développement VOYELLE, largement utilisée pour modéliser les systèmes multi-agents. Nous avons complété cette méthode par l'utilisation du langage AUML (Agent UML) pour l'analyse et la conception de notre système. Pour implémenter notre système, nous avons utilisé l'environnement de développement Eclipse avec la plateforme des agents JADE, et pour concevoir et implémenter l'interface de l'application, nous avons utilisé le Framework JAVAFX.

Les résultats obtenus sont satisfaisants, et une comparaison entre les heuristiques FIFO, glouton et glouton avancé montre que l'utilisation de l'heuristique glouton avancé réduit significativement le temps d'attente global des véhicules sur les routes et optimise la gestion du trafic routier.

Perspectives

Pour les perspectives futures, nous proposons :

- D’étendre l’architecture proposée par l’utilisation de plusieurs agents administrateurs, chacun responsable d’une intersection de la ville, avec des mécanismes de négociation entre ces agents pour optimiser la gestion du trafic routier.
- D'utiliser d’autres heuristiques et méthodes d’optimisation plus sophistiquées pour améliorer notre système autant que possible.
- D’utiliser ce système comme noyau d’un système réel de gestion du trafic routier, en intégrant des capteurs et actionneurs et en tirant parti des évolutions technologiques dans le domaine de l’Internet des Objets (IoT). Ce système réel permettrait également de collecter des données de circulation, utilisées pour améliorer le simulateur.

Bibliographie

[1] Sébastien Faye (2014). Contrôle et gestion du trafic routier urbain par un réseau de capteurs sans fil. Thèse de doctorat en Informatique et réseaux. Université de Paris, ENST.

[2] Maria Viorela Muntean. Multi-Agent System for Intelligent Urban Traffic Management Using Wireless Sensor Networks Data. Sensors 2021, 22, 208. DOI: 10.3390/s22010208.

[3] Sathiyaraj Rajendran, & A. Bharathi. (2020). An efficient intelligent traffic light control and deviation system for traffic congestion avoidance using multi-agent system. Transport, 35(3), 327-335. DOI: 10.3846/transport.2019.11115

[4] Saju Sankar S & Vinod Chandra. A multi-agent ant colony optimization algorithm for effective vehicular traffic management. In Proceedings of the International Conference on Swarm Intelligence, Belgrade, Serbia, 14–20 July 2020; Springer: Cham, Switzerland, 2020; pp. 640–647. DOI: 10.1007/978-3-030-53956-6_59

[5]: Jinghui Wang, Wei Lv, Yajuan Jiang, Shuangshuang Qin et Jiawei Li. A multi-agent based cellular automata model for intersection traffic control simulation[J]. Physica A: Statistical Mechanics and its Applications, 2021, 584: 126356. DOI:10.1016/j.physa.2021.126356

[6] https://www.securite-routiere-az.fr/t/trafic/

[7] https://gcononmerci.org/bon-a-savoir/infographie-cest-quoi-un-trafic-routier/

[8] https://www.ecologie.gouv.fr/sites/default/files/17089_Gestion-trafic-milieu urbain_Web_planches_FR.pdf

[9] https://www.lafabriquedelacite.com/publications/mieux-comprendre-la-congestion-urbaine-pour-y-repondre-its-the-economy-stupid/

[10] https://collectivitesviables.org/articles/apaisement-de-la-circulation.aspx

[11] https://www.bison-fute.gouv.fr/imprimer,article11038.html

[12] https://www.qonexio.fr/guide/contexte-enjeux-multiples-du-secteur-du-transport/

[13] https://journals.openedition.org/vertigo/12816

[14] http://www.bv.transports.gouv.qc.ca/mono/1135462.pdf

[15] Sébastien Faye, Claude Chaudet et Isabelle Demeure. “A Distributed Algorithm for Adaptive Traffic Lights Control”. Dans : 15th IEEE Intelligent Transportation Systems Conference (ITSC 2012). Anchorage, USA, sept. 2012.

[16]: M. Behrisch, L. Bieker, J. Erdmann et D. Krajzewicz. “SUMO- Simulation of Urban MObility : An Overview”. Dans : The Third International Conference on Advances in System Simulation (SIMUL 2011). Barcelona, Spain, oct. 2011, p. 63–68.

[17]: András Varga et Rudolf Hornig. “An overview of the OMNeT++ simulation environment”. Dans : Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops. ICST (Institute forComputer Sciences, Social Informatics et Telecommunications Engineering). 2008, p. 60.

[18] https://www.cnrtl.fr/definition/agent

[19] https://www.larousse.fr/dictionnaires/francais/agent/1629

[20] https://turing.cs.pub.ro/auf2/html/chapters/chapter2/chapter_2_2_1.html

[21] https://turing.cs.pub.ro/auf2/html/chapters/chapter2/chapter_2_1_2.html

[22] https://www.geeksforgeeks.org/agents-artificial-intelligence/

[23] https://fr.wikipedia.org/wiki/Agent_intelligent

[24] https://www.journaldunet.fr/intelligence-artificielle/guide-de-l-intelligence artificielle/1501293-agent-intelligent-definition-et-exemples/

[25] https://www.irit.fr/~Chihab.Hanachi/Cours/SMA/CoursAgentsI.pdf

[26] https://botpress.com/fr/blog/quels-sont-les-differents-types-d-agents-de-l-intelligence-artificielle

[27] https://theses.hal.science/tel-00474542

[28] https://archipel.uqam.ca/5684/1/M12885.pdf

[29] https://www.lirmm.fr/~ferber/ProgAgent/agents_cognitifs.pdf.

[30] https://www.techno-science.net/glossaire-definition/Systeme-multi-agents.html

[31] https://24pm.com/117-definitions/423-systeme-multi-agents.

[32] https://fr.wikipedia.org/wiki/Syst%C3%A8me_multi-agents.

[33] https://www.researchgate.net/figure/Schema-dun-systeme-multi-agentFerber-1995_fig3_337544841.

[34] https://www.lirmm.fr/~ferber/publications/LesSMA_Ferber.pdf

[35] https://www.mcours.net/cours/pdf/leilclic3/leilclic933.pdf

[36] https://thema.univfcomte.fr/theoq/pdf/2005/TQ2005%20ARTICLE%2015.pdf

[37] https://www.eyrolles.com/Informatique/Livre/organisation-et-applications-des-sma-9782746204393/

[38] https://jade.tilab.com/

[39] Finin, T.; Fritzson, R.; McKay, D.; McEntire, R. (1994). "KQML as an agent communication language". Proceedings of the third international conference on Information and knowledge management - CIKM '94. p. 456. doi:10.1145/191246.191322. ISBN0897916743. S2CID 1129799

[40]https://theses.hal.science/search/index?q=Contr%C3%B4le+et+gestion+du+trafic+routier+urbain+par+un+r%C3%A9seau+de+capteurs+sans+fil

[41]https://theses.hal.science/search/index?q=++Mod%60eles+de+distribution+pour+la+simulation+de+trafic+multi-agent

[42] [Arpaci-Dusseau, Remzi H.; Arpaci-Dusseau, Andrea C. (2014), Operating Systems: Three Easy Pieces [Chapter Scheduling Introduction] , Arpaci-Dusseau Books.

[43] https://fr.wikipedia.org/wiki/Round-robin_(informatique)

[44] Corbató, Fernando J.; Merwin-Daggett, Marjorie; Daley, Robert C. (1962). "An experimental time-sharing system". Proceedings of the May 1-3, 1962, spring joint computer conference on - AIEE-IRE '62 (Spring).
p. 335. doi:10.1145/1460833.1460871. S2CID 14363753

[45] Silberschatz, Abraham; Galvin, Peter Baer; Gagne, Greg (2008). Operating system concepts (8th ed.). Hoboken, N.J.: Wiley. ISBN 978-0470128725.

[46] B. Bauer. Extending UML for the Specification of Interaction Protocols. submission for the 6th Call for Proposal of FIPA and revised version part of FIPA 99, 1999.

[47] J. Odell, H. v. D. Paranuk, B. Bauer: Representing Agent Interaction Protocols in UML, submitted for Autonomous Agents 2000, 2000

[48] https://www.rapport-gratuit.com/auml-la-modelisation-des-systemes-a-base-dagents/

[49]https://www.ptidej.net/courses/ift6251/fall06/presentations/061030/061030%20-%20Yoursa.pdf

[50] https://theses.hal.science/tel-00203489/preview/these-JT.pdf

[51] https://appmaster.io/fr/blog/quest-ce-que-java-definition-signification-caracteristiques

[52] https://openjfx.io/

[53] https://www.jadeworld.com/jade-platform/about-jade-platform

[54]https://elearning.univmsila.dz/moodle/pluginfile.php/601778/mod_resource/content/1/jade.pdf

[55] https://www.techno-science.net/definition/517.html

[56] https://fr.wikibooks.org/wiki/Programmation_JavaFX/Scene_Builder

[57] https://djug.developpez.com/java/jade/creation-agent/

[58] https://gist.github.com/gyassine/a6689d1d7b5332da561f73401917d605

Annexe A:
Analyse et conception

1. Introduction :

Après avoir abordé les concepts des applications mobiles et des systèmes multi-agents dans les chapitres précédents, ce chapitre se concentre sur les phases d'analyse et de conception de notre système. La première section offre une description fonctionnelle détaillée du système à développer. Ensuite, nous introduisons la méthode VOYELLE et le langage de modélisation AUML (Agent Unified Modeling Language) que nous avons utilisés pour la modélisation. La troisième section identifie les agents et explore les diverses interactions entre eux. Enfin, la partie conception présente différents diagrammes illustrant notre application.

