# Une méthode incrémentale de conception dirigée par les tests pour la simulation multi-agent

Lancelot SIX[a,c]
lancelot.six@quiet-oceans.com

Julien SAUNIER[b]
julien.saunier@insa-rouen.fr

Zahia GUESSOUM[c]
zahia.guessoum@lip6.fr

Sio-Song IENG[a]
sio-song.ieng@ifsttar.fr

[a] IFSTTAR — 14-20 Boulevard Newton — Cité Descartes, Champs sur Marne — F-77447 Marne la Vallée Cedex 2

[b] INSA Rouen —Avenue de l'Université —76801 Saint-Étienne-du-Rouvray Cedex

[c] Lip6 — 4 place Jussieu — 75005 Paris

**Résumé**

*L'approche multi-agent est par nature adaptée à une conception incrémentale des modèles. La modularité de l'approche permet la conception progressive des éléments du système cible, par l'ajout de nouvelles entités, de nouveaux modes d'interaction et d'organisation, ou l'inclusion de nouveaux comportements. Cependant, les méthodes de conception de logiciels usuelles ne sont en général pas applicables dans leur ensemble au développement des systèmes de simulation à base d'agents, la principale difficulté étant l'émergence de comportements collectifs. Dans cet article, nous proposons une méthode de conception dirigée par les tests adaptée aux systèmes multi-agents, inspirée à la fois des modèles en spirale et de la conception dirigée par les tests. L'ajout d'une fonction est spécifiée au niveau du système, tandis que l'analyse et la conception se fonde sur la vérification des propriétés individuelles. Nous illustrons cette méthode avec un cycle de conception d'un modèle de poids-lourd dans une simulation de trafic.*

***Mots-clés :*** *Conception incrémentale, Simulation de trafic, Modélisation*

**Abstract**

*Multi-agent systems are well suited to incrementally develop models. Their modularity allow to develop the system by gradually adding new interaction or organization models, new entities, or new behaviors. However, usual software engineering methods cannot be fully used to develop multi-agent systems due to their lack in considering un-anticipated emergent phenomena. In this article, we propose a test-driven inspired incremental development method tailored for agent based simulation models. We illustrate the use of this method to improve a heavy vehicle model to be used in traffic simulations.*

***Keywords:*** *Incremental development, Traffic simulation, Modeling*

## 1 Introduction

La simulation multi-agent fournit un outil puissant pour la reproduction et l'étude de comportements humains sociaux. Elle a été appliquée avec succès aux systèmes complexes tels que l'économie, les réseaux sociaux ou le transport. Dans ces différentes applications, les agents possèdent des comportements complexes et difficiles à spécifier, puisque ceux-ci sont à la fois réactifs et cognitifs. De plus, les modèles à base d'agents sont généralement conçus pour étudier des propriétés macroscopiques émergentes, qui sont le résultat des interactions entre agents.

Dans cette perspective, créer un modèle d'agent est une tâche non triviale, car les sorties du modèle (les phénomènes émergents) ne sont pas directement spécifiées dans le modèle initial [6]. De plus, quand un modèle produit un phénomène émergent adéquat, il n'est souvent pas évident de savoir laquelle des hypothèses de modélisation en est responsable [11]. La simulation de trafic illustre ces difficultés. Le principal objectif du modélisateur se situe sur la qualité au niveau macroscopique, de façon à comprendre et prédire les flux et les optimiser. Depuis la fin des années 50, de nombreux modèles ont été proposés et validés de cette façon [3, 14, 15].

De façon à faciliter la conception de modèles à base d'agents, nous proposons de nous inspirer d'une approche à base de tests. Nous appliquons cette méthode à la conception d'un modèle de véhicules lourds, lesquels sont pour l'instant rarement pris en compte dans les simulations de trafic, malgré un impact fort sur

les flux de véhicules [4]. Le modèle IDM [25] est une illustration de ce problème : les camions sont seulement considérés comme des véhicules longs capables de faibles accélérations et décélérations. Cependant, d'autres facteurs différencient les poids-lourds des véhicules légers telle qu'une dynamique changeante en fonction notamment de la charge et de la déclivité, ce qui a un impact sur les stratégies de conduite (et notamment d'anticipation).

L'approche agent est particulièrement adaptée à une conception incrémentale des modèles. Un cycle de développement incrémental permet de bâtir la solution logicielle par étapes, en ajoutant une ou des fonctionnalités (incréments) à chacune de ces étapes. Une partie des modèles multi-agents tire avantage de cette propriété et sont construits sur la base d'un modèle existant, en se concentrant sur le développement d'un aspect qui intéresse le modélisateur. Un exemple est le modèle HDM [17], qui est une sur-couche sur IDM [25] se concentrant sur les aspects stratégiques de la conduite. Les bases offertes par le modèle d'origine (capacité de suivi de véhicule) ne sont pas remises en cause et le modèle est enrichi.

Cependant, les méthodes de conception de solutions logicielles usuelles ne sont en général pas applicables dans leur ensemble au développement des systèmes de simulation à base d'agents de par leur nature, puisque l'émergence des phénomènes macroscopiques ne peut être directement spécifiée. Dans la section suivante, nous précisons les motivations de notre approche de conception, puis décrivons des méthodes de conception usuelles. En section 3, nous proposons notre méthode de conception dirigée par les tests, et en illustrons une itération dans la section 4. Enfin, nous concluons et proposons quelques perspectives en section 5.

