RoSMAS[2]: Road Supervision based Multi-Agent System Simulation

Mohamed Habib KAMMOUN[1,2], Ilhem KALLEL[1,3] and Mohamed Adel ALIMI[1,4]
[1] REGIM: REsearch Group on Intelligent Machines, National Engineering School of Sfax, University of Sfax, Tunisia
[2] Department of Computer Science and Communication, Sfax Faculty of Science, University of Sfax, Tunisia
[3] Department Computer Science, ISIGK, University of Kairaouen, Tunisia
[4] Department of Electrical Engineering, National Engineering School of Sfax, University of Sfax, Tunisia
E-mail: habib.kammoun@gmail.com; {ilhem.kallel; adel.alimi}@ieee.org

Abstract **This paper proposes a new approach as well as a simulation based agent technology for road supervision system called RoSMAS[2] (Road Supervision based Multi-Agent System Simulation). This paper presents the architecture, the model of our system with the interactions between the different agents: the cities supervisors, the roads supervisors and the car agents. In particular, we model the interaction to search the best path for cars. In a simulation situation, we use the *TurtleKit* tool on the multi-agent platform *MadKit*. A multi-agent simulator with graphic interface has been achieved to visualize, test, discuss and justify the use of RoSMAS² as road traffic administration.**

Keywords- Road traffic supervision, Multi-agent system, Simulation, RoSMAS².

I. INTRODUCTION

One effect of the technological explosion and the growth of people movements is the increase of the road, naval and aerial traffic. The security, environment and financial constraints accelerate the need of an efficient organization and managing for such complex systems. The result is the birth of intelligent transportation systems (ITS). These systems allow reinforcing the road security thanks to a better cooperation between the vehicle and its infrastructure. In fact, the auto-detection of incidents or clutter situations permits to mobilize intervention means more quickly and better adapted to such situation.

Many works dealing with road traffic have been elaborated, such as the ArchiSim project [1] that introduces new possibilities of traffic regulation in order to ensure as well as possible a comfortable automotive circulation. The Flexiroad project [2] is a solution of road traffic administration, including classifications and speed measures of vehicles.

Following the enormous evolution of number of vehicles, the situations of clutters and traffic jams in all road networks become a plain state. These problems are owed to a bad use of the network mainly, to the absence of real-time information; from where the necessity of an efficient road supervision system supporting complexities, distributions and dynamicity.

The objectives of a road supervision system are to fluidize and to secure the traffic, to anticipate and to react on events, to control the road equipments, to inform the drivers and the partners, from afar, etc.

A new term has been added; it's the one of the *intelligent road*, having as objective to achieve finer and more effective real-time traffic administration, what brings a better request management and a surer displacements. For this, several communication tools and geographical localization systems by satellite (as GPS Global Positioning System and in future GALILEO) are available. Thus, it's possible to position, and therefore to follow vehicles, to communicate with them and with their drivers.

To test and to confirm the road supervision systems, some simulators are used to help looking to the same phenomena with a different level of abstraction and to help understanding, and therefore to take some decisions.

The domain of road administration traffic is well adapted to a based agents approach due to its geographically distributed nature.

Naturally, a user of road network wants to reach its destination with the best compromise time and distance. Thus, we propose a system simulator helping the driver, while distributing intelligence according to a multi-agent approach.

We introduce in the following section some multi-agent approach fundamentals, while putting the accent on the modeling, programming and simulation methods. The third section presents our road supervisor system based agent approach, its organization and interactive architecture. The simulation environment, as well as some results, is presented in the fourth section. The last section gives a conclusion and some perspectives.

II. THE MULTI-AGENT APPROACH

The multi-agent research field, coming from Distributed Artificial Intelligence (DAI), appeared following the evolution of data processing under the flight of objects programming, parallel programming and networks communication. The Multi-Agent approach becomes used in several domains as e-commerce, web information management, remote training and teaching, man-machine interfaces, simulation, etc. It deals with modeling and implementing parallel or distributed systems, considering at the same time the system architecture, agents' architectures and interactions [3].

A. Multi-Agent System

The purpose of multi-agent systems [4] is to study the collective concept putting the accent on organizations and interaction structures such as cooperation, negotiation, actions coordination. They implement several independent entities, having each one precise goals adapted to achieve

some tasks. Nowadays, these systems impose themselves as one of the most adapted paradigms to conceive adaptive, evolving and dynamic systems. They are presented being like a set of agents in interaction in order to achieve their goals or to accomplish their tasks [5] [6].

An agent is a dynamic system; it has an internal state that changes according to its perceptions, and a behavior qualifying the manner of reacting to its environment. Figure 1 schematizes the interaction of an agent with its environment.