2. Description du système réalisé :

Dans le cadre de ce projet de fin d'études, notre objectif est de développer un système de gestion du trafic routier axé sur l'optimisation de l'encombrement. Ce système repose sur les technologies mobiles et l'intelligence artificielle, et adopte une approche multi-agents.

Dans cette approche, les utilisateurs du système, qu'ils soient conducteurs ou administrateurs responsables du trafic routier, seront représentés par des agents logiciels spécialement conçus. Ces agents logiciels auront la capacité d'accomplir diverses tâches de manière autonome.

L'objectif principal est de fluidifier la circulation sur les routes et de réduire les embouteillages, rendant ainsi la conduite plus agréable.

3. Choix méthodologique :

Pour modéliser notre système, nous avons opté pour la méthode VOYELLE et le langage de modélisation AUML (Voir Chapitre 03).

4. Analyse :

4.1. Identification des utilisateurs :

Dans notre système complexe, nous avons identifié un utilisateur clé : le conducteur, qui joue un rôle spécifique. De plus, nous avons deux utilisateurs secondaires : Google Maps et le GPS.

- **Conducteur** Il s'agit des utilisateurs qui accèdent au système dans le but de rechercher des itinéraires alternatifs pour réduire la congestion routière. Parmi les actions principales effectuées par les conducteurs dans le système, on peut citer :
  - L'envoi de leur position GPS et de la direction.

- La consultation des notifications envoyées par l'agent administrateur (voir la section 4.2).

❖ **Google Maps :** Google Maps est un service de cartographie en ligne développé par Google, offrant des vues de cartes, des images satellites, et des informations de trafic en temps réel. Accessible via navigateur web et applications mobiles, il facilite la navigation, la recherche de lieux d'intérêt, et la planification des trajets pour les conducteurs, piétons, et utilisateurs des transports en commun.

❖ **GPS :** Le GPS permet de déterminer avec précision la position des véhicules sur les routes. Cette information est cruciale pour surveiller le trafic en temps réel et identifier les zones de congestion.

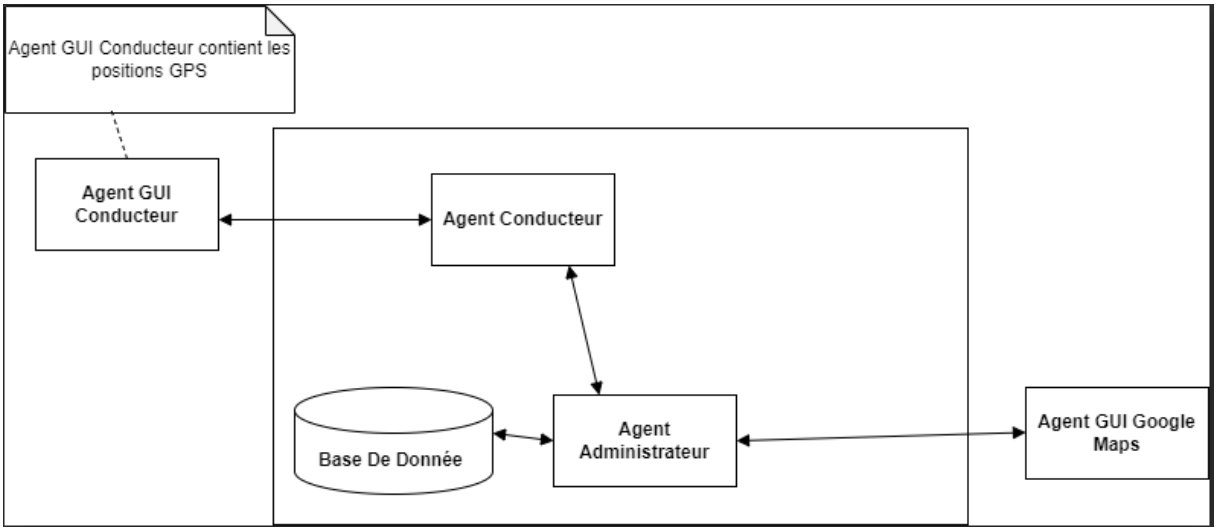

Figure 3.2. Architecture générale du système

4.2. Identification des agents :

La figure 3.2 présente l'architecture générale de notre système, composé de plusieurs agents, chacun ayant un rôle bien défini. Dans la suite de cette section, nous décrirons les différents agents de notre système ainsi que leurs rôles spécifiques.

4.2.1. Les agents système :

❖ **Agent conducteur :** Il représente virtuellement un conducteur dans le système de gestion du trafic routier. L'agent conducteur pourrait être programmé pour accomplir diverses tâches. Il communique avec les autres agents du système pour garantir une conduite sûre et efficace, recevant notamment des informations de l'agent interface conducteur.

- **Agent Administrateur :** il est responsable de la supervision et de la gestion des opérations de système, ainsi que de la création et de la gestion d'autres entités dans le système.

4.2.2. Les agents interfaces :

- **Agent GUI Conducteur :** L'agent interface, fonctionnant sur l'appareil du conducteur, propose une interface qui facilite l'interaction des conducteurs avec le système. Il offre aux conducteurs l'accès à toutes les informations essentielles et leur permet d'accomplir différentes tâches, simplifiant ainsi la communication et la coordination entre le conducteur et le système de manière efficace.
- **Agent GUI Google Maps :** Cet agent agit comme une interface connectée au système, permettant ainsi l'accès à des informations géographiques.

4.3. Identification des interactions :

Après avoir identifié les différents agents composant notre système, nous entamons cette étape en identifiant les interactions entre ces agents.

- **Interaction Agent GUI conducteur / Agent conducteur :** L'Agent Interface Conducteur a la capacité d'envoyer la position GPS actuelle à l'agent conducteur, ainsi que de lui demander de rechercher un itinéraire ou de signaler un incident. L'agent Conducteur répond ensuite en fournissant les résultats du traitement de ces requêtes.
- **Interaction Agent GUI Google Maps / Agent Administrateur :** L'Agent GUI Google Maps peut envoyer des demandes d'informations à l'Agent Administrateur, par exemple pour obtenir des données spécifiques sur les positions des conducteurs. L'Agent Administrateur peut également envoyer des notifications à l'Agent GUI pour définir des itinéraires alternatifs sur la carte.
- **Interaction Agent Conducteur / Agent Administrateur :** L'Agent Conducteur envoie continuellement ses positions GPS à l'Agent Administrateur, qui utilise ces données pour gérer le trafic routier. L'Agent Conducteur transmet également les informations nécessaires pour obtenir un itinéraire alternatif plus rapide, et l'Agent Administrateur répond en fournissant cet itinéraire.

4.4. L'environnement : Dans le contexte des Systèmes Multi-Agents (SMAs), les agents opèrent dans un environnement partagé avec d'autres agents et entités, physiques ou logiques, avec lesquels ils interagissent. Ces agents sont des entités autonomes capables de percevoir,

raisonner, décider et agir de manière indépendante pour influencer leur environnement tout en collaborant avec d'autres agents. Le tableau ci-dessous présente l'environnement spécifique de chaque agent dans notre système :

Agent	**Environnement**
Agent GUI Conducteur	Conducteur + Agent Conducteur
Agent Conducteur	Agent GUI Conducteur + Agent Administrateur
Agent Administrateur	Agent Conducteur + Agent GUI Google Maps + Agent Administrateur
Agent GUI Google Maps	Agent Administrateur

Table 3.1 : L'environnement de chaque agent de notre système.

4.5. L'organisation : L'organisation décrit comment les agents interagissent et sont structurés pour collaborer ou rivaliser, afin d'atteindre les objectifs définis dans leur environnement commun. La structure organisationnelle de notre système est la suivante :

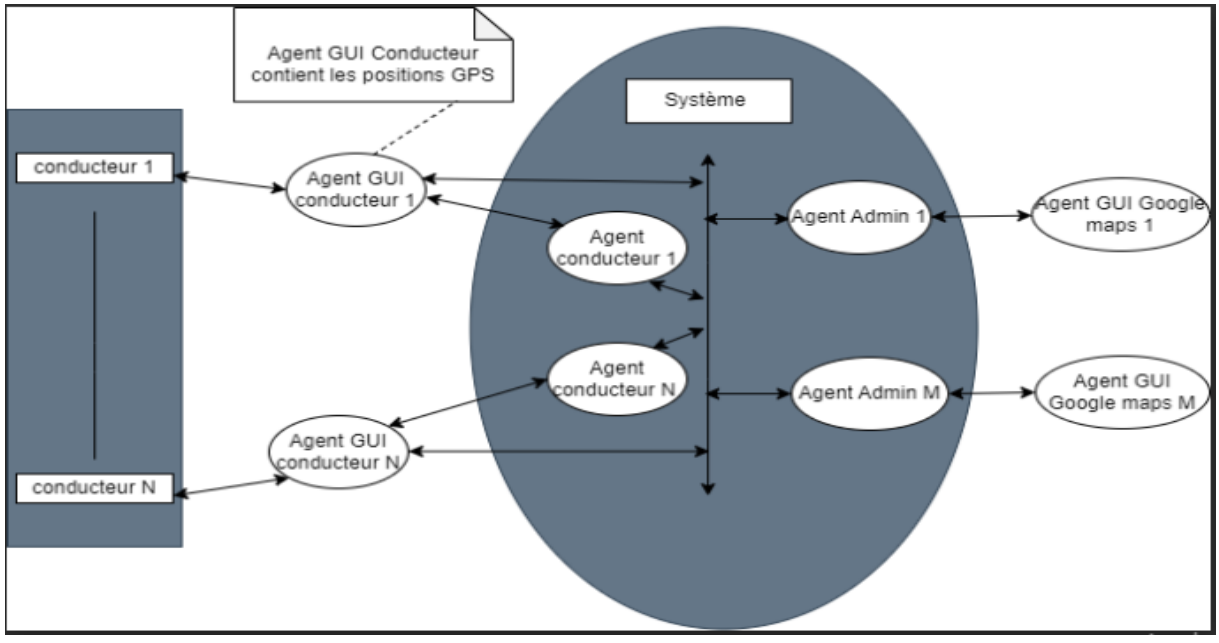

Figure 3.3. Organisation du système.