## 2 Motivation et état de l'art

Le premier et principal point de difficulté de l'utilisation de méthodes de génie logiciel pour la conception de systèmes multi-agents est lié au principe d'émergence qui est recherché par de tels systèmes. Cette approche postule que la modélisation des comportements individuels des composants du système est suffisante pour faire émerger le comportement du tout. Or, les processus de développement usuels sont dirigés par les buts (spécification du système dans son ensemble). Le développement incrémental est fait pour ajouter une fonctionnalité attendue au logiciel. Ces buts correspondent généralement à un besoin applicatif, par exemple produire un simulateur permettant de reproduire une ou des propriétés d'un objet d'étude et répondre à une question sur cet objet d'étude. Il y a donc une incompatibilité entre une approche où le développement est orienté par le résultat et un développement où le résultat n'est pas nécessairement anticipé par le modélisateur.

Par ailleurs, Galan et al. [11] soulignent que l'interaction de nombreux modèles - qu'ils concernent les agents, l'environnement, les communications ou encore la plateforme elle-même - entraîne un niveau de complexité accru. En effet, afin d'aboutir à une implémentation fonctionnelle de chacun de ces composants, de nombreux choix doivent être pris *a priori*. Par exemple, une discrétisation de l'environnement spatial peut être nécessaire à la réalisation de la plateforme, et ce choix impactera nécessairement l'implémentation des agents évoluant dans ce monde, ainsi que les résultats de simulations. Cette difficulté d'exprimer l'ensemble des décisions nécessaires à la réalisation et éventuellement à la reproduction d'un système à base d'agents est la principale motivation des travaux de Chevrier et Fatès [5]. Ils proposent une méthode de formalisation de systèmes à base d'agents visant à ne pas permettre la présence de choix implicites, qui empêcheraient l'appropriation et l'analyse du modèle par une tierce partie (utilisateur du modèle, relecteur apportant un regard critique sur la proposition…).

Enfin, le processus de modélisation implique différent acteurs. Trois acteurs jouent des rôles importants selon [8] : le *thématicien* est l'acteur apportant une expertise sur le système modélisé et définissant les besoins applicatifs. Ceux-ci sont généralement exprimés de manière non formelle et peuvent donc contenir des ambiguïtés et des imprécisions.Le *modélisateur* est l'acteur qui, à partir du modèle non formel fourni par l'expert thématique, produit une version formelle du modèle, à l'aide d'un langage de modélisation. Enfin, l'*implémenteur* est l'acteur qui produit une version exécutable du modèle produit par le modélisateur. L'interaction entre les différents acteurs, chacun disposant de méthodes et connaissances propres, peut introduire un grand nombre d'approximations. Le mode d'interaction entre ces différents acteurs n'est par ailleurs pas guidé par les méthodes de développement dont nous avons connaissance.

Dans l'objectif d'aboutir à un modèle de dé-

veloppement incrémental [1] tenant compte des spécificités ainsi que des difficultés intrinsèquement liées aux systèmes de simulation à base d'agents, nous proposons une méthode dirigée par la vérification. Nous présentons donc dans un premier temps deux approches de développement répandues dont nous nous inspirons qui sont 1) le modèle en spirale et 2) le développement dirigé par les tests.

**Le développement en spirale.** La première approche que nous abordons est le développement en spirale proposé par [2]. Alors que dans la plupart des méthodes antérieures le développement était centré sur le code fourni, cette approche est focalisée sur le risque. Pour chaque itération, l'évaluation des changements envisagés, de leurs alternatives, et des risques qui leurs sont associés est plus importante que l'amélioration elle-même.

Pour ce modèle de développement, chaque itération se décompose en quatre phases (ou cadrants) : *(1) Description des objectifs*- la première étape consiste à évaluer les objectifs de l'itération et à s'interroger sur leur pertinence. De manière similaire à ce qui est proposé par les méthodes agiles, il s'agit de pouvoir s'adapter à des changements pouvant survenir dans les besoins. *(2) Analyse des risques*- il s'agit du point central du cycle en spirale. Chaque itération comporte une phase d'évaluation des risques associés aux objectifs (risque de dépassement de délais, risque de blocage technique, *etc.*). L'objectif est de ne continuer l'itération que si les risques sont suffisamment contrôlés. Il s'agit donc d'un point de décision important permettant de garder la maîtrise des développements effectués soit en préférant une alternative moins risquée, soit en abandonnant l'itération courante. *(3) Développement*- il s'agit de l'activité usuelle de production du logiciel ainsi que des différentes phases de tests. *(4) Préparation de la phase suivante*- enfin, la dernière étape de l'itération consiste à planifier la suite des itérations.

Contrairement aux cycles de développement classiques qui sont centrés soit sur la délivrance de documentations (comme c'est le cas pour le développement en cascade) soit sur la délivrance de codes (comme c'est le cas pour les méthodes agiles), le développement en spirale se concentre sur l'analyse des risques et de la pertinence des travaux à effectuer. Ainsi, ce modèle propose différents points de décision permettant d'orienter ou réorienter les évolutions à venir selon leur pertinence.

**Développement dirigé par les tests.** L'autre famille de méthodes dont nous nous inspirons est le développement dirigé par les tests, particulièrement utilisé dans les méthodes agiles [23]. Les méthodes agiles sont des méthodes de développement incrémentales. Elles se caractérisent par des cycles très courts ayant pour objectif de pouvoir réagir très rapidement à un changement du besoin et de minimiser le temps nécessaire à fournir une version exploitable du logiciel. Le développement d'une solution est décomposé en un ensemble de fonctionnalités, chacune d'elles faisant l'objet d'une itération. L'ordre des itérations dépend de la priorité accordée à chaque fonctionnalité ainsi qu'au coût de développement associé.

