An Agent Based Framework for Urban Mobility Simulation

Nicolas Marilleau ∗ Christophe Lang ∗ Pascal Chatonnay ∗ Laurent Philippe ∗

∗Laboratoire d'Informatique
de l'Université de Franche-Comté
CNRS FRE 2661
16 route de Gray
25030 Besançon CEDEX
marilleau, lang, chatonnay, philippe@lifc.univ-fcomte.fr

Abstract

Mobility study is composed of many research areas which one interests us: urban mobility. In the literature, urban mobilities are represented by analytical techniques like stochastic laws or they are defined by simulation tools like Multi-Agents Systems (MAS). The goal of our work is to define citizen behaviour in order to observe population dynamics by a simulation. This strategy is facilitated by a meta-model and a toolkit which are used with a particular method. The latter begins by a conceptual representation of each mobile and finishes by a mobility simulator. This paper aims at describing the mobility simulation toolkit. Thanks to this framework, mobility simulator development is simplified. It allows us to create distributed applications which are based on MAS.

1 Introduction

Mobility is more and more studied in many research areas which one interests us : urban mobility. These works are often achieved by a software development. Up today, many simulation softwares of prospective evolution like Mobisim [1] have been created. They help the analysis of particular systems. Due to their mobility representation, they can be applied to describe specific kinds of motion. For example, Vanbergue in [25] focuses on migrations in Bogota.

The goal of our work is to create a generic meta-model[1] and a generic toolkit. They aim at describing spacial mobilities and implementing simulators by following a methodology.

[1]A meta-model is a tool which intends to create another models [21]

Distributed tools, particularly Multi-Agent Systems (MAS) are an interesting answer to our problem. A MAS is composed of autonomous entities, called agents which interact together and move on an environment in order to accomplish tasks.

Agents represent studied mobiles and evolve on a spatial environment. Agents and their environments are built according to real data coming from Geographical Information Systems (GIS). Each agent is, firstly, located at a position which is defined by, for example, geographical laws. Then, it moves according to its behaviour.

We apply the meta-model on an urban environment. We want to exhibit urban dynamics by describing people behaviours and simulate them in a virtual city. This strategy intends to aggregate movements of each person to observe group trips.

In [16, 17], we have presented a meta-model which aims at describing people which move on urban environment. The goal of this paper is to present the associated toolkit we use to develop a simulator of urban mobility.

In a first part, we define the meaning of mobility and how it is implemented in frameworks presented in the literature. Then, we introduce a methodology which use our meta-model and our toolkit we focus on the fourth part. Due to the structure of our meta-model and our toolkit, a generated simulator is based on agent concepts. We describe the general structure of it before presenting each agent with accuracy. Finally, this article is illustrated by a case of study.

2 How mobility concepts are implemented ?

Mobility studies are often achieved by creating models and software in order to design and simulate different kinds of trips. Geographers often focus on human journeys in an urban environment. They make models and implement simulators to describe and simulate different systems like cities or crossroads. These tools help town administration (design of roads, managing of big events, ...).

In the literature, mobility is divided into four main levels: macroscopic [14] mesoscopic [26], microscopic [2, 12, 18] and nanoscopic [7]. According to the level of mobilities we want to describe, different techniques are used. Indeed,

analytical tools are often used to study macroscopic mobilities. Several models of mobility and simulator are based on differential equations [9], statistical queues [6] or Markov chains [11] to study town traffic flows. These techniques can be associated with others which are used to describe microscopic mobilities. cellular automatons [24] or MAS [20] are famous microscopic simulation tools.

Due to their properties, MAS are more and more used in many research area. Interest for this concept increase while computer power grows. Thus many frameworks have been developed. According to [5], there are three kinds of toolkits:

- *general frameworks* (MadKit [10]) are very open. These tools can be used in a wide panel of situations.
- *oriented frameworks* (SWARM [19]) contain functions which help users to create agents with few properties like a representation of a spatial environment.
- *specific frameworks* (Mobidyc [8]) are straightforward. They can be used in a narrow panel of situations.