5. Conception :

- L'approche VOYELLE ne requiert pas de notation spécifique pour la modélisation des systèmes. Par conséquent, nous avons décidé d'utiliser le langage AUML. Notre conception se déroulera comme suit :
- Identification et description des cas d'utilisation, précisant les interactions des utilisateurs avec notre système.
- Modélisation des interactions entre agents à l'aide de diagrammes protocolaires d'interaction.
- Élaboration de diagrammes de classes.

5.1. Identification et description des cas d'utilisations

5.1.1. Diagramme de cas d'utilisation

La figure 3.4 illustre le diagramme de cas d'utilisation global de notre système. Dans ce diagramme, nous avons un acteur principal qui est le conducteur.

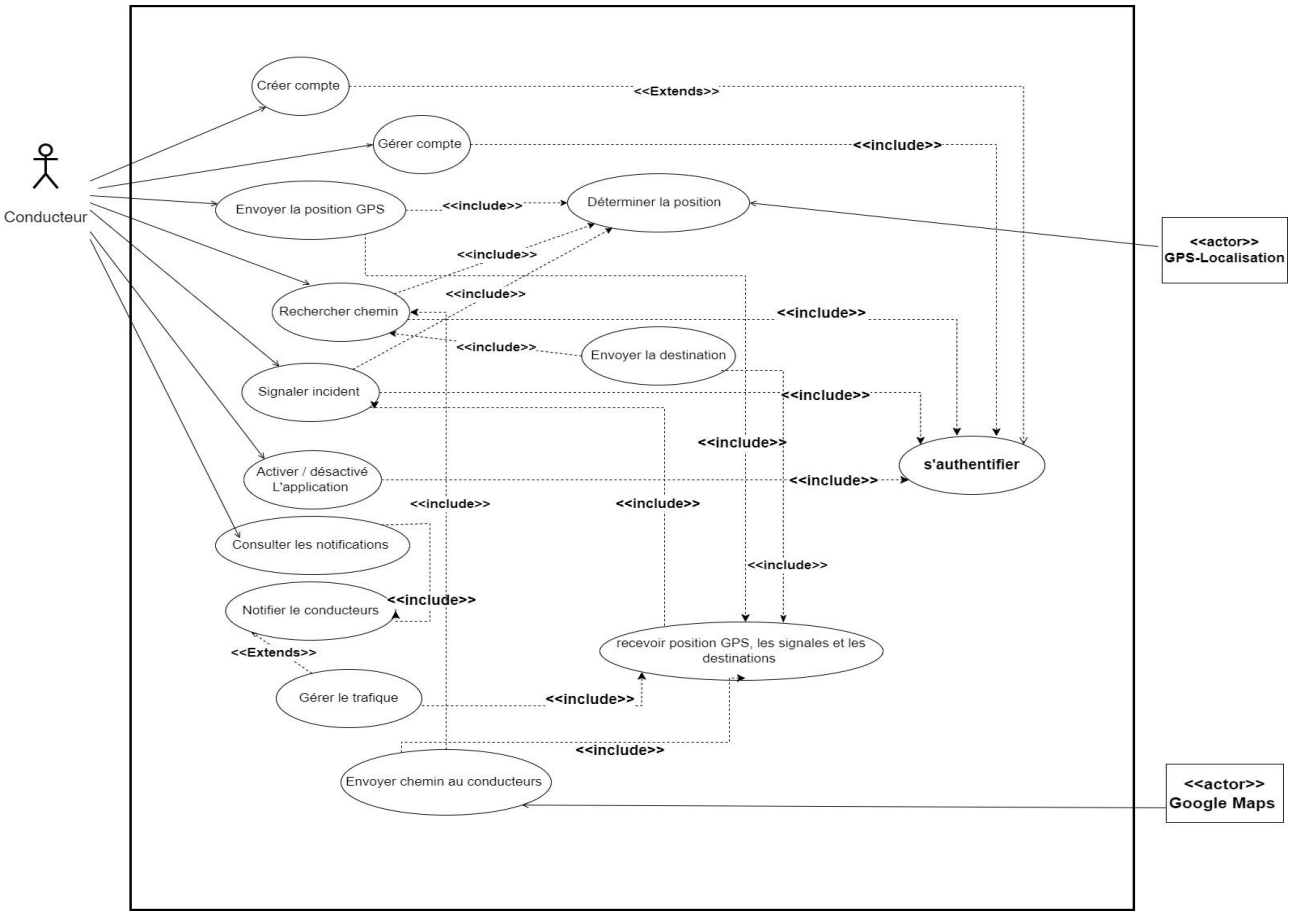

Figure 3.4. Le diagramme de cas d'utilisation

5.1.2. Diagrammes de séquence :

Dans la prochaine partie, nous allons montrer des diagrammes de séquences système qui expliquent comment les choses se passent dans notre système.

5.1.2.1. Diagramme de séquence « s'authentifier » :

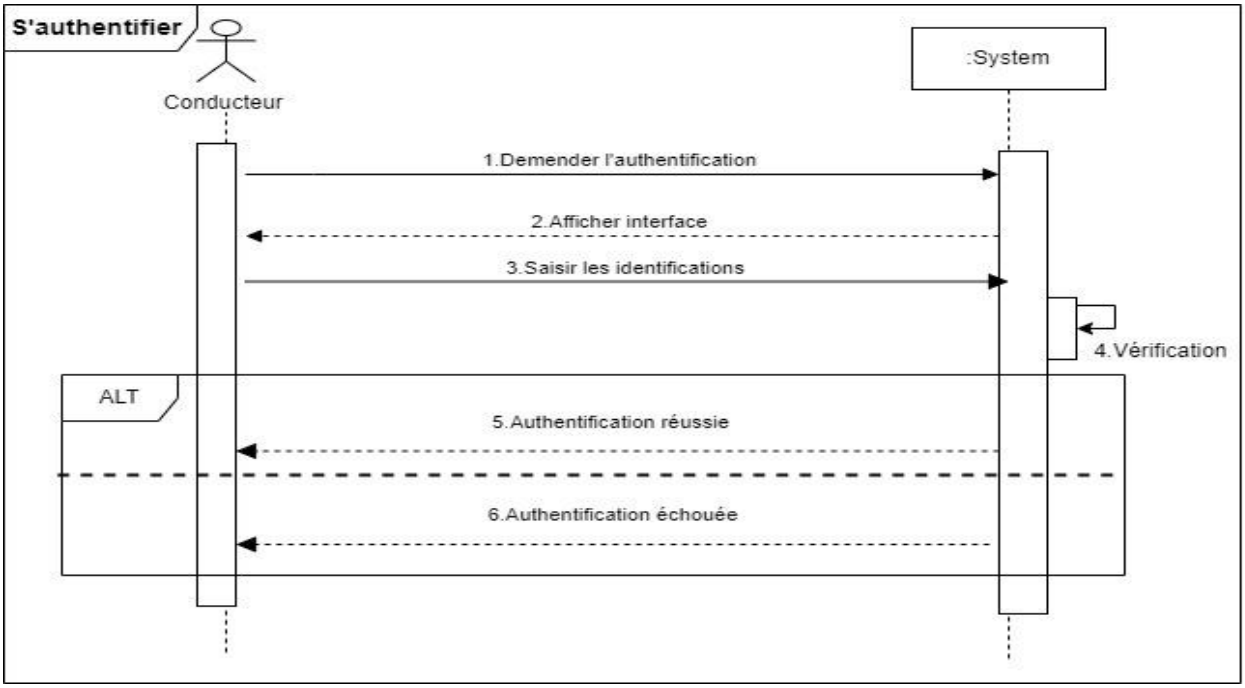

Figure 3.5. Diagramme de séquence « s'authentifier »