Une partie de ces méthodes (dont *«eXtreme Programming»* (XP)[1], pour ne citer qu'un exemple) se fonde sur un développement dirigé par les tests. Il s'agit d'écrire les tests auxquels sera soumis le code produit avant même que le code ne soit lui-même écrit. Une fois les tests écrits, l'objectif est de produire une implémentation qui permette de satisfaire l'ensemble des tests. Cette approche va à l'opposé de ce qui est généralement pratiqué, à savoir écrire un code et ensuite la série de tests permettant de s'assurer que le code est conforme aux spécifications.

L'écriture *a priori* des tests a plusieurs avantages. Tout d'abord, elle est une façon d'expliciter et de mettre à l'épreuve la spécification du module. Cela permet de se concentrer sur l'interface du code à produire. Cette première étape permet donc de détecter au plus tôt certaines erreurs de conception qui aboutiraient à une interface incomplète ou difficilement utilisable. De plus, une fois les tests écrits, l'objectif est de produire une implémentation telle que l'ensemble des tests soient satisfaits. Ceci incite donc à la concision et évite de produire des fonctionnalités non nécessaires.

L'ensemble des tests produits pour une itération peuvent être utilisés pour effectuer des tests de non régression qui seront utilisés dans deux cas : lors d'un éventuel remaniement du code (*refactoring*) afin d'en faciliter la maintenance dans le futur, ou lors du développement de nouvelles

1. L'approche incrémentale est à distinguer des approches itératives. Dans une approche incrémentale, chaque cycle de développement a pour objet l'ajout ou la modification d'une fonctionnalité alors que dans un processus itératif, une itération correspond au déroulement d'un cycle de développement comprenant analyse, conception, implantation et tests, sans pour autant nécessairement modifier le comportement du logiciel produit.

fonctionnalités afin de vérifier que les modifications apportées ne dégradent pas l'existant.

# 3 Amélioration de modèles d'agents dirigée par les tests : TIM4MAS

En s'inspirant des deux méthodes précédentes, nous présentons dans cette section notre modèle de développement de systèmes à base d'agents qui adapte ces deux modèles à une utilisation dans le cadre de systèmes multi-agents : *Test driven Incremental development Method for Multi Agent Systems* (TIM4MAS). Cette méthode est adaptée à l'amélioration de modèles existants, il est préférable de se référer aux méthodes de conception multi-agent classiques (voir section 5) pour une conception *ex-nihilo* de systèmes.

Le processus de développement dirigé par la vérification repose sur cinq phases :

- L'identification et la caractérisation d'une propriété du comportement qui doit être reproduite. Cette première phase consiste à identifier un aspect du comportement des agents à modéliser.
- L'évaluation de l'impact attendu de l'aspect du comportement sur le déroulement de la simulation.
- La spécification des propriétés de cette facette à l'aide d'un ensemble de tests.
- L'implémentation. Cette phase consiste à proposer une implémentation d'un modèle (ou un ensemble de modifications d'un modèle existant) telle qu'elle permette de reproduire les propriétés attendues (au niveau microscopique).
- La planification des phases suivantes, en fonction des résultats de validation (au niveau macroscopique).

La figure 1 montre une adaptation du cycle en spirale adapté au développement de modèles d'agents. Les trois premiers cadrants correspondent à la phase de spécification des besoins. Le rendu de cette première grande phase est la spécification des tests que l'implémentation devra par la suite être en mesure de passer. La phase suivante correspond au développement proprement dit. Enfin, la dernière correspond au point de planification de la suite du projet.

## 3.1 Identification d'une facette de comportement

La première étape du développement consiste à identifier une facette $f$ du comportement

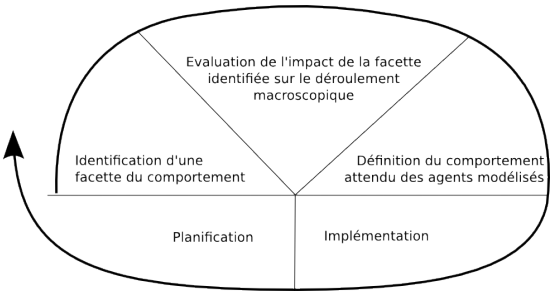


FIGURE 1 – Déroulement général d'une itération de développement.

| | | |
|---|---|---|
| Rôle | expert thématique | L'expert thématique identifie la facette sur laquelle l'itération sera centrée. |
| Entrées | $M_{SMA} = \{m_1, ...m_n\}$ | modèle de SMA à l'itération courante, composé de d'un ensemble de modèles d'agents et d'un modèle d'environnement |
| Sorties | $f$ | facette du comportement |

d'agent(s) devant être améliorée. Une facette du comportement est un aspect de l'action d'un agent dans une situation donnée. Par exemple, la stratégie de décélération d'un véhicule à l'approche d'une zone de congestion ainsi que sa procédure de changement de voie sont des facettes du comportement d'un conducteur.

Une facette peut également être un mécanisme interactionnel entre plusieurs agents.

L'identification et la caractérisation de ces facettes est une tâche qui relève du domaine de l'expert thématique (selon les rôles décrits par [8]). Il est en charge d'identifier où il estime que des améliorations sont opportunes.

## 3.2 Évaluation de l'impact attendu

Une fois l'identification de la facette achevée, il faut évaluer si les modifications envisagées sont pertinentes du point de vue du système modélisé, et si elles peuvent potentiellement modifier son fonctionnement. Une modification n'ayant pas d'impact significatif sur le comportement du système ne représente pas un objectif prioritaire en terme de développement du modèle. De cette analyse résulte un point de décision pour savoir s'il faut développer le modèle ou plutôt identifier un autre aspect du comportement plus pertinent ou prioritaire du point de vue du système. L'analyse peut être menée grâce à la littérature, la connaissance experte du domaine, ou le développement d'outils spécifiques (par exemple

l'étude des phénomènes d'hystérésis dans [24]).