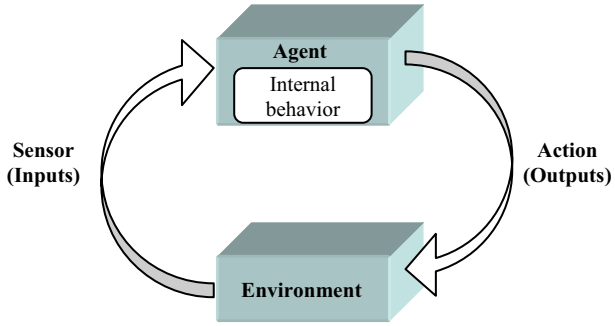

Figure 1. Standard Respresentation of an agent in its environment

As this figure shows, an agent, defined by a set of perceptions (inputs), a set of actions (outputs) and an internal behavior, is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to achieve its design objectives.

B. *Agents Oriented Modelling*

The multi-agent technology is an individual centred paradigm. In fact, it considers that it is possible to model, not only the individuals and their behaviors, but also the interactions between them. The macroscopic level of global system dynamics results directly from the interactions between individuals that compose the microscopic level.

A model is a simplified representation of the reality that helps us understanding the operational system. Classic models use mathematical representations of relationships between different identified entities. However, this modeling doesn't take in account the individual features of the system, whereas the multi-agent approach directly models the interactions generated by individual behaviors [7].

C. *Agents Oriented Programming*

The Agent Oriented Programming (AOP) was proposed by Shoham [8] twelve years ago, which is introduced as a generalization and continuity of Object Oriented Programming (OOP) whitch extensively encouraged the creation of this kind of systems. The following table, due to Van Parunak, sketches this continuity: Paradigm / responsibility.

Shoham assumes that an AOP system consists of three main components: a formal language to describe mental states, a programming language to define agents and a methodology to transform applications not involving actions into agents.

TABLE I. CONTINUITY PARADIGME/RESPONSABILITY

	How	When	Why
Object	X		
Process	X	X	
Agent	X	X	X

To summarize, an object or an agent encapsulates a state and behavior; the difference is that an object has neither goal nor satisfaction research, contrarily an agent tempts to satisfy its objectives and to control its behavior. The agent's concept, and therefore the agent oriented programming [4], provides the required abstraction level. Indeed, multi-agent programming advantages justify the degree of agents' autonomy and flexibility while considering every agent like a source of control within its system.

The agent oriented programming is not a particular implementation technique; it supposes that we can develop programs in which several agents interact, while putting accent on agents social dimension. Thus, most researchers try to describe general principles bound to this approach [5]. Such language must represent the agents' features and interpret messages according to speech act theory.

Nowadays, an increasing number of software tools are available for development of agent oriented systems. Recently, several agent oriented programming languages have emerged [9].

D. *Multi-Agent Simulation*

The simulation is a way of testing models coming from several domains; it is used to confirm the efficiency of this model and to refine it so that it will respects real world constraints. Thus, it facilitates the understanding of the real world and permits to explain some phenomena.

The distributed simulation, by parallel machines, has the advantage to have a classic implementation while avoiding networks problems and remote messaging, but this type of machine is not again too available.

The multi-agent simulation [4] is based on the idea that it is possible to represent entities behaviours in one environment, and agent's interaction phenomenon. The simulation permits to test and to value several use cases without having resort to expensive and difficult tests.

The models of this type of simulation constitute a more and more solicited tool for analysis and complex phenomena understanding. Thereafter, we describe our system as well as the adopted simulation model.

III. A ROAD SUPERVISION MULTI-AGENT SYSTEM

A. *General Idea*

The road supervision system based on multi-agent approach has as objectives, firstly to reach the best road network exploitation and administration, secondly to react to user needs, such as the proposition of a best path attempting his destination. In this context, we propose a short definition of an agent: "*an agent is an entity evolving in an environment, capable to act there and to interact to reach its goal or destination*".

Our approach involves three kinds of agents: City Supervisor Agent (*CSA*), Road Supervisor Agent (*RSA*) and Car Agent (*CA*). By reflexive reasoning, the last one is composed of two agents: Interface Car Agent (*ICA*) and Robot Car Agent (*RCA*). They have, however, the same functioning cycle and exchange messages based on asynchronous point-to-point communication. Each agent lives according to a cycle bound to an iterative process of reception / deliberation / action (figure 2) [10]. The reception represents the identification and the interpretation of all received messages in the mailbox. In addition, it reconstitutes them according to the agent internal believes. The deliberation expresses the whole internal process so that an agent accomplishes its action according to its internal rules while taking into account static and dynamic knowledge. The action describes the operation that an agent executes in order to be able to update its dynamic knowledge, to send a message to another agent, or to act in the environment. It is at this phase that *RCA*s execute their movement commands.