The meaning of the mobility concept is different according to multi-agent platforms. Some of them, like *GrassHopper* [3], allows *mobile computing* i.e. agents moving from a server to another one. Others contain functions which simulate motions. Swarm [19] allows us to simulate agent journeys in a spatial environment. So, two kinds of motions are proposed in Multi-Agent frameworks: real mobilities and simulated mobilities.

3 From conception to simulation

We want to describe persons who move in a city by defining their behaviours, their goals, their beliefs... The goal is to observe human trips by a simulation. Mobiles are defined by agents which move in a virtual environment. These autonomous entities are cognitive and look like VON-BDI (*Value Obligation Norm-Belief Desire Intention*) agents [4], an extension of BDI agents [13]. VON-BDI agents have been created to describe human. They contains elements which compose BDI agents and new elements like *Value* which represents agent's habits, *Norm* for describing society's habits and *Obligations* which are constraints.

An agent-based meta-model have been created to describe human journeys by representing mobiles' behaviours. It keeps a traditional methodology which is used to develop software (see the figure 1). First of all, a specification document must be written. It describes, textually, the system which we want to study. In fact, it gives a global idea about mobiles that we are going to describe. Then, three steps are needed to obtain a concrete simulator :

1. *The Global description* is the first step. It intends to define how a system is structured. This stage can be achieved thanks to two sub-meta-models. The first one aims at designing mobile structure. It represents mobile beliefs, desires, and so one. The second one is used to describe structure of groups. It shows every element, for example beliefs, desires, that members of a group have in common. Both of these two meta-models use UML diagrams. This language allows us to simply describe mobiles and groups. But, it is not very accurate. It does not describe mobile behaviour rules.
2. *The specification* is the second step. Contrary to the UML description, it represents, with accuracy, the studied system. This stage uses two sub-meta models. One of them, the mobile specification shows how mobiles move, communicate, learn and so on. In fact, a mobile specification allows us to define mobile mental attitudes. The other sub-meta model, the group specification, instantiates mobiles and organises them into groups. A group model locates mobiles in a virtual environment, it defines mobile mental state at the beginning of a simulation. A specification is written with a language called PLOOM-UNITY. This language takes advantage of two formalisms: Mobile-UNITY [22, 23] and PLOOM [15]. It allows us to describe agents behaviour rules by algorithms and predicates. It is not constrained by computer hardware specificities where mobiles are going to be simulated. When this step is achieved, we obtain a mobility model.
3. *The development* aims at implementing mobiles which have been described before. This implementation is facilitated by a toolkit which keeps meta-model structure. Indeed it was created to help users during a simulator creation. This library is composed of classes which are an abstraction of desires, beliefs, and viewed elements. It contains a pattern of mobile which is, in fact, a MadKit agent sub-class. So, when we implement a mobile, we need to create classes defined in UML class diagrams and fill gaps of the mobile pattern by algorithms which have been defined in the mobile specification.

This methodology helps users to design mobiles and to implement them in a simulator. Three steps are necessary to create a simulator. Each of them uses a different tool. Up today, transitions between these different stages are done manually. However, they are simple.

4 Toolkit presentation

The presented toolkit is used at the last step of our methodology. Its main goal is to facilitate development of

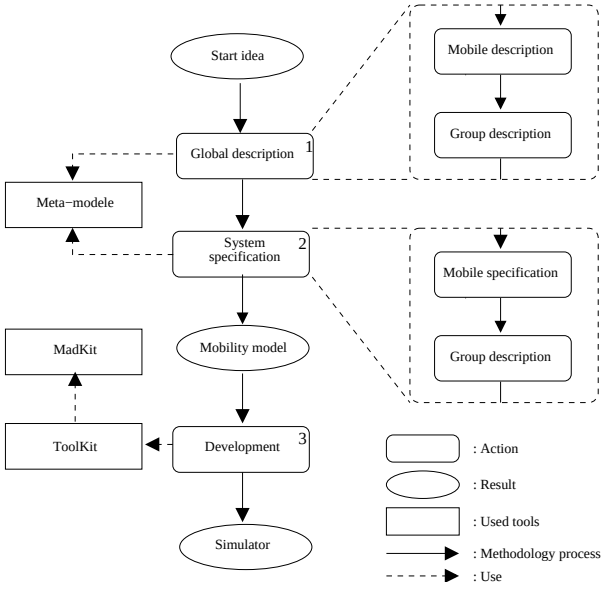

Figure 1. Steps of simulator creation

applications which simulate models of mobility we create according to our meta-model. In fact, it is a pattern we extend in order to create simulators urban mobility simulator.