5.1.2.2. Diagramme de séquence « déterminer la position » :

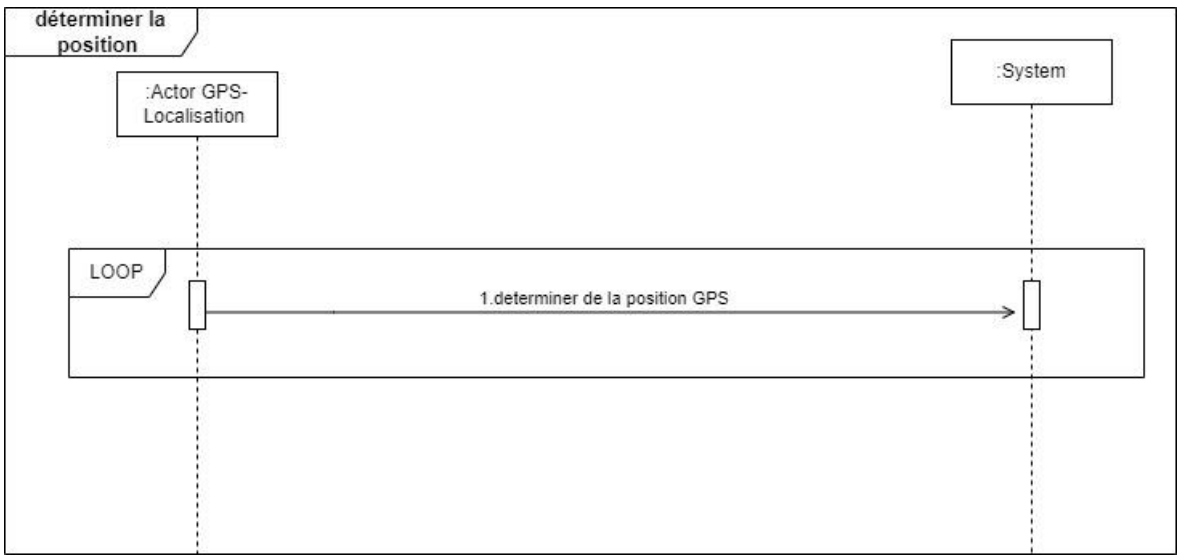

Figure 3.6.Diagramme de séquence « déterminer la position »

5.1.2.3. Diagramme de séquence « rechercher chemin » :

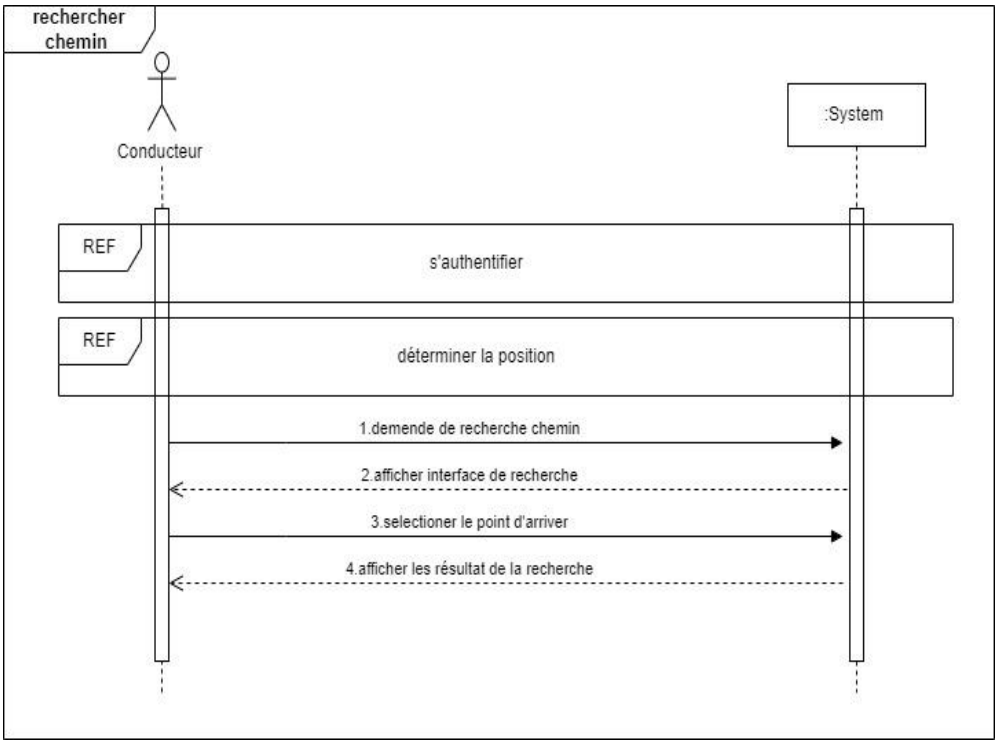

Figure 3.7. Diagramme de séquence « rechercher chemin »

5.1.2.4. Diagramme de séquence « gérer trafic » :

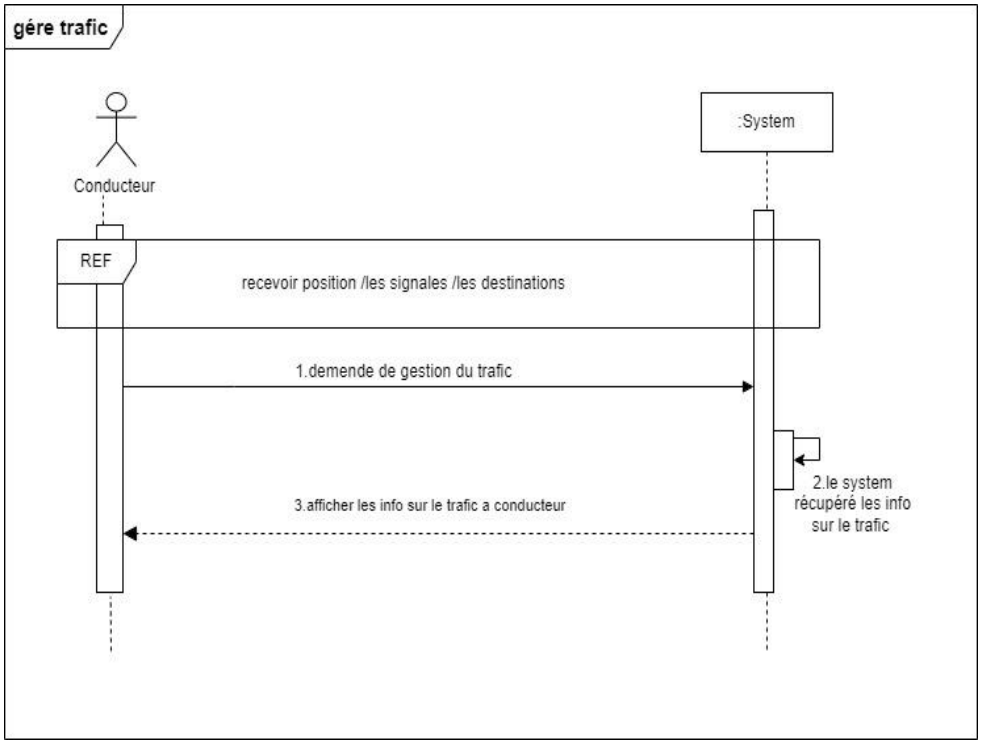

Figure 3.8. Diagramme de séquence « gérer trafic »

5.1.2.5. Diagramme de séquence « notifier conducteur » :

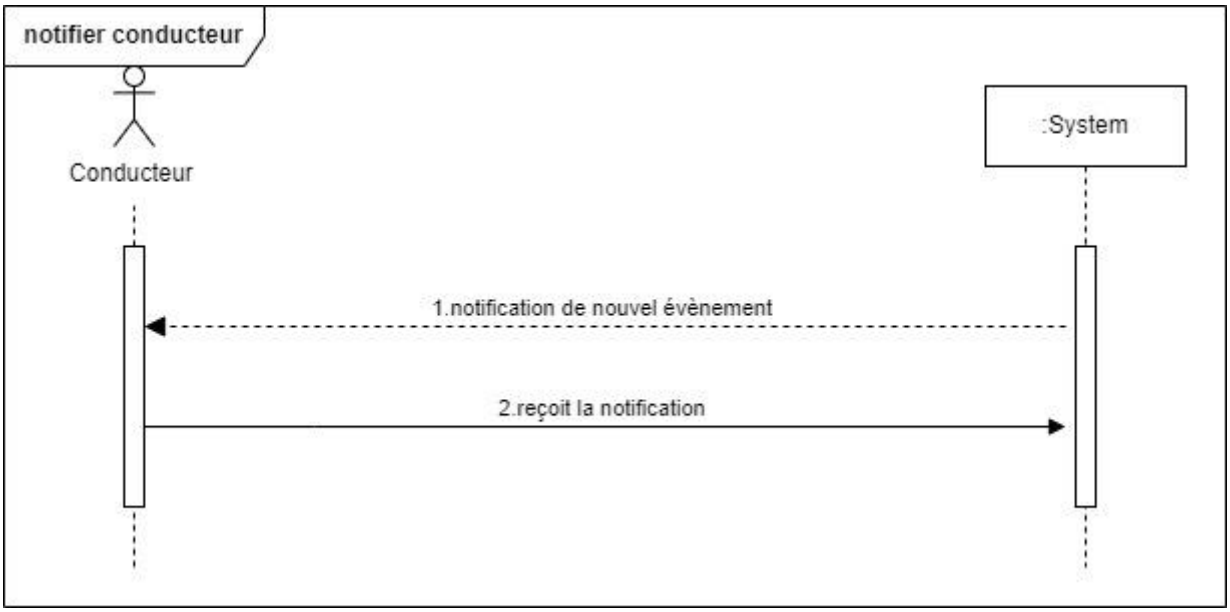

Figure 3.9. Diagramme de séquence « notifier conducteur »