The acquaintances of an agent represent his relations with other agents susceptible to give some advantages. An acquaintance links join a role with the agents that must interact with the one that takes it in charge. This link doesn't permit to know a priori the role of these agents.

To better organize our system, our agents follow the Aalaadin model, more known under the AGR model [11]. As shown in figure 3, the Aalaadin model is based on three core concepts: agent, group and role. An agent is only specified as an active communicating entity which plays roles within groups. The group is considered as atomic sets of agent aggregation. The role is an abstract representation of an agent function, service or identification within a group.

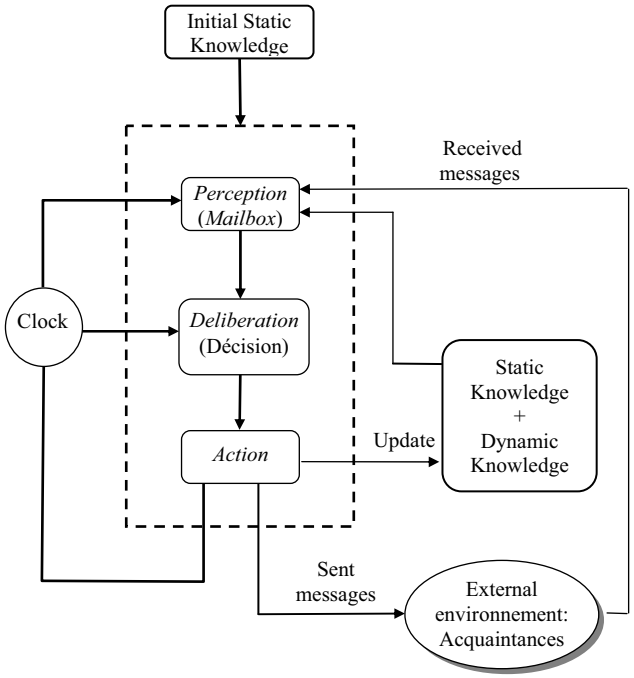

Figure 2. Agent functional structure

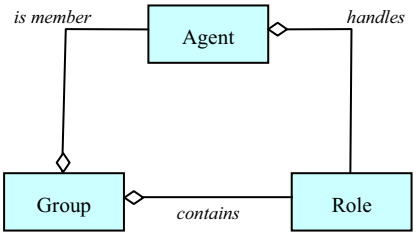

Figure 3. The AGR model (Agent Group Role)

B. *Hierarchical Organization System Architecture*

The problem of the road supervision can be distributed naturally and carry himself well for a hierarchical vertical architecture. Indeed, a vehicle circulates in the roads and a set of road is part of a city. Therefore, to take advantage better of cooperative characters of the agents while minimizing the risk of objective conflict, we choose to represent our system with a hierarchical organizational structure. This type of architecture supports the analysis, the conception and the execution of a SMA composed of heterogeneous agents. The figure 4 present three levels of our system as well as the acquaintance links between the three types of agents: *CSA*, *RSA* and *CA*.

For a better organization, the road supervisors group and the vehicles group are decomposed in subgroups. A change of groups can occur in the level of every *CA* if this moves to another road and eventually to another city.

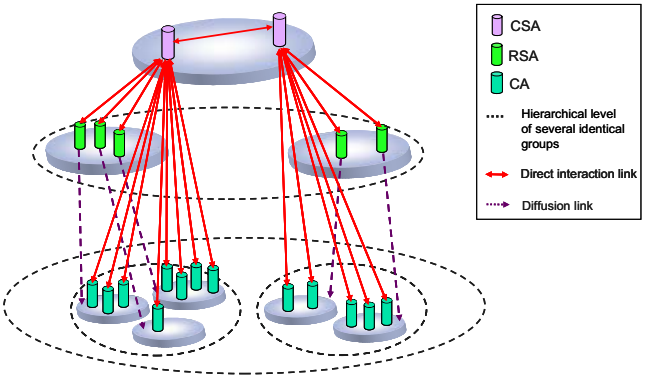

Figure 4. Hierarchical Organization System Architecture

Knowing the environment plan and state, the *CSA* role is to look for the best path helping the *CA* to reach its destination. This research is triggered following the driver's request through the *ICA*, or when the *RCA* comes to a crossroad. The *RSA* computes the traffic road in terms of the flux index expressed by equation (1). The path flux index is the average of flux index in the different roads belonging at this path. This equation is used by the *CSA* in the algorithm of path choice (Figure 5) to follow by the *CA*.

$$\rho = \sigma * l \quad \text{with} \quad \sigma = \frac{nv}{nvMax} * \frac{ts}{tt} \qquad (1)$$

with l is the length of road, nv is the number of *CA* circulating in the road, $nvMax$ is the number maximum of vehicules in the road, ts represents the time put in the situation of clutter by the vehicles circulating in the road in tt period of time.