### 3.3 Spécification par les tests

| | | |
|---|---|---|
| Rôle | Expert thématique | L'export exprime, via un jeu de test, les propriétés du modèle qu'il souhaite obtenir. |
| Entrées | $M_{SMA}, f$ | modèle de SMA, facette étudiée |
| Sorties | $\mathcal{T}_f$ | jeu de tests |

Une fois qu'une facette du comportement a été identifiée comme pertinente par l'expert, il convient de s'assurer que le modèle d'agent développé peut la reproduire de manière adéquate. La spécification — non-formelle — du comportement que doivent avoir les agents est également une tâche qui relève de l'expert thématique. Dans l'approche que nous proposons, plutôt que de décrire un modèle de comportement, l'expert définit les propriétés de celui-ci.

La description des propriétés que le comportement des agents doit respecter est assimilée à la description d'un ensemble de $n$ tests $\mathcal{T}_f = \{t_1, t_2, ..., t_n\}$ liés à la facette $f$. Ces tests forment l'interface entre d'une part l'expert thématique qui exprime ce qu'il attend du modèle à produire et d'autre part le modélisateur et l'implémenteur qui doivent fournir une implémentation fonctionnelle répondant au besoin et définie au niveau microscopique. Il est pertinent de fournir plusieurs tests dépendant des différents paramètres à prendre en compte comme par exemple la puissance du véhicule considéré. De manière analogue à des tests logiciels «classiques» qui doivent couvrir un ensemble large de cas d'utilisation d'un programme, les propriétés spécifiées doivent être aussi exhaustives que possible afin de pouvoir être assimilées à une spécification complète.

### 3.4 Modélisation / Implémentation

Une fois les propriétés du modèle décrites par l'expert thématique, la tâche du modélisateur ainsi que de l'implémenteur est de produire une version exécutable d'un modèle ayant les propriétés désirées. De cette façon, la problématique d'ajout d'approximations qui arrive en passant d'une description originale d'un modèle faite par l'expert à son équivalent exécutable par un ordinateur est évitée. Le modèle n'étant jamais décrit initialement, il ne peut être question d'approximation. Si les propriétés sont reproduites, le modèle correspond à l'attente. En revanche, si les choix de modélisation faits par les intervenants mènent à un modèle exécutable ne remplissant pas les propriétés désirées alors l'implémentation est incorrecte [2]. On cherche donc lors de l'implémentation à s'assurer que notre modèle exécutable $m_v$ vérifie $m_v \in \mathcal{M}_{valides} = \{m | \forall t \in \mathcal{T}_f, t(m)\}$.

| | | |
|---|---|---|
| Rôle | Modélisateur + Implémenteur | Le modélisateur propose, avec le concours de l'implémenteur, un modèle opérationnel répondant aux exigences exprimées par le thématicien. |
| Entrées | $M_{SMA}, \mathcal{T}_f$ | modèle SMA, jeu de tests |
| Sorties | $m_v$ | modèle amélioré |

Dans le procédé que nous présentons ici, l'accent est donc mis sur le lien qui existe entre les propriétés du modèle idéal et les propriétés de l'implémentation qui en est faite. Il s'agit d'identifier l'adéquation entre un modèle abstrait et sa mise en œuvre, ce qui correspond à un processus de vérification [22]. Notons que ce processus est à distinguer d'un processus de validation dont l'objet est de s'assurer que le système complet permet de répondre avec suffisamment de précision à une question que l'on se pose sur le système modélisé [18]. Étant donné la nature des modèles à base d'agents, le processus de spécification et de tests décrit ici est appliqué à l'échelle d'un agent, et donc individuel (microscopique). Cependant, et notamment pour l'ajout de facettes interactionnelles, des tests sur des propriétés d'échelles mésoscopiques ou macroscopiques peuvent être intégrés à $\mathcal{T}_f$.

Cette phase n'est pas directement intéressée par la validité macroscopique du système produit, mais bien à la concordance entre les propriétés désirées pour les comportements individuels avec le comportement des entités effectivement mises en œuvre.

Cette approche est particulièrement inspirée du développement dirigé par les tests et cherche à en reproduire les propriétés. Ainsi, elle incite à la concision et à produire une version *a minima* permettant de répondre aux exigences exprimées. Un modèle exhibant plus de propriétés que celles attendues se montrerait être une perte de temps puisque soit les propriétés développées

2. L'utilisation ici du terme «incorrecte» ne correspond pas à sa définition usuelle selon laquelle une implémentation est dite incorrecte si elle ne correspond pas au modèle abstrait dont elle doit être la traduction. Il s'agit ici de décrire une implémentation ne présentant pas les propriétés attendues.

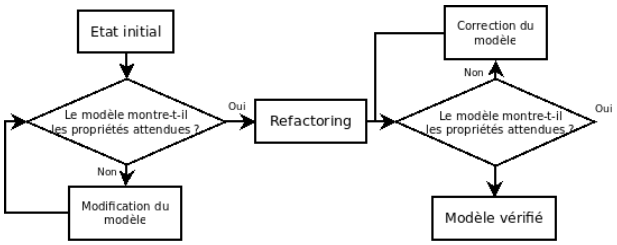


FIGURE 2 – Processus de conception et développement orienté par les tests incluant une phase de *refactoring*.

ne correspondent pas à un besoin exprimé par l'expert thématique, soit ces propriétés sont intéressantes, auquel cas elles devraient faire l'objet d'une itération à part entière. De même, le processus est adapté à l'adoption d'une phase de «*refactoring*» permettant d'accroître la maintenabilité du programme produit comme le présente la figure 2.