5.1.2.6. Diagramme de séquence « lancer application » :

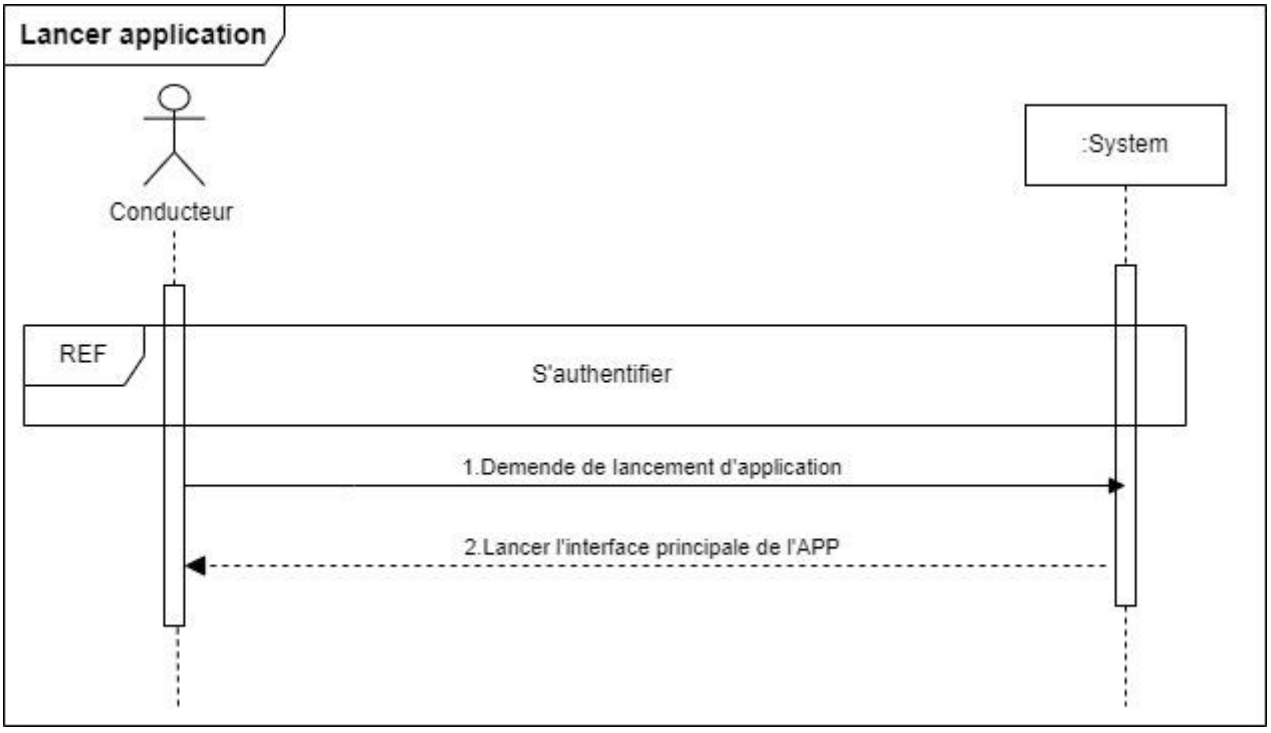

Figure 3.10. Diagramme de séquence « lancer application »

5.1.2.7. Diagramme de séquence « créer compte » :

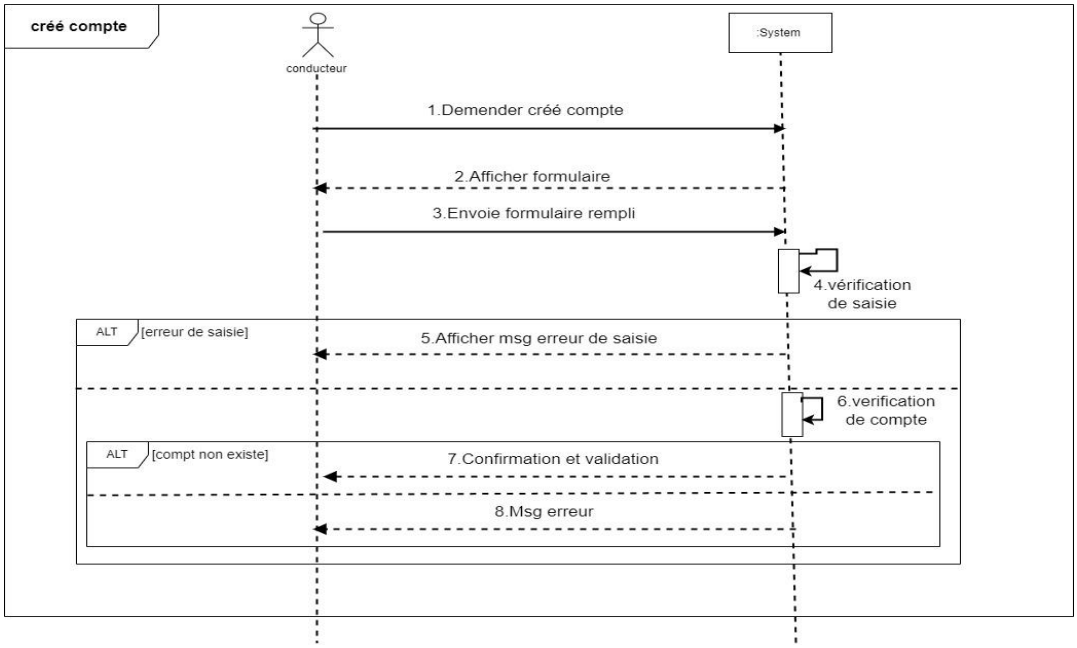

Figure 3.11. Diagramme de séquence « créer compte »

5.1.2.8. Diagramme de séquence « signaler un incident » :

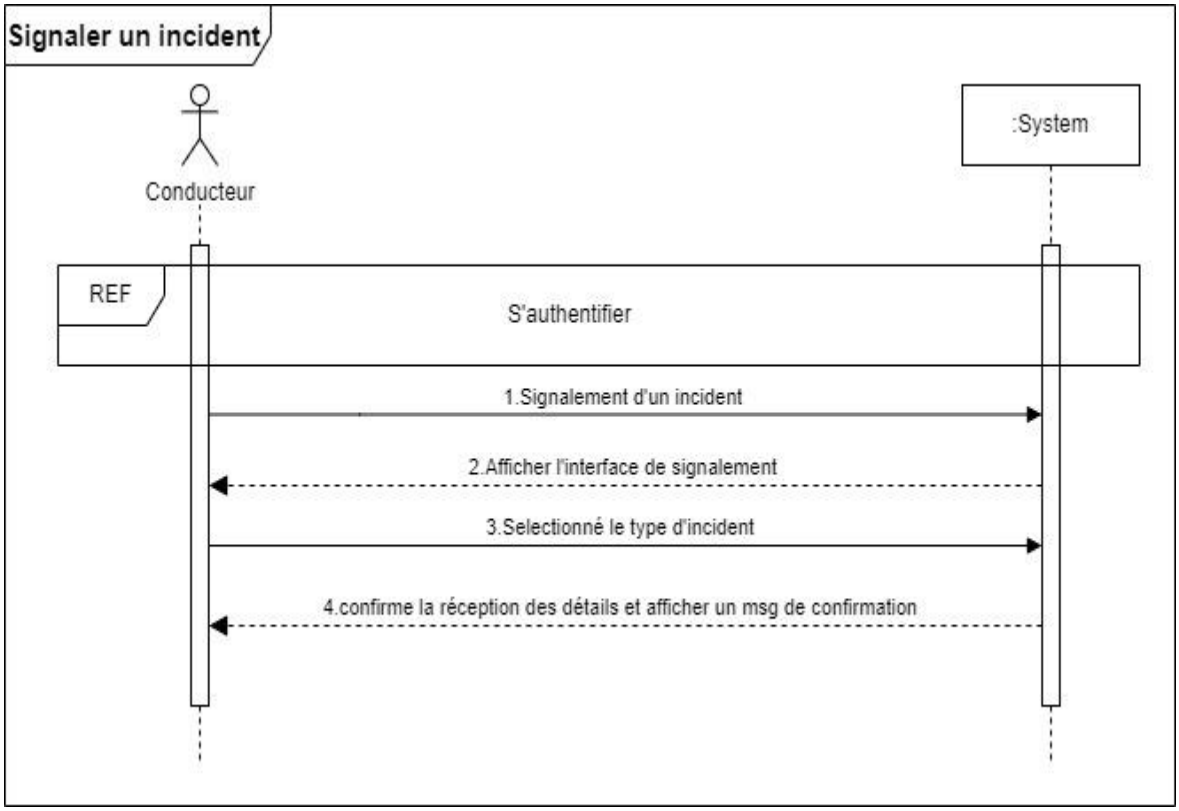

Figure 3.12. Diagramme de séquence « signaler un incident »

5.1.3. Diagramme de protocole d'interactions

5.1.3.1 Diagramme de protocole d'interactions « s'authentifier » :

Dans ce diagramme, l'Agent GUI envoie une demande de connexion contenant toutes ses informations et soumet une requête d'authentification à l'agent administrateur. Celui-ci répond avec une demande de validation pour vérifier la validité des données. Si les données sont valides, l'agent administrateur envoie un message de confirmation, crée un agent conducteur, et établit la connexion. Sinon, il renvoie un message d'erreur.