Algorithm *PathChoice* (agent: *AV*)
Search the possible paths to reach the destination;
Calculate σ for roads belonging possible paths;
For each path do,
Calculate flux index for path;
End For;
Order the paths according to flux index;

Figure 5. Path choice algorithm

The *ICA* is an interface between the car driver and the *CSA*, he communicates with a geographical positioning system to receive the coordinates of the *CA*, which can execute, via its *RCA*, the following commands: *forward ( )*, *left ()* and *right ()*.

Our collaboration diagram illustrated by the figure 6 represents the context of collaboration based on messages. This diagram puts in evidence the agents' organization participating in an interaction. The different elements of this diagram are:

- The apexes represent our system actors or agents
- The arcs represent interaction links
  - ——— : collaboration link
  - - - - - : cooperation link.
- The arrows ⟶ indicate the asynchronous stimulus (message instances) sent or received.
- The sequence number represents the chronological ordering of a message in the stream of control.
  1. ask for best path
  2. ask for flux index
  3. send the flux index
  4. send the possible paths
  5. send the best path

As presented in the interaction diagram (figure 7), our system operates by the following way: the driver sends its request, *getNextRoad (),* through the *ICA*, to its *CSA*; two possible cases: the destination road is situated in the same city or no. If the destination is located in another city, the *CSA* asks for *getTrafficRoad ()* to the concerned *CSA* then updates its orbital dynamic knowledge base. In both cases, the *CSA* consults its knowledge base and executes the order *searchBestRoad ()* to find the best road to follow.

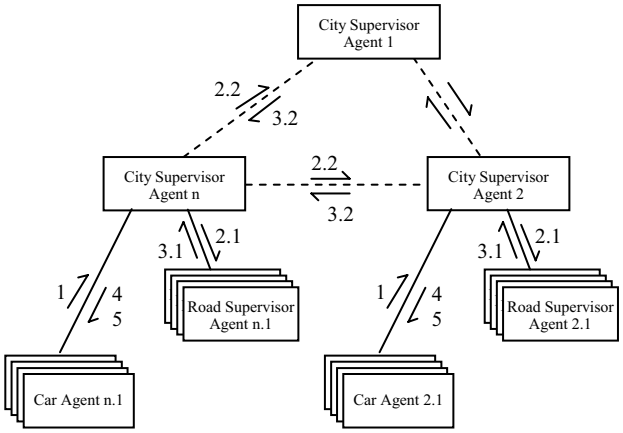

Figure 6. Communication diagram

After receiving, via the *ICA*, the *CSA* proposition, the *CA* can accept or refuse this proposal and send, following its planning, one of the commands *forward ()*, *left ()* or *right ()* to the *RCA* allowing him thus to progress toward its destination.