### 3.5 Planification

| | | |
|---|---|---|
| Rôle | Expert thématique | L'expert thématique valide (aux vues de ses objectifs de simulation) le système, et initie au besoin une nouvelle itération. |
| Entrées | $M_{SMA}, \mathcal{O}$ | modèle SMA, objectifs de simulation |
| Sorties | $d$ | décision |

De manière analogue au modèle en spirale, notre modèle se termine par une phase de planification. Son objectif est entre autre de déterminer si le système multi-agent $M_{SMA}$ est valide au regard de l'objectif $\mathcal{O}$ ou si des itérations supplémentaires sont nécessaires.

Cette phase est l'occasion de confronter le modèle développé à des données réelles. Ceci est généralement lié au processus de validation, mais la confrontation aux données réelles peut également être le processus par lequel l'expert identifie de nouveaux points sur lesquels le comportement des agents est insuffisant.

Dans le cas où les objectifs macroscopiques n'ont pas été atteints, malgré une cohérence à l'ensemble des tests définis dans la phase de spécification, cela signifie soit que l'amélioration était insuffisante, auquel cas il est nécessaire d'ajouter de nouvelles facettes, soit que l'hypothèse selon laquelle l'introduction de cette facette permet d'atteindre l'objectif de simulation est erronée. Dans ce cas, l'itération suivante sera réalisée sur la base du modèle précédent et non du nouveau modèle.

Ce dernier cadrant sert donc 1) à déterminer s'il est nécessaire ou non de poursuivre le développement du modèle plus avant et 2) à profiter de la confrontation du modèle à des données réelles afin d'identifier des pistes d'évolution. Cette phase relève de la responsabilité de l'expert thématique, dont le rôle peut être rapproché de celui du «*product owner*» des méthodes agiles, dans le sens où il exprime le besoin et valide la solution en fin de développement.

## 4 Illustration d'un cycle de conception

Dans cette partie, nous illustrons la méthode proposée à l'aide de l'application suivante : proposer un modèle permettant de prendre en compte les spécificités des véhicules lourds dans les simulations de trafic routier réalisées à l'aide de systèmes à base d'agents. Afin de servir de base à l'implémentation de nos travaux, nous avons choisi d'utiliser la plate-forme de simulation ARCHISIM. Cette plate-forme dédiée à la simulation microscopique du trafic routier a été le support de nombreux travaux de recherches tels que ceux illustrés dans [10]. ARCHISIM permet de faire interagir différents acteurs indépendants sur une même infrastructure routière. Sur cette base, nous souhaitons identifier et concevoir un modèle de poids lourds améliorant la plateforme de façon incrémentale.

### 4.1 Identification d'une facette du comportement

Afin d'étudier l'apparition des phénomènes émergents que sont les vagues de sur-congestion, l'expert thématique identifie une facette de comportement qu'il juge intéressant. Il peut, par exemple, décider de s'intéresser aux accélérations des véhicules. Il doit alors déterminer s'il est pertinent d'investir dans la reproduction des capacités d'accélération et si cela aura un impact sur les simulations.

### 4.2 Évaluation de l'impact attendu

Si nous considérons l'expérience des modèles macroscopiques du premier ordre supposant des changements de vitesse instantanés, il semblerait que cet aspect ne soit pas important dans la propagation des ondes de sur-congestion. Cependant, les travaux de [9] mettent en avant

que, dans un trafic proche de la saturation, le temps qu'un véhicule met à rejoindre sa vitesse désirée suite à l'apparition d'une perturbation (par exemple suite à l'insertion à vitesse faible d'un véhicule sur l'infrastructure) influence fortement la formation de congestions. Nous pouvons en déduire qu'un modèle de comportement de véhicules lourds doit être en mesure d'exhiber des capacités d'accélération réalistes afin de pouvoir reproduire leur influence sur l'écoulement d'un trafic simulé.

### 4.3 Définition des tests

Nous cherchons alors à proposer un test permettant de vérifier la propriété d'accélération différenciée entre véhicules légers et lourds. Pour ce faire, nous mesurons en situation réelle le profil d'accélération sur un véhicule de type semi-remorque. Le véhicule utilisé pour cette expérience est un ensemble tracté par un IVECO Stralis d'une puissance de 440 chevaux. L'ensemble tracté a une masse de 39T. La donnée de vitesse du véhicule est extraite de son bus «CAN» [3] à l'aide d'outils de diagnostique spécialisés.

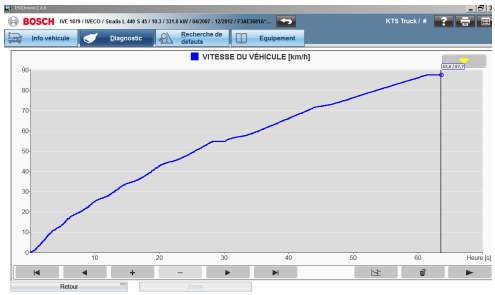


FIGURE 3 – Profil d'accélération d'un véhicule composé d'un semi-remorque / tracteur.

La figure 3 montre que le véhicule parvient à atteindre sa vitesse maximale ($88km.h^{-1}$, ce qui correspond à la vitesse à laquelle le véhicule est bridé) en environ 60 secondes.

### 4.4 Modifications du modèle

Le modèle de simulation ARCHISIM dispose de bases permettant la simulation du comportement de véhicules lourds. Nous évaluons la capacité de ce modèle à reproduire les capacités d'accélération souhaitées. Nous montrons dans cette partie 1) en quoi le modèle original est incapable de répondre aux exigences 2) comment nous proposons d'améliorer le modèle et 3) la vérification microscopique des comportements obtenus. Ces différentes étapes, réalisées par le modélisateur et avec l'aide de l'implémenteur, correspondent à la phase de développement.