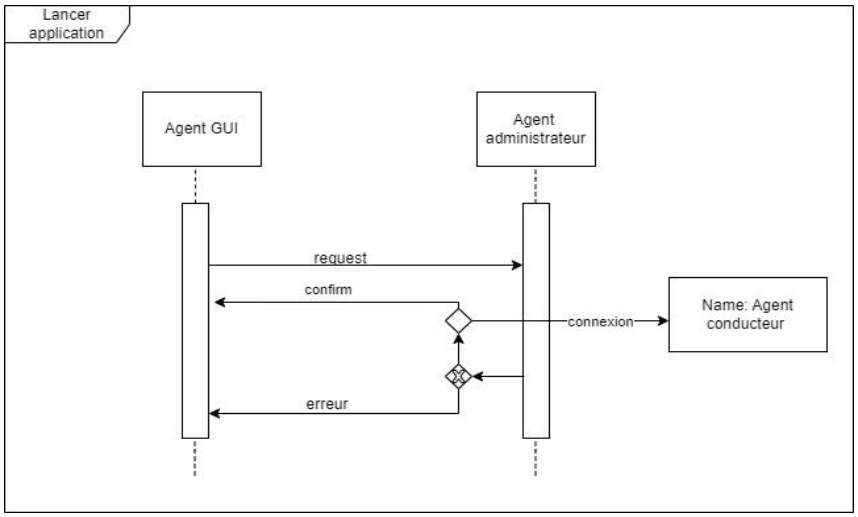

Figure 3.13. Diagramme de protocole de s'authentification

5.1.3.2. Diagramme de protocole d'interactions « rechercher chemin » :

Dans ce diagramme, l'agent GUI envoie une requête de recherche de chemin à son agent conducteur. L'agent conducteur reçoit la requête et la transmet à l'agent administrateur. Ce dernier traite la requête et choisit le chemin, puis envoie cette information à l'agent Google Maps pour qu'il dessine le chemin sur la carte. Ensuite, l'agent Google Maps transmet le chemin dessiné à l'agent administrateur, qui le transmet à l'agent conducteur. Enfin, l'agent conducteur envoie le chemin à son agent GUI.

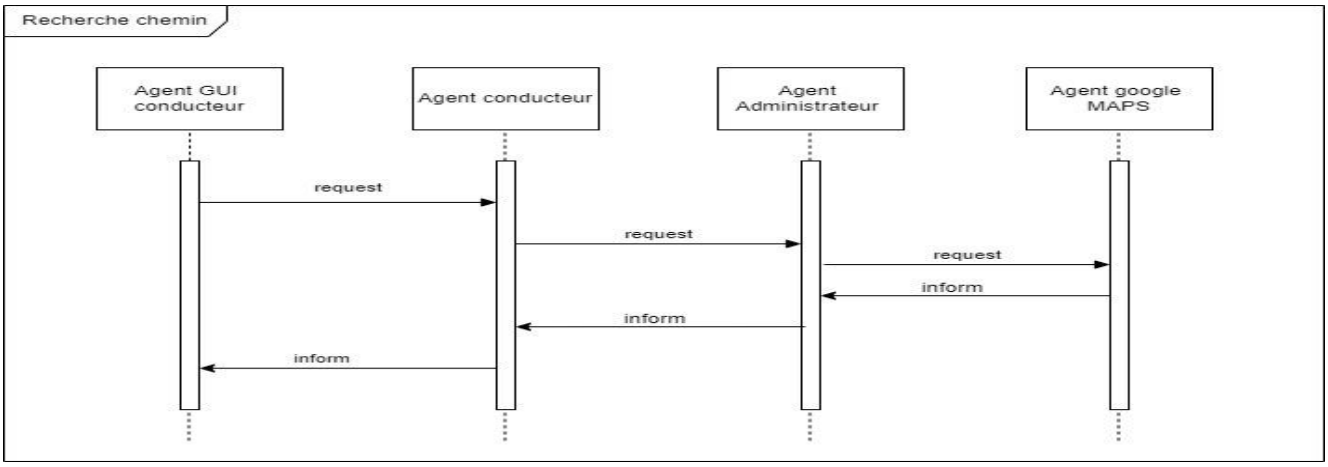

Figure 3.14. Diagramme de protocole d'interactions « rechercher chemin »

5.1.3.3. Diagramme de protocole d'interactions « signaler incident » :

Dans ce diagramme, l'agent GUI envoie une requête de signalement d'incident à l'agent conducteur. L'agent conducteur reçoit la requête et notifie l'agent administrateur du type d'incident. L'agent administrateur traite l'incident et, une fois le traitement terminé, envoie une confirmation à l'agent conducteur via un message INFORM. L'agent conducteur reçoit la confirmation et informe ensuite l'agent GUI de la situation.

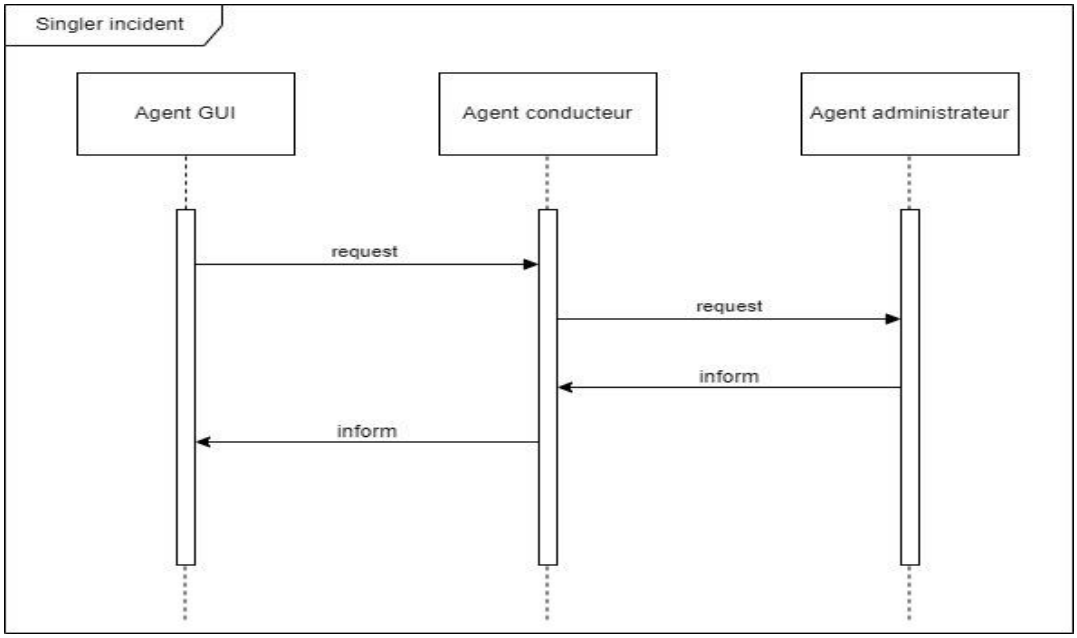

Figure 3.15. Diagramme de protocole d'interactions « signaler incident »

5.1.3.4. Diagramme de protocole d'interactions « gérer trafic » :

Dans ce diagramme, l'agent administrateur envoie une requête incluant des décisions à plusieurs agents conducteurs. Chaque agent conducteur reçoit la requête et suit les instructions reçues en réponse à travers un message INFORM.

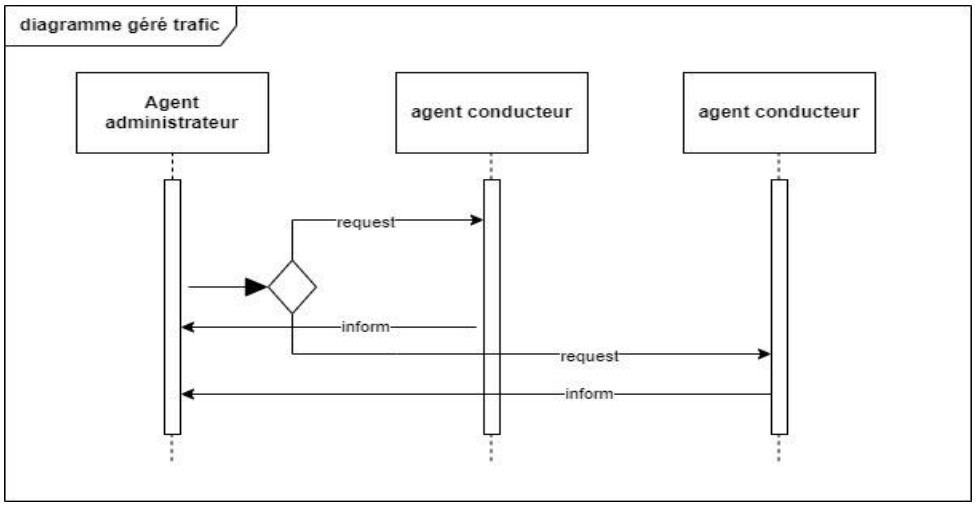

Figure 3.16. Diagramme de protocole d'interactions « gérer trafic »

5.1.3.5. Diagramme de protocole d'interactions « déterminer position » :

Dans ce diagramme, l'agent GUI envoie une requête pour déterminer la position (obtenue à partir de la localisation GPS) à son agent conducteur. L'agent conducteur reçoit la requête et transmet cette position à l'agent administrateur. Ce dernier traite la requête et gère le trafic.