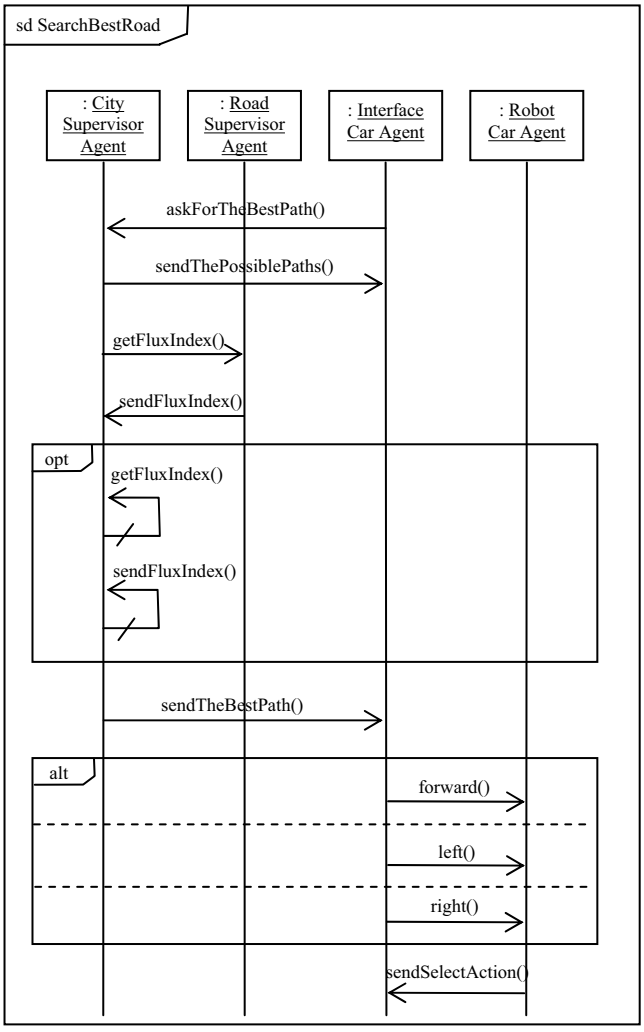

Figure 7. Sequence diagram

To describe the interactions between the different agents, we present in figure 7 the interaction diagram using AUML notations (Agent Unified Modelling Language) [12][13]. This diagram describes the agents' behaviours searching the best path to follow by the *CA*. The vertical lines symbolise the agents' life lines, whereas arrows represent messages transiting from an entity to another. Operator "*opt*" designates an optional combined fragment; it represents a behaviour that can occur.

We notice that our system architecture looks like Internet network, where the *CSA*s are assimilated to routers since they manage routing tables of accessible roads. The execution of *searchBestRoad ()* command uses the same techniques as Internet network. Therefore, the research of the best path to reach destination is similar to the research of the best path to send a message from a starting IP address to a destination IP address.

IV. System Simulation: RoSMAS²

The road supervision multi-agent system described in the precedent section represents the kernel of the simulation. We nominate the whole system RoSMAS², acronym for "*Road Supervision Multi-Agent System Simulation*".

A. Simulation Environment

Nowadays, we note the birth of some multi-agent simulation platforms. Following a survey on multi-agent platforms, we adopt the MadKit platform [14] (Multi-Agent Development Kit) since it is built around the adopted Aalaadin agent/group/role model. In addition to these concepts, the platform adds three design principles: micro-kernel architecture, agentification of services and component model for graphical interface. This toolkit is a generic platform based on the general conceptual model which underlies it. MadKit itself is a set of Java classes' packages that implements the agent kernel, the various libraries of messages, the protocol of communication and the specific concepts of agents (task, goals, etc.). It also includes a graphical development environment and many system and demonstration agents. The MadKit kernel handles only the following tasks: Control of local groups and roles, Agent life-cycle management and Local message passing. MadKit provides several kinds of predefined messages such as StringMessage, XMLMesssage and ActMessage. The latter is the starting point for defining ACLMessages and KQMLMessages (message in conformity with the standard of FIPA: Foundation for Intelligent Physical Agents) [15]. The MadKit manages the *mailBox* with the FIFO principle (First In First Out).

When building a multi-agent system, it might be necessary to launch hundred of agents. A standard MadKit agent encapsulates a small simulation engine. The synchronous engine provides five essential building blocks: the Referenceable Agent, the Scheduler Agent, the Activator class, the Watcher agent and the Probe class, things that justify the use of this platform comparing to other existing platforms [16].

To take advantageous from the TurtleKit tool [17], reactive agents are considered in RoSMAS². In fact, a TurtleKit tool is a reactive agent execution tool that runs on the "synchronous engine" of MadKit platform. To execute the simulation, TurtleKit offers the launcher agent having the role to set up, launch and manage turtles (figure 8).

Created via the TurtleKit simulation viewer, our simulation environment contains several roads in only one city for raison of simplicity; a variable number of cars circulate with random and autonomous way; one *CSA* for city and one *RSA* for each road (figure 9-11). The environment handles three kinds of cars: one intelligent car representing the *CA* behaviour, many classic cars and some bad cars to simulate clutter events.

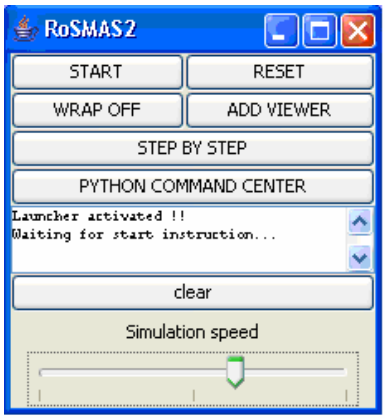

Figure 8. The simulation Launcher's default GUI

B. Simulation results

During the simulation, we compare the behaviours of intelligent and classic cars coming firstly by the road 1 and reaching their destination: road 5. Figures 9 and 10 present the choice of the best road while avoiding a clutter existing in road 4. The intelligent car receives the best road to follow, road 3, according to the *CSA* advice based on the roads flux index.