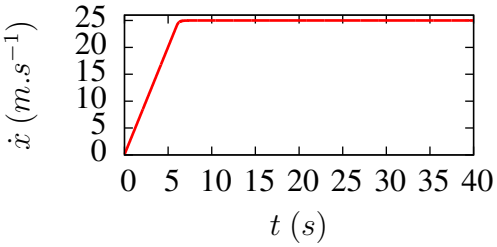


FIGURE 4 – Accélération d'un PL dans ARCHISIM

**Évaluation du modèle existant.** La figure 4 montre le profil en accélération d'un véhicule lourd passant d'une vitesse de $0km$ à $90km$ sur une route plate tel que simulé par ARCHISIM. Cette figure fait apparaître que le modèle ARCHISIM, dans sa version initiale, n'est pas en mesure de reproduire le comportement attendu. Il appartient donc au modélisateur de proposer un amélioration permettant de remédier à cela. Le résultat de cette modélisation sera appelée ARCHIPL.

**Proposition de modélisation.** Afin d'obtenir des profils d'accélération réalistes pour les véhicules ARCHIPL, nous proposons d'intégrer une version adaptée du modèle proposé par [21] au modèle ARCHISIM. Ce modèle dynamique est utilisé afin de déterminer quelle est l'accélération maximale qu'un véhicule peut adopter. Cette accélération maximale est fonction de plusieurs facteurs propres au véhicule (sa vitesse courante, sa puissance…) ainsi que de différentes caractéristiques de son environnement telles que l'inclinaison de la route, la vitesse du vent… Le modèle mis en œuvre simplifie la dynamique des véhicules en différents points de façon à obtenir un compromis entre fidélité du modèle et capacité de paramétrisation.

Afin de modéliser l'effet de la puissance tractrice moteur, un bilan simplifié des forces s'appliquant au véhicule est fait. Comme l'illustre la figure 5, les forces considérées dans ce modèle sont les suivantes : 1) La force motrice, 2) les forces aérodynamiques freinant l'avancement du véhicule, dues au vent relatif, 3) la force de gravité ralentissant le franchissement de pentes, essentiellement du fait de la masse du véhicule, et 4) la force de réaction du support.

3. CAN : Controller Area Network

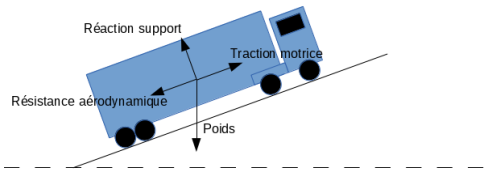


FIGURE 5 – Bilan des forces appliquées à un véhicule sur la route.

Le poids, force verticale, peut se décomposer en deux composantes telles que $\vec{F}_{poids} = \vec{P}_{perp} + \vec{P}_{para}$. $\vec{P}_{perp}$ est perpendiculaire au support et est l'opposée de la force de réaction du support. $\vec{P}_{para}$ est la composante du poids parallèle au support de valeur donnée par $m_a * g * sin(\alpha)$ où $\alpha$ est l'inclinaison du support et $g$ l'accélération de la pesanteur.

La force de traction $\vec{F}_{mot}$ est générée par le moteur du véhicule. Nous supposons que le moteur est exploité au maximum de sa puissance, et négligeons les pertes mécaniques (internes, et liées au contact pneumatique / chaussée). L'intensité de la force motrice délivrée par le véhicule est donnée par

$$F_{mot} = \left\| \vec{F}_{mot} \right\| = P_a / \left( \dot{x}_m \left( t \right) \right) \qquad (1)$$

la force motrice et la vitesse du véhicule étant dirigées selon le même axe. Notons le cas particulier où le véhicule quitte l'arrêt. Dans ce cas, $\dot{x}_a = 0$, et l'équation 1 ne peut être calculée. La force de traction maximale possible est bornée par $F_{mot}^{max} = \mu * F_{essieu\_tracteur}^{z} = g * M_{essieu\_tracteur} * \mu$ [4].

Enfin, la force de résistance aérodynamique $\vec{F}_{aero}$ (qui s'exerce à l'opposé du mouvement du mobile, donc uniquement opposée à la force motrice) a une intensité de $\left\| \vec{F}_{aero} \right\| = q * \mathcal{S} * c_x$. $\mathcal{S}$ est la surface de référence du véhicule (surface exposée au vent), $c_x$ le coefficient de traînée du véhicule, $q = \frac{1}{2} \rho \dot{x}_a^2$. $\rho$ est la masse volumique de l'air et $\dot{x}_a$ la vitesse longitudinale du véhicule.

L'accélération maximale possible pour un véhicule $a$ à un instant donné $t$ est donc donnée, une fois le bilan des forces appliqué, par $\gamma_a^{max}(t) = \frac{1}{m_a} \left[ \frac{P_a}{\dot{x}_a} - k * \dot{x}_a^2 - m * g * \sin\left(\alpha\right) \right]$.

4. cette valeur étant constante pour un véhicule donné, elle est fournie par un paramètre du modèle.

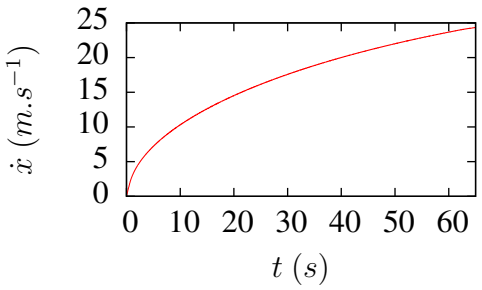


FIGURE 6 – Profil d'accélération d'un véhicule ArchiPL.

**Évaluation des capacités d'accélération du modèle ARCHIPL.** Nous cherchons maintenant à évaluer la pertinence des améliorations que nous avons apportées au modèle ARCHISIM dans le cadre du développement du modèle ARCHIPL. Pour ce faire, nous réutilisons le scénario que nous avons utilisé pour mettre en défaut le modèle ARCHISIM. Nous confrontons le résultat donné par le modèle mis en œuvre avec le profil d'accélération tel que défini dans le test.