This toolkit contains specific tools which intend to simulate urban phenomenas. It is an oriented agent-based framework. It contains several abstract classes which allow us to implement a virtual town and mobiles which move on it.

This framework is based on a multi-agent platform called MadKit [10]. The latter solves every problem which are associated with agent life management and communication between agents.

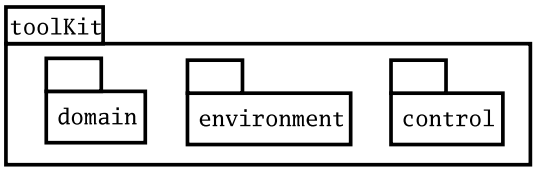

Figure 2. General structure of the toolkit

It is implemented with JAVA language. It is divided into three packages like our meta-model (see figure 2):

- *Domain* contains every class we must extend to implement mobiles and to manage them.
- *Environment* is composed of elements which intend to describe a virtual city by streets and locations (building, house, shop).
- *Control* allows us to implement every simulation access points. It permits to observe and modify a simulation.

A simulator is the result of a development step which aims at extending classes of *Domain*, *Environment* and *Control* packages and a compilation step. Generated simulator can be distributed on grids.

5 Simulator structure

A simulator created with the presented toolkit uses several technologies which come from distributed system area. It is based on a MAS associated with Enterprise Java Beans (EJB) and a database (see figure 3). Agents are organised into four categories:

- *environment agents* aim at representing a virtual city. They manage space elements like buildings or roads, and mobile motions.
- *control agents* allow user to interact with a simulation (seeing motions, modifying simulation parameters).
- *mobile agents* represent mobile we want to study.
- *master agents* manage mobile agent. They can create or destroy them. They are allowed to modify mobile parameters. For example, a master agent can stop an agent evolution when a control agent ask a pause of a simulation.

Several computers can work together in order to execute a simulator. On each site, there is a master agent and an environment agent. It allows us to divide a virtual city into areas. Each area is managed by an environment agent. The associated master agent takes care of mobiles which move on the zone.

Agent cooperation is done by sending messages. These interactions follow FIPA-ACL specifications. For each message, we define, obviously, a sender and a receiver, an ontology name and a content which is written in XML.

Representing time in a distributed mobility simulator is an important problem which must be solved by implementing a distributed clock. We choose to use an EJB in order to create a centralised clock which can be accessed by every agent of a MAS. This bean is managed by an application server called JONAS(Java Open Application Server).

The second goal of the application server is to manage environment data as streets, crossroads or locations. In fact, we will create different kinds of entity beans to save information into an OpenGIS database. Thus, information saved in a database is seen through objects. There is no SQL code in agent source.

An application based on our toolkit contains two kinds of agents: simulation agents to represent mobiles we study and system agents (MasterAgent, Environment and Control) to manage simulation.