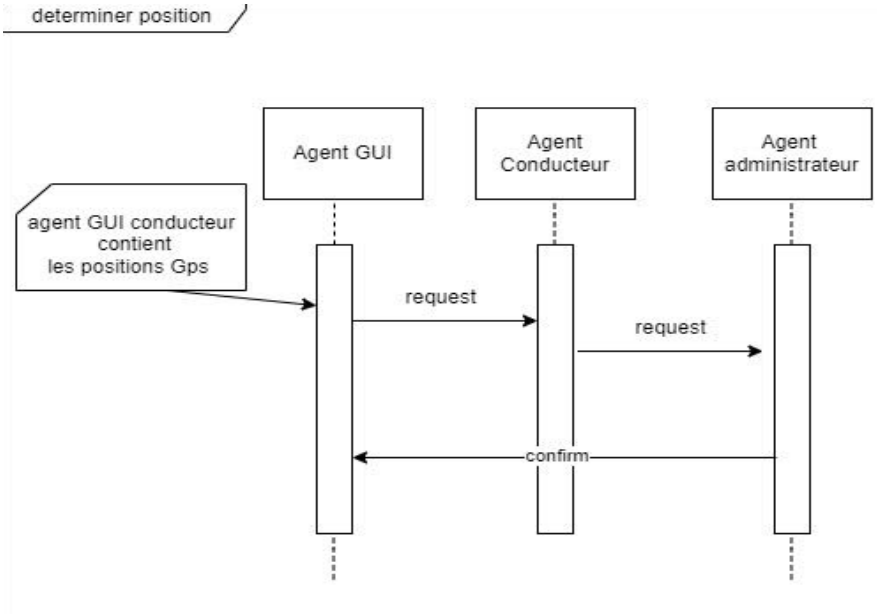

Figure 3.17. Diagramme de protocole d'interactions « déterminer position »

5.1.4. Diagramme de classes :

Dans cette section, nous allons élaborer des diagrammes pour illustrer l'organisation de notre système. Tout d'abord, nous créerons un diagramme représentant les différentes composantes de notre système. Ensuite, nous établirons un autre schéma pour visualiser les différentes classes d'agent qui interagissent avec notre système.

5.1.4.1. Diagramme de classes du système :

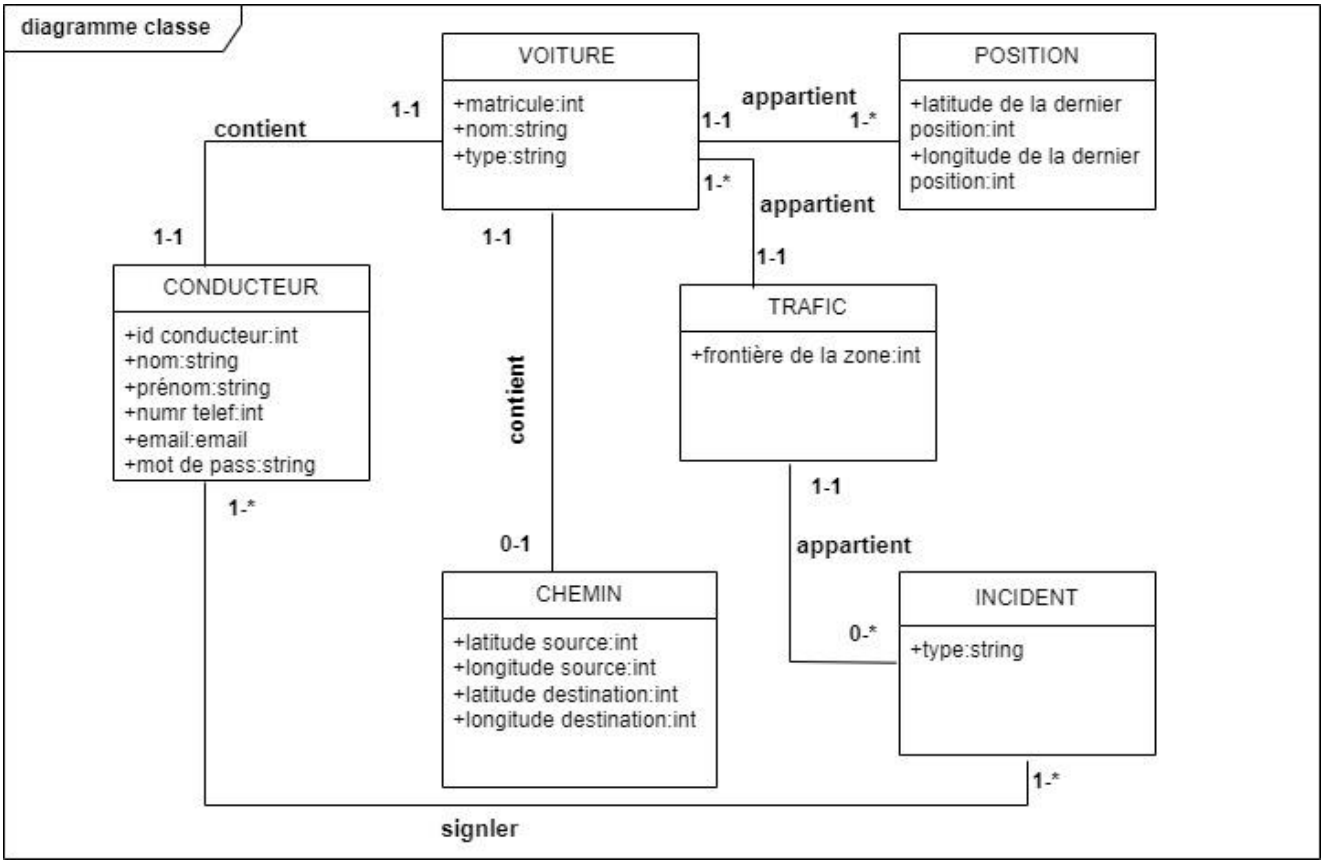

Figure 3.18. Diagramme de classes du système

5.1.4.2. Diagramme de classes Agent :

Voici le diagramme de classes des agents de notre système. Ce diagramme montre quatre classes d'agents qui héritent de la classe principale "Agent".

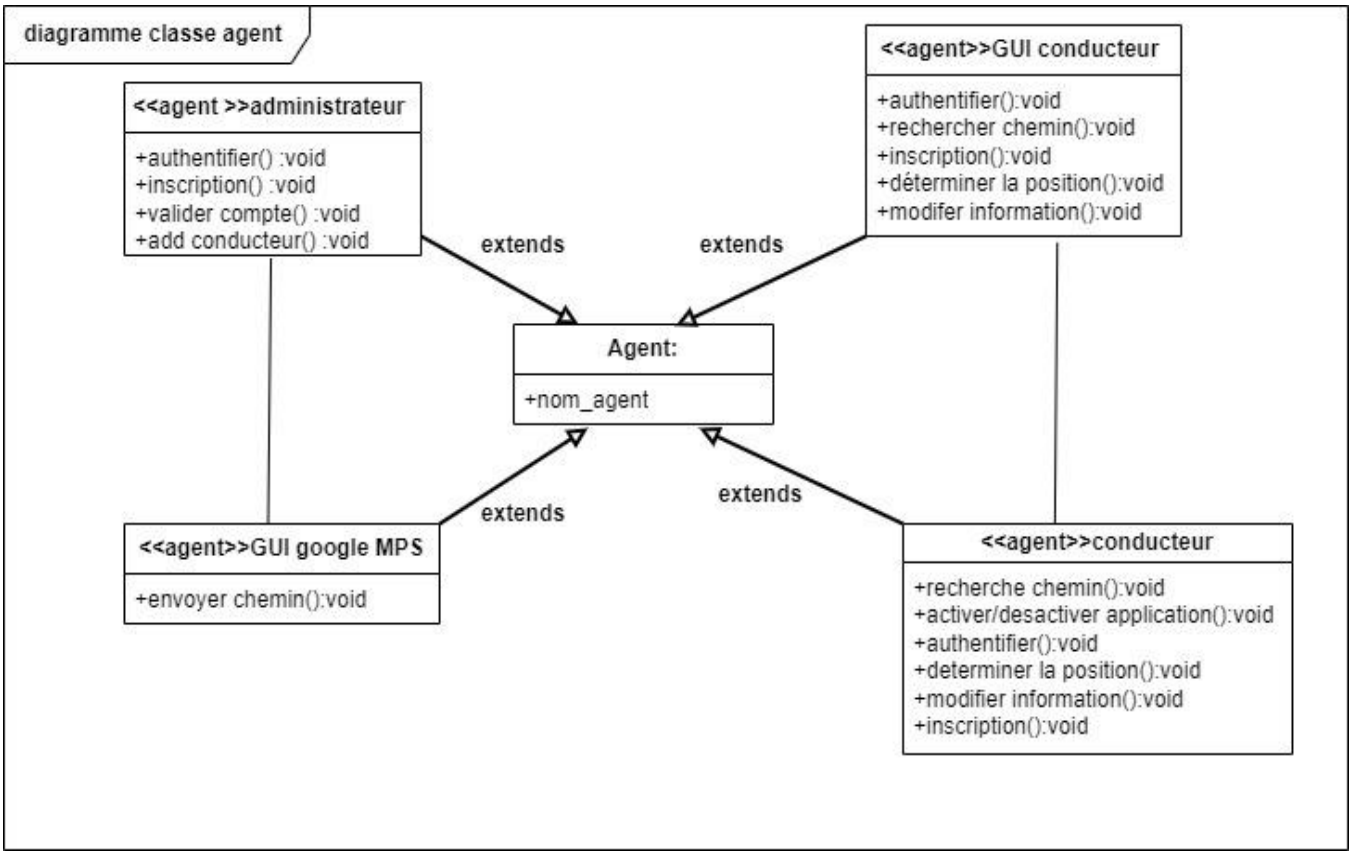

Figure 3.19. Diagramme de classe Agent

Les figures de 3-20 au 3-23 donnent une représentation détaillée de différentes classes d'agents de notre système.