Le véhicule simulé, un véhicule de type «PL3» correspondant à un ensemble semi-remorque / tracteur dans la plate-forme ARCHISIM, parvient à atteindre la même vitesse que le véhicule réel en environ 65 secondes. Bien que les résultats expérimentaux et simulés montrent des différences durant les premières phases de l'accélération (les 15 premières secondes), l'amélioration est importante par rapport aux résultats présentés dans la figure 4, dans laquelle le PL atteint cette vitesse en 7 secondes environ.

Ces résultats permettent d'achever le cadrant courant de développement du modèle ARCHIPL.

### 4.5 Planification

Ce cadrant (relevant de l'expert thématique) a pour objet d'une part la validation macroscopique du système de simulation une fois le modèle individuel modifié intégré et d'autre part le choix d'initier ou non une nouvelle itération (et donc de s'intéresser à une nouvelle facette de comportement).

Les résultats de validation associés à cette tâche dépassent le cadre de cet article et ne sont donc pas abordés ici. L'itération concernant la modélisation des accélérations des véhicules lourds dans ARCHISIM est donc considérée achevée.

## 5 Discussion

**Les méthodes de développement de système à base d'agents.** De nombreux travaux se sont intéressés aux méthodes de réalisation de modèles multi-agents. Une revue de différentes méthodes ainsi que de nombreuses références peuvent être trouvées dans [12]. Ainsi, une grande quantité de méthodes est disponible dans la littérature afin de guider le développement de systèmes à base d'agents. Nous pensons notamment aux méthodes Gaia [26], MaSE [7], INGENIAS [20], ou Prometheus [19] pour n'en citer que quelques-unes. Ces méthodes proposent des outils et démarches applicables dans le cadre de la capture des besoins, de l'analyse, de la conception et éventuellement de l'implantation. Gaia propose ainsi un cadre conceptuel permettant l'analyse et la conception d'un système à base d'agents centré sur le concept d'organisation. Cette méthode propose de décrire le système en terme d'agents, environnement, rôles, responsabilités, autorisations, règles et organisations. MaSE propose d'analyser le système par ses buts, et de définir les rôles servant à atteindre ces buts. Ingenias ré-utilise des concepts et outils de RUP [16] en y ajoutant des concepts spécifiques aux systèmes multi-agents tels que l'agent, le rôle, la tâche et l'organisation.

Ces méthodes sont centrées sur le développement initial de modèles à base d'agents, mais non sur leur évolution. Elles fournissent un cadre permettant de comprendre et de spécifier une organisation en décrivant ses membres, leurs rôles, leurs buts, leurs interactions ou leurs moyens. Dans le cas général comprendre ces relations n'est pas une tâche aisée. Cependant, en ce qui concerne la conception d'un système de trafic à base d'agents, cette analyse est presque triviale. Les buts de chacun des agents sont indifférenciés, et ce, qu'ils soient poids lourds ou véhicules légers : atteindre sa destination en minimisant son temps de trajet. Les plans d'actions élaborés par les différents conducteurs sont similaires. Les contraintes propres à chaque agent font cependant que leur exécution peut différer légèrement et entraîner des perturbations au niveau des populations de véhicules. Nos travaux se focalisent surtout sur l'identification et la reproduction de ces variations, dont les conséquences macroscopiques ne sont pas nécessairement bien anticipées. De plus, la littérature existante du trafic nous fournit une base suffisante de modèles théoriques et opérationnels fonctionnels. Une partie des choix de conception sont donc réutilisables. Notre travail cherche à raffiner des modèles de comportements de manière à faire ressortir les variations entre les comportements des individus s'ils impactent l'écoulement du trafic. Les choix de conception déjà effectués ne sont *a priori* pas remis en cause, sauf s'ils montrent leurs limites.

**Une question d'échelles.** Lors de la réalisation d'un système de simulation à base d'agents, l'objet d'étude — c'est-à-dire l'objet dont le modélisateur cherche à comprendre et reproduire les propriétés — est d'ordre macroscopique. Les attentes et exigences en terme de résultats se portent sur les propriétés émergentes. De ce fait, l'activité de validation — activité par laquelle on cherche à s'assurer que le modèle produit permet, par analogie, d'étudier un objet réel avec suffisamment de précision — est intrinsèquement liée à l'ordre macroscopique. L'activité de développement des modèles d'agents utilisés est, elle, réalisée à l'échelle microscopique. La simulation à base d'agents a donc cela de particulier que les objets qu'elle cherche à étudier ne sont jamais modélisés directement, et que les modèles produits ne sont pas étudiés directement. Les modèles d'agents n'étant pas étudiés directement, la notion de validation ne s'applique pas à eux.

Le processus de développement que nous décrivons cherche à maintenir une distinction entre le travail sur les modèles microscopiques et la construction macroscopique. Les phases de travail sur les entités microscopiques ne sont ainsi pas liées à une activité de validation. Cette dernière ne peut avoir lieu que dans la dernière étape du cycle incrémental, et est nécessaire à l'arrêt des itérations.

## 6 Conclusion

L'utilisation de procédures de développements assurant la qualité et l'exactitude des codes produits est particulièrement critique. Comme le souligne [13], la qualité des résultats scientifiques obtenus à l'aide de codes informatiques produits en dehors de cadres de développement rigoureux est fortement affectée.