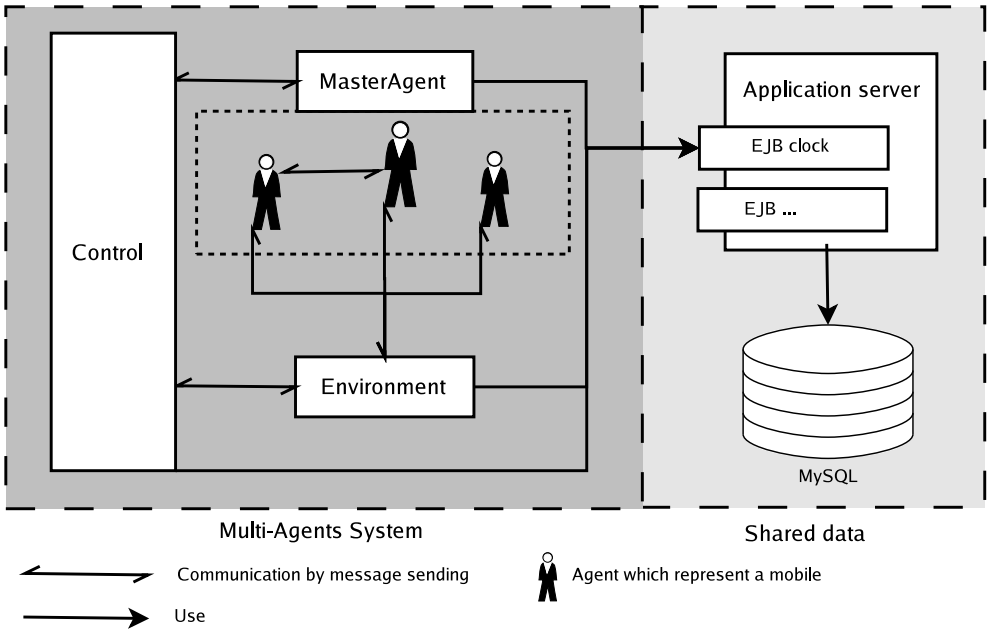

Figure 3. Structure of a simulator

6 Description of agent categories

In previous section, we seen structures of our toolkit and generated simulators. In this section, we pinpoint each part of our framework. We present how mobiles, environment and controllers are structured and how they work.

6.1 Mobiles

Mobiles are represented by autonomous agents. They learn during their life according to interactions they have with the virtual city and other mobiles. They move by following a plan computed thanks to their knowledge, their view, their objectives and their behaviour rules.

Mobile structures we define in conception step thanks to our meta-model [17], is kept in simulator development step (see figure 4). Indeed, we need to extend few JAVA classes called *Belief*, *SeenObject* and *Desire* in order to store knowledge, view and objective information.

A kernel of a mobile is implemented by extending an abstract class (*MobileAgent*). This element contains several predefined functions like *move* or *changeStreet*. These two methods allow agent to move on the virtual city. In addition, *MobileAgent* class owns abstract functions we must extend. For example, we need to implement functions called *computeView* and *actionComputing*. The first one intends to change mobile knowledge according to his view. The second one contains low level behaviour rules.

Behaviour rules are divided into two levels. The high level intends to define cognitive rules which aim at computing agent trip according to is mental state (knowledge, view,...). These rules are implemented by functions written by developers. These methods are called by low level behaviour rules. The goal of them is to move agent according to the path computed before. They are implemented in *actionComputing* function by a succession of conditions.

Each mobile is represented on the evolution world by an instance of *ActiveMobile* sub-class. This object defines how other mobiles can see the current agent. Therefore, it is an agent face. It contains a method (*isViewed*) which aims at computing agent view after each movement.

When a mobile wants to move, it calls *move* operation of *MobileAgent* class. This function sends a message to environment which contains motion vector. When the environment agent receives the message, it changes agent location and computes another mobile view. This perception is sent to mobile by another message. It is a movement acknowledgment.

6.2 Environments

Environment has two goals. The first one intends to define a space where studied mobiles move. The second one aims at creating an ambient traffic and services (shop, restaurant,...) in order to disturb mobile evolutions.

An environment describes an urban space with a set of streets we divide into cells. So, a relative location can be associated for each mobile. This position is defined by the couple of numbers (*street number*, *cell number*). A mobile move cell by cell on a street. When it arrives at the end of