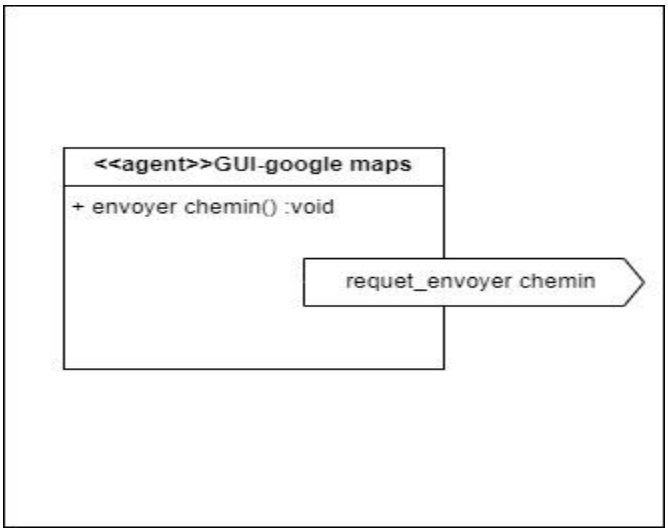

Figure 3.20. Agent GUI-Google Maps

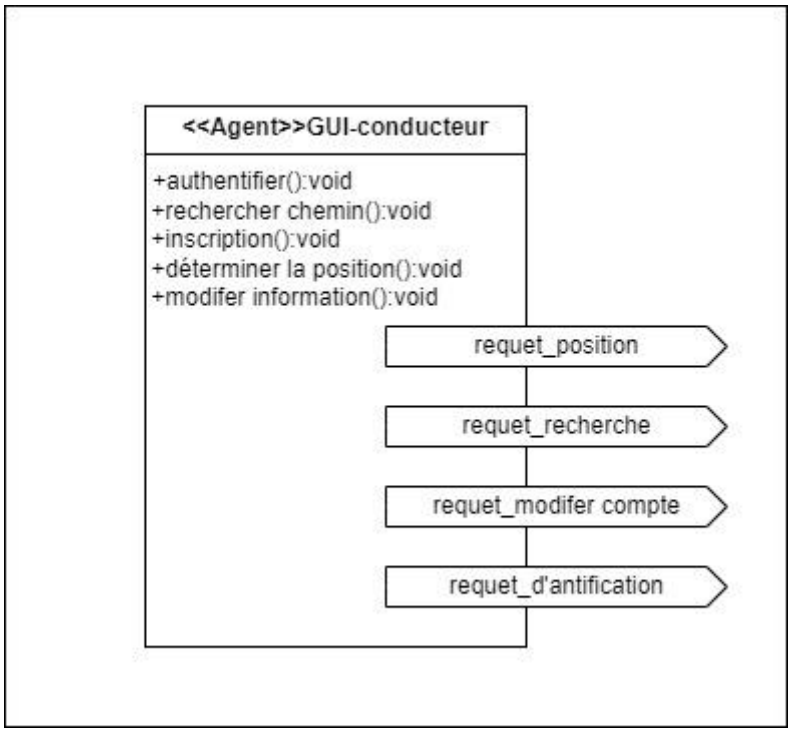

Figure 3.21. Agent GUI-conducteur

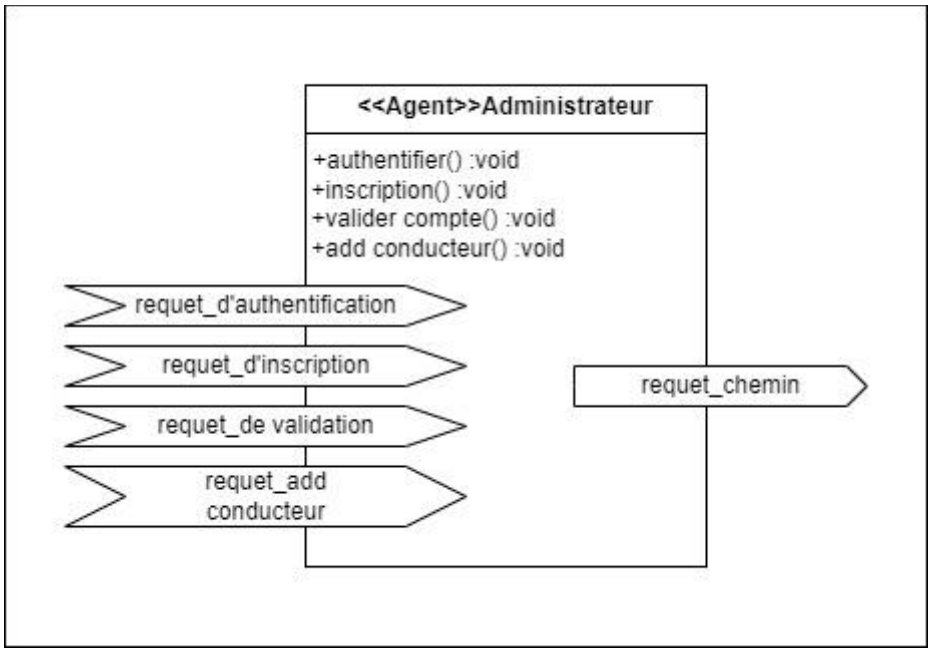

Figure 3.22. Agent GUI-administrateur

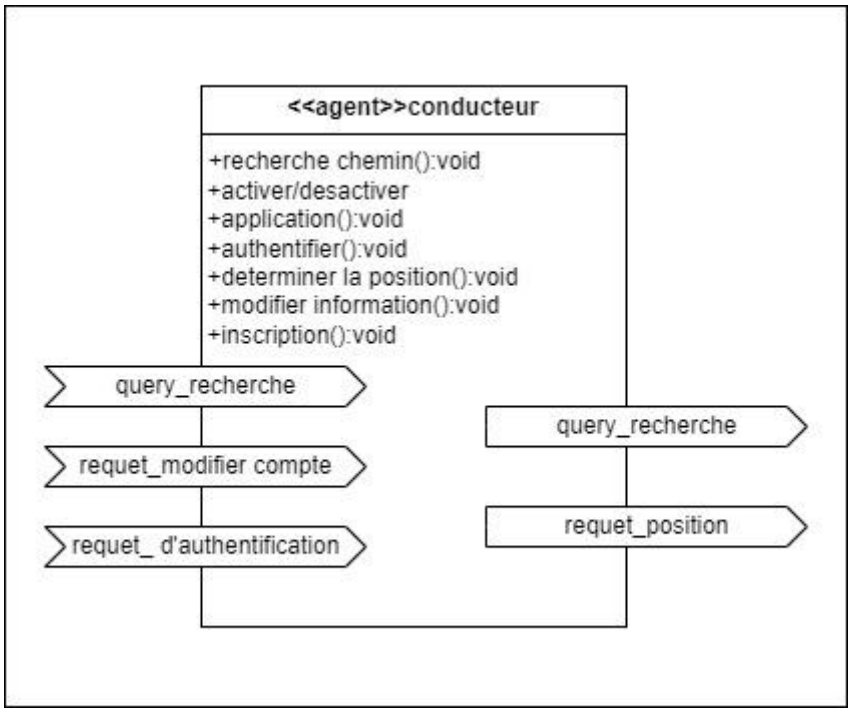

Figure 3.23. Agent conducteur

6. Conclusion :

Dans cette section, nous avons détaillé notre processus de planification et de conception du système en utilisant la méthodologie Voyelle ainsi que le langage AUML. Initialement, en suivant la méthodologie Voyelle, nous avons identifié les agents impliqués dans le système, sa structure globale, ainsi que l'environnement spécifique de chaque agent. Par la suite, une fois les différentes composantes du système identifiées, nous avons élaboré des diagrammes AUML pour le représenter. Ces diagrammes incluent le diagramme de cas d'utilisation, le diagramme de séquence, et le diagramme de protocole d'interactions, décrivant les interactions entre les différentes parties du système. De plus, nous avons présenté le diagramme de classe d'agent, mettant en avant l'aspect statique de notre système.