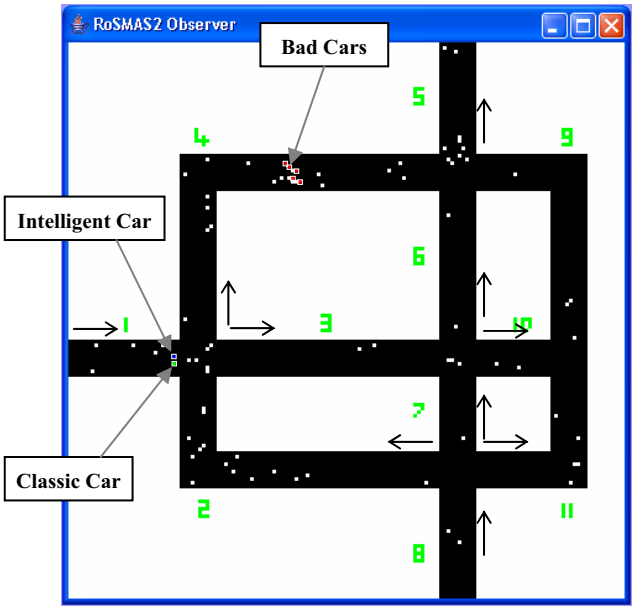

Figure 9. Simulation environment for RoSMAS²

Figures 10 and 11 reveal the efficiency of our algorithm presented in the figure 5. In fact, it is also adapted for a choice between two paths having the same fluidity, since the path length is a parameter of the function (1).

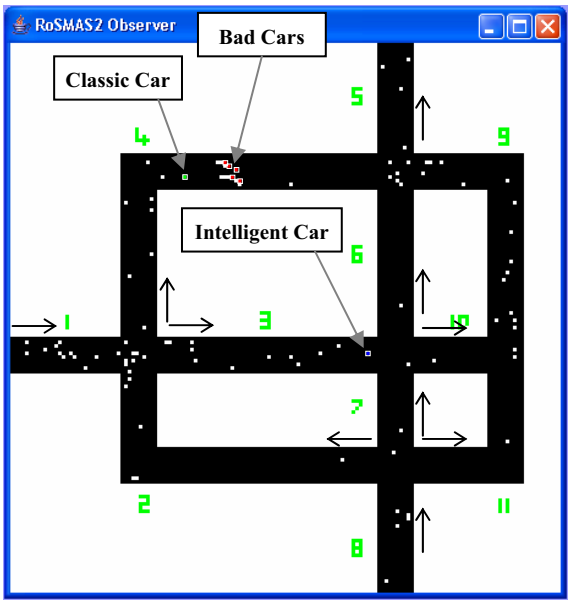

Figure 10. Best path choice for the second crossroads

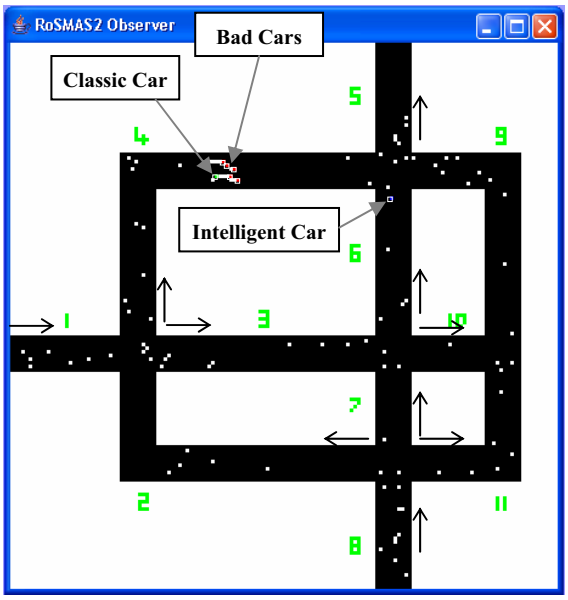

Figure 11. The *CA* achieve their destination

Figures 12 and 13 detail the messages sent between *CA* and *CSA* for the choice of the best road in the first and second step. They also present the values of the flux index for the possible path computed by the *CSA*. These messages are considered as a cooperation tool between the *CSA* and the *CA*.

```
<Kernel> : [turtle208] Car Agent --> City Supervisor Agent : Road1 to Road5 ?
<Kernel> : [City Supervisor 1] flux index in road 4 : 82
<Kernel> : [City Supervisor 1] flux index in road 3 and 6 : 29
<Kernel> : [City Supervisor 1] flux index in road 3, 9 and 10 : 47
<Kernel> : [City Supervisor 1] follow road 3
<Kernel> : [turtle208] forward
```

Figure 12. Communication messages between *CSA* and *CA* for the first crossroads

```
<Kernel> : [turtle208] Car Agent --> City Supervisor Agent : Road3 to Road5 ?
<Kernel> : [City Supervisor 1] flux index in road 6 : 22
<Kernel> : [City Supervisor 1] flux index in road 9 and 10 : 38
<Kernel> : [City Supervisor 1] follow road 6
<Kernel> : [turtle208] turn left
```

Figure 13. Communication messages between *CSA* and *CA* for the second crossroads

The user of the simulator can visualize the flux index of every road. Figure 14-16 plot the flux index of the possible paths to reach road 5. Precisely, figure 14 shows, that following the increase of flux index, a clutter took place at the road 4. Consequently, the *CSA* detect this clutter and inform a *CA* of the situation in road 4. The difference in the flux index plotted by figure 15 and 16 is due to the difference of the distance between the 1st and 2nd path.