Dans cet article, nous avons présenté une méthode de développement incrémental de modèles d'agents, TIM4MAS, inspirée de deux approches classiques de la littérature : le modèle en spirale et le développement dirigé par les tests. Dans l'approche proposée, le pilotage du développement est confié à l'expert thématique. Celui-ci définit les propriétés attendues du

modèle d'agent plutôt que son fonctionnement. Tout modèle exécutable remplissant les attentes peut ainsi être considéré comme vérifiant les besoins de l'évolution courante, et la question de la validité du modèle peut être posée. Cette approche permet de limiter les risques d'apparition d'approximations handicapantes dans le processus d'implémentation d'un modèle, et permet de se concentrer sur les aspects ayant la plus forte influence sur l'apparition des phénomènes émergents recherchés.

Nous avons illustré l'utilisation de cette approche pour la conception d'un modèle de poids-lourd sur la base d'un modèle de véhicule classique. Plus généralement, l'aspect incrémental de notre méthode est adapté aux processus de recherche en simulation à base d'agents, en permettant à chaque cycle d'introduire de nouveaux éléments à un modèle existant, puis d'en vérifier la non-régression. La mise à disposition de bibliothèques de spécification permettrait alors de mettre en œuvre des bancs d'essai communs.

## Références


[1] K. Beck. Embracing change with extreme programming. *Computer*, 32(10) :70–77, 1999.

[2] B. W. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5) :61–72, 1988.

[3] M. Brackstone and M. McDonald. Car-following : a historical review. *Transportation Research Part F : Traffic Psychology and Behaviour*, 2(4) :181–196, 1999.

[4] S. Chanut. *Modélisation dynamique macroscopique de l'écoulement d'un trafic routier hétérogène poids lourds et véhicules légers*. PhD thesis, Institut National des Sciences Appliquées de Lyon, 2005.

[5] V. Chevrier and N. Fatès. An example of a multi-agent system described as a discrete dynamical system. In *European Workshop on Multi-agent Systems (EuMAS)*, volume 1, page 17, 2010.

[6] N. David, J. Sichman, and H. Coelho. Towards an emergence-driven software process for agent-based simulation. *Multi-Agent-Based Simulation II*, pages 49–78, 2003.

[7] S. A. DeLoach, M. F. Wood, and C. H. Sparkman. Multiagent systems engineering. *International Journal of Software Engineering and Knowledge Engineering*, 11(03) :231–258, 2001.

[8] A. Drogoul, D. Vanbergue, and T. Meurisse. Multi-agent based simulation : Where are the agents ? In *Multi-agent-based simulation II*, pages 1–15. Springer, 2003.

[9] A. Duret. *Hétérogénéités du trafic autoroutier. Identification, qualification, modélisation et impact sur l'écoulement*. PhD thesis, Ecole nationale des travaux publics de l'état, 2010.

[10] S. Espié. *Simulation comportementale et réalité virtuelle : vers une simulation globale du système de trafic*. PhD thesis, UPMC, December 2004. Habilitation thesis.

[11] J. Galán, L. Izquierdo, S. Izquierdo, J. Santos, R. Del Olmo, A. López-Paredes, and B. Edmonds. Errors and artefacts in agent-based modelling. *Journal of Artificial Societies and Social Simulation*, 12(1) :1, 2009.

[12] J. J. Gómez-Sanz, M.-P. Gervais, and G. Weiss. A survey on agent-oriented oriented software engineering research. In *Methodologies and Software Engineering for Agent Systems*, pages 33–62. Springer, 2004.

[13] E. C. Hayden. Mozilla plan seeks to debug scientific code. *Nature*, 501(7468) :472–472, 2013.

[14] D. Helbing. Traffic and related self-driven many-particle systems. *Reviews of modern physics*, 73 :1067, 2001.

[15] S. Hoogendoorn and P. Bovy. State-of-the-art of vehicular traffic flow modelling. *Proceedings of the Institution of Mechanical Engineers, Part I : Journal of Systems and Control Engineering*, 215(4) :283–303, 2001.

[16] P. Kruchten. *The rational unified process : an introduction*. Addison-Wesley Professional, 2004.

[17] Y. Luo and L. Bölöni. Modeling the conscious behavior of drivers for multi-lane highway driving. *Workshop on Agents in Traffic and Transportation*, pages 93–103, 2012.

[18] M. Minsky. Matter, mind and models. 1965.

[19] L. Padgham and M. Winikoff. Prometheus : A methodology for developing intelligent agents. In *Agent-Oriented Software Engineering III*, pages 174–185. Springer, 2003.

[20] J. Pavón, J. J. Gómez-Sanz, and R. Fuentes. The ingenias methodology and tools. *Agent-oriented methodologies*, 9 :236–276, 2005.

[21] H. Rakha, I. Lucic, S. Demarchi, J. Setti, and M. Aerde. Vehicle dynamics model for predicting maximum truck acceleration levels. *Journal of transportation engineering*, 127(5) :418–425, 2001.

[22] R. Sargent. Verification, validation, and accreditation : verification, validation, and accreditation of simulation models. In *Proceedings of the 32nd conference on Winter simulation*, pages 50–59. Society for Computer Simulation International, 2000.

[23] M. Shannon, G. Miller, and R. Prewitt Jr. *Software Testing Techniques : Finding the Defects that Matter*. Cengage Learning, 2004.

[24] L. Six, S. Ieng, J. Saunier, and Z. Guessoum. Les boucles d'hystérésis comme outil d'analyse des comportements de conducteurs. *Journés Francophones sur les Systèmes Multi-Agents*, 2012.

[25] M. Treiber, A. Hennecke, and D. Helbing. Microscopic simulation of congested traffic. In *Traffic and Granular Flow*, volume 99, pages 365–376, 2000.

[26] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Organisational abstractions for the analysis and design of multi-agent systems. In *Agent-Oriented Software Engineering*, pages 235–251. Springer, 2001.