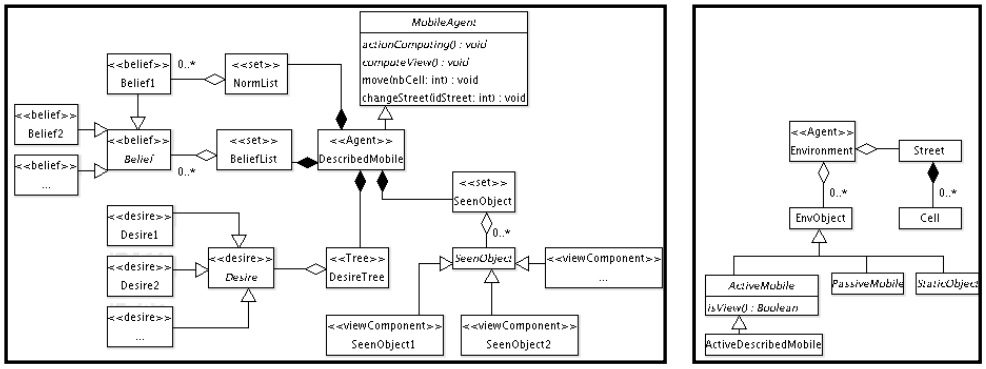

Figure 4. Structure of the toolkit

a street, it goes to a queue before moving to another street. At the end of each street, we associate several queues to represent crossroads. Each of them is associated with few directions (next streets). This technique solves problem of crossroads priorities.

The second role of environment agent is to manage mobile movements and the landscape of the evolution world (building, ambient traffic, and so on). It moves agent on its map, it generate an ambient traffic. These elements are represented by an object. We define three kinds of entities (see figure 4):

- *ActiveMobile* is a representation of mobile agent in an evolution world. In fact, Each class which extends *ActiveMobile*, is associated with a sub-class of *MobileAgent*. It defines data that other agents can read. It is an agent face.

- *PassiveMobile* defines several particles which evolve on streets in order to disturb mobile agent trips. These objects move according to stochastic laws.

- *StaticObject* is an element which does not move on an evolution world. But it evolves according to a life-cycle. In fact we define different states for each *StaticObject*. An environment agent change a state to another one when a particular event arise. These objects are important in our system because they symbolise buildings and services. Mobiles can access and could be attracted by *StaticObject* to achieve desires. For example, a *StaticObject* can describe a shop with two state : open and closed.

So environment agents have to do many tasks like: receiving motion messages, moving mobiles and computing their view, moving particles and changing static elements states by following a life cycle. Environment agents can not achieve these tasks alone. Therefore, they cooperate notably when a mobile wants to go from a street managed by an environment agent A to another one supervised by an agent B.

6.3 Control

The goal of a control agent is to allow user to interact with the simulator. A controller can show mobile journeys with a graphical interface or save motion in a database. It can focus on movements of every mobile or pinpoint knowledge changes of a particular agent. In fact, a controller defines how users see a simulation and which simulation parameter they can modify.

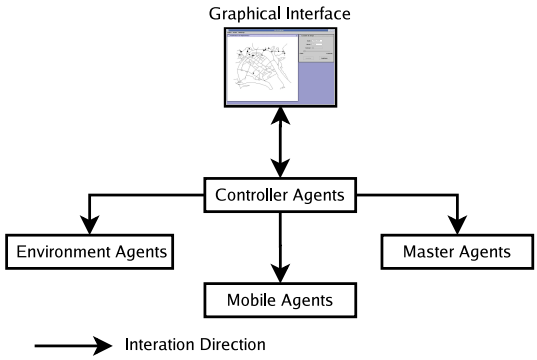

Figure 5. Controller interactions with other

Controllers keep always the same structure (see figure 5). They are composed of interaction systems like a Graphical User Interface and an agent which aims at leading other agent of the simulator. For example, when a street is created by a graphical interface, the associated agent sends a message to an environment agent to insert the new object in the simulation.

Using a controller agent is an interested way to lead a simulation because, it permits us to have a GUI separated

from the simulation. This technique facilitate the creation and the destruction of a controller when a simulation is already started.

7 A case of study

This framework is used in a project called MIRO (Modélisation Intra-urbaine des Rythmes Quotidiens). The goal of this work is to describe human trips which take place in a town by using a bottom-up strategy. At first we want to describe different kinds of citizens by behaviour rules, cognitive maps and few tasks. Then we intend to simulate mobiles in a virtual city.