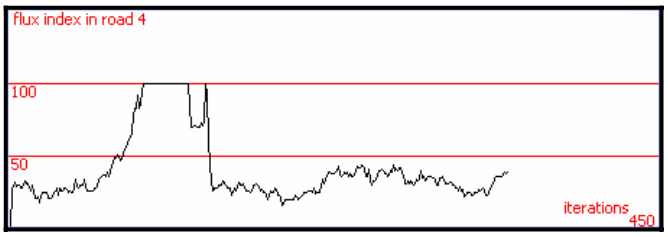

Figure 14. Flux index observer in Road 4

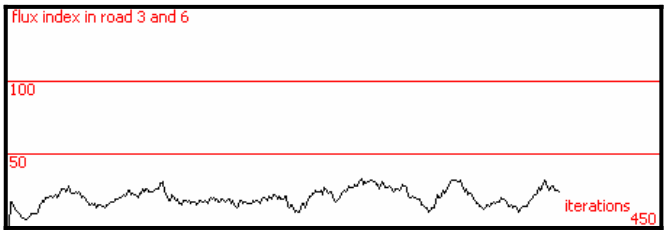

Figure 15. Flux index observer in Road 3 and 6

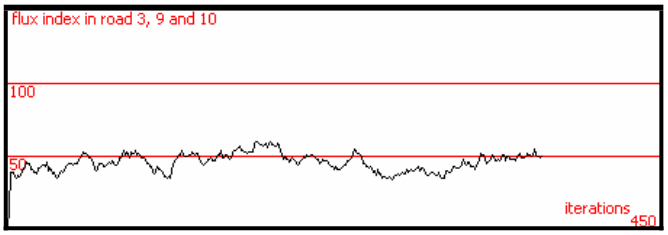

Figure 16. Flux index observer in Road 3, 10 and 9

Figure 17 plots the flux index in the city representing the average of flux index of different roads in city 1. We can see the negative influence of the clutter in the road 4.

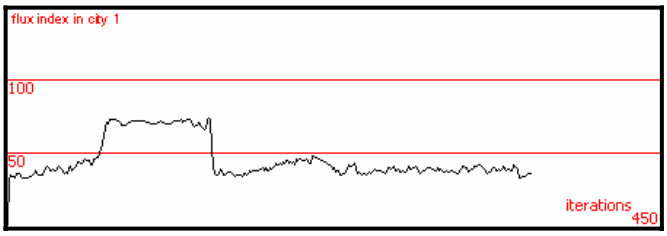

Figure 17. Flux index observer in city 1

The results illustrated by the figures 9-17 confirm the objective of our approach: a multi-agent approach for a road supervision system simulation, allowing improving the exploitation and administration for the road network traffic, thanks to the distribution and cooperation between the different agents composing our system.

Other simulations are done with other positions of clutter. Thus, the *CA* always reaches its destination with the best path. Figure 18 present other simulation of RoSMAS² with a clutter situated in road 6. We notice that road 4 is the best road to follow in relation to the two other paths: the first (roads 3 and 6) presents a clutter in road 6; the second (roads 3, 9 and 10) is longer than the road 4.

V. Conclusion

In this paper, we presented a hierarchical architecture as well as the model of our road supervision system based on a multi-agent approach. We presented the collaboration and interaction diagrams for the research of the best path. A road network simulator, RoSMAS², has been achieved while using TurtleKit tool of MadKit multi-agent platform. Some simulation results are also presented.

In relation to others systems of road supervision, our system present a better organization to the road network with a hierarchical architecture based on a model organizational multi-agent, use a universal modeling language, offer a global vision to the road network and adapt well at urban environment that to interurban environment.

As perspectives we intend in the near future to add other cities in our simulation system, and then to take into account other road infrastructures as well as the presence of crossroads and why not, validate our system in the real world. It seems quite applicable for our case to adopt the Internet network routing techniques. We think that it will permit to give better results for routing traffic.

Also, a problem of resource conflict can be present when a car takes in the same time, the same trajectory. Consequently, coordination and planning strategies are necessary.