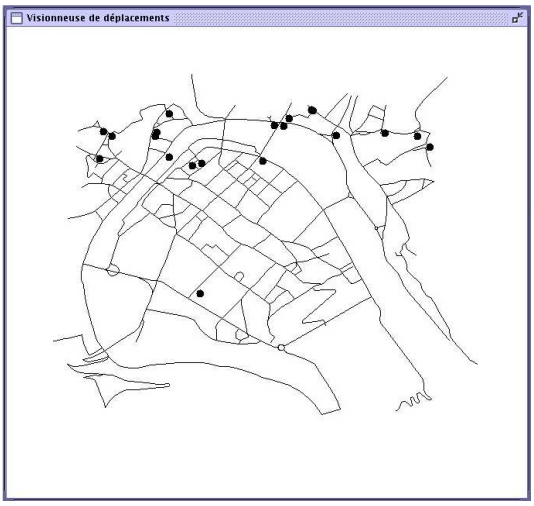

Figure 6. Screen shot of the map controller

Up today, we have implemented an application which simulates pedestrians. These mobiles move with a realistic speed (average 5 km/h). The time is managed by a component we can access through a Graphical Interface (see figure 7). Mobile trips take place in a representation of the center of Besançon (see figure 6). At the beginning of the simulation, they have no knowledge of world and they want to go to a location. As long as their desire are not achieved they move. When a mobile arrives at the end of a street it chooses randomly a next one in his view. Several tests have been done in order to validate toolkit functionalities. A simulator has already been execute in an heterogeneous cluster. The latter is composed of three computers with different hardwares and operating systems. This simulation contains approximately one hundred agents.

8 Conclusion and futher work

We have presented a framework based on multi-agent system. This toolkit facilitates development of urban mobility simulators. It defines citizens by agents which move on an environment (a virtual city) managed by an agent.

Figure 7. Screen shot of the clock controller

This toolkit is used by following a methodology. At first, mobile and group structures are represented by using UML diagrams. Then a formal language called Ploom-Unity is used to define mobile behaviour and instantiate them into groups. After these two steps we obtain a qualitative mobility model. We need to implement it to obtain results. This task is facilitated thanks to the presented toolkit.

The framework is divided into three main parts called: *domain*, *control* and *environment*. The first one contains every class we use to develop mobiles. The second one aims at implementing interfaces we use to interact with a simulator. The last one is constituted by elements which are used to describe an environment.

In a simulator generated by our toolkit, each mobile is represented by an agent which has beliefs, goals and behaviours. It is also managed by agents because environment and control are composed of agents. This structure and MadKit function allow us to create distributed simulator.

Several functions must be added to the toolkit. For example, we need to improve environment in order to add ambient traffic and building management.

A version which include mobile computing concept has been developed. Yet, we have to add load balancing functionality in order to increase velocity.

Up today this framework is applied to a project called MIRO. It will used to simulate a demand transport system.

References

[1] ATN. Simulation de scénario d'évolution, prospective de la mobilité urbaine 20 ans. Rapport de résultats, Ministère de l'équipement et du logement, DRAST, 2002.

[2] J. Barcelò. Microscopic traffic simulation: A tool for the analysis and assessment of its systems. In *Highway Capacity Committee, Half Year Meeting*, Lake Tahoe, 2001.

[3] C. Baumer, M. Breugst, S. Choy, and T. Magedanz. Grasshopper - an universal agent platform based on omg masif and fipa standards. Rapport technique, IKV++, 2000.

[4] G. Beavers and H. Hexmoor. In search of simple and responsible agents. In *The GSFC Workshop On Radical Agents*, 2002.

[5] F. Bousquet, C. Le Page, and J. P. Müller. Modélisation et simulation multi-agents. In *deuxièmes assises du GDRI3*, Nancy, France, 2002.

[6] N. Cetin, A. Burri, and K. Nagel. A large-scale multi-agents traffic microsimulation based on queue model. In *The third swiss transport research conference*, 2003.

[7] N. Daiheng. 2dsim: A prototype of nanoscopic traffic simulation. In *Intelligent Vehicles Symposium*, pages 47–52, Columbus, Ohio,USA, 2003.