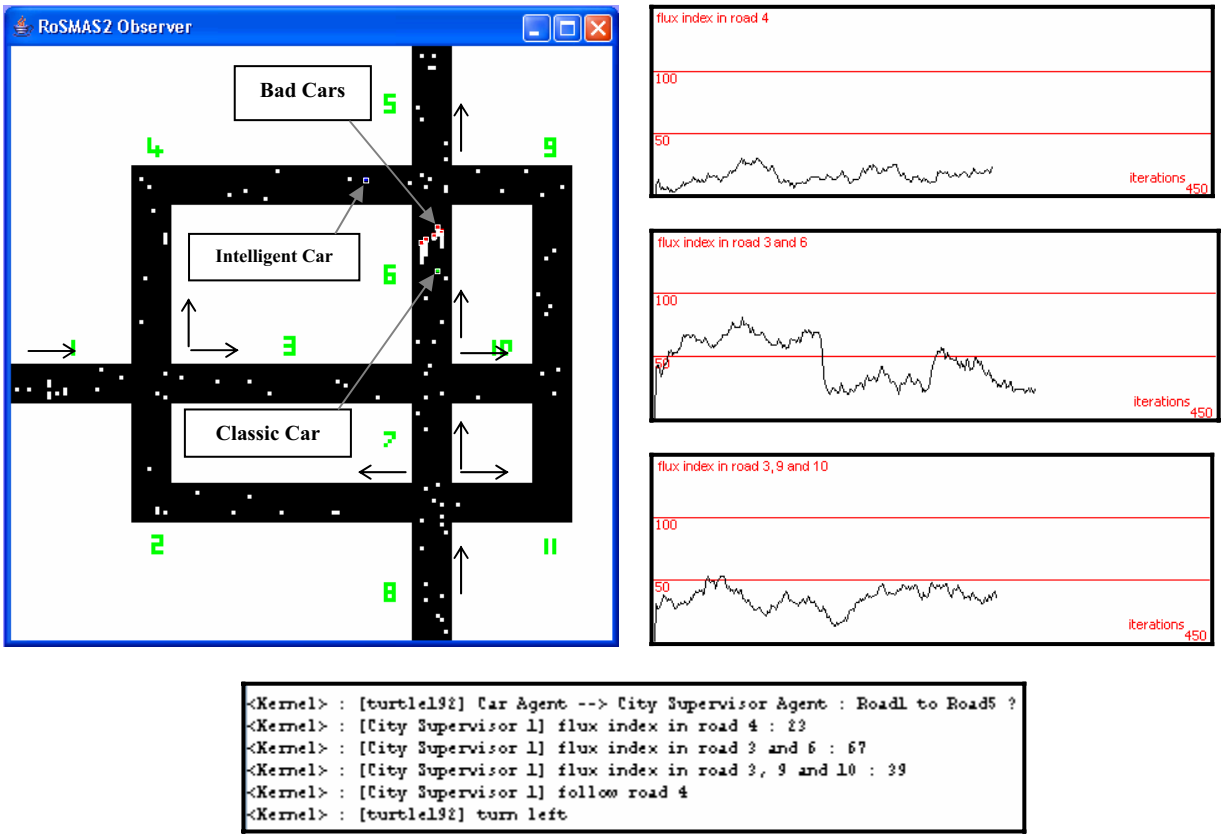

Figure 18. Other simulation of RoSMAS²

ACKNOWLEDGMENT

The authors would like to acknowledge the financial support of this work by grants from the General Direction of Scientific Research and Technological Renovation (DGRSRT), Tunisia, under the ARUB program 01/UR/11-02.

REFERENCES

[1] ArchiSim, http://www.inrets.fr/ur/simus/archisim.htm

[2] Flexiroad, Road Traffic Analysis, http://www.capflow.com/solutions-flexiroad-en.php

[3] S.J. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd Edition, Prentice-Hall, 2003.

[4] J. Ferber, *Multi-agent system: an introduction to distributed artificial intelligence*, Addison-Wesley, 1999.

[5] N.R. Jennings, "On agent-based software engineering", *Artificial Intelligence*, 117(2) 277-296, 2000.

[6] M. Wooldridge, *An introduction to MultiAgent System*s, John Wiley and Sons Ltd, February 2002.

[7] F. Klügl, C. Oechslein, F. Puppe and A. Dornhaus, "Multi-agent modelling in comparison to standard modelling", *In AIS'2002*, F.J. Barros and N. Giambiasi (eds.), pp. 105-110, 2002.

[8] Y. Shoham, "Agent-oriented programming", *Artificial Intelligence*, 60 (1), pp. 51-92, 1993.

[9] A.F. Seghrouchni and