[8] V. Ginot and C. Le Page. Mobydic, a generic multi-agents simulator for modeling population dynamics. In P. P. D. Pobil, J. Mira, and A. Moonis, editors, *Tasks and methods in applied artificial intelligence*, volume 1416. LNCS, Springer, 1998.

[9] C. Gloor, D. Cavens, E. Lange, K. Nagel, and W. Schimd. A pedestrian simulation for very large scale applications. *Multi-agenten-systeme in der geographie(Klagenfurter Geographishe Schriften*, 23, 2003.

[10] O. Gutknecht, J. Ferber, and F. Michel. Madkit: une architecture de plate-forme multi-agents générique. Rapport de recherche 00061, Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier, 2000.

[11] M. L. Hazelton. Day to day variation in markovian traffic assignment models. *Transportation research*, 36B:637–648, 2002.

[12] D. Helbing, A. Hennecke, V. Shvetsov, and M. Treiber. Micro- and macro-simulation of freeway traffic. *Mathematical and Computer Modelling*, 35(5/6):517–547, 2002.

[13] D. Kinny. A methodology and modelling technique for systems of BDI agents. In *the seventh European Workshop on Modelling Autonomous Agents in a Multi-agents world*, Eindhoven, The Netherlands, 1996.

[14] A. Kotsialos, M. Papageorgioy, C. Diakaki, Y. Pavlis, and F. Middleham. Traffic flow modeling of large-scale motorway networks using the macroscopic modeling tool metanet. *IEEE Transactions on Intelligent Transportation Systems*, 3(4):282–292, 2002.

[15] C. Lang. Contribution à l'élaboration d'un méta-langage multi-agents pour la description des systèmes complexes. Mémoire de dea, Laboratoire d'Informatique de l'Université de Franche-Comté, 1995.

[16] N. Marilleau. An agent based meta-model for urban mobility modeling. In *The first Internationnal Conference on Distributed Frameworks for Multimedia Applications, DFMA'2005*, pages 168–175, Besançon, France, 2005.

[17] N. Marilleau, C. Lang, P. Chatonnay, and L. Philippe. A meta-model of group for urban mobility modeling. In *procs. of International Conference on Active Media Technology, AMT 2005*, pages 397–400, Takamatsu, Japan, 2005.

[18] E. J. Miller, J. D. Hunt, J. E. Abraham, and S. P. A. Microsimulating urban systems. *Computer, Environment and Urban Systems, Elsevier*, 28:9–44, 2004.

[19] N. Minar, R. Burkhart, and M. Langton. The swarm simulation system, a toolkit for building multi-agents simulations. Technical report, Santa Fe Institute, 1996.

[20] K. Nagel and B. Raney. Large scale multi-agents simulations for transportation applications. In *Behavioral Responses to ITS*, Eindhoven, Pays-Bas, 2003.

[21] OMG. Meta object Facilitors (MOF). Technical report, OMG, 2002.

[22] G.-C. Roman, P. J. McCann, and J. Y. Plun. Mobile UNITY: Reasoning and specification in mobile computing. *ACM Transactions On Software Engineering And Methodology*, 6(3):250–282, 1997.

[23] G.-C. Roman and J. Payton. Agent coordination paradigms in mobile unity. In B. E., G. A., and R. E., editors, *The 10th International Workshop on Abstract State Machines*, 2589, pages 126–150. Lecture Notes in Computer Science, 2003.

[24] P. Simon and H. Gutowitz. A cellular automaton model of bi-directional traffic. *Physical Review E*, 57(2):2441–2444, 1998.

[25] D. Vandbergue. *Conception de simulation multi-agents : Application la simulation des migrations intra-urbaines de la ville de Bogota*. PhD thesis, Laboratoire d'Informatique de Paris 6, 2003.

[26] K. T. Waldeer. Numerical investigation of a mesoscopic vehicular traffic flow model based on a stochastic acceleration process. *Transport Theory and Statistical Physics*, 33(1):31–46, 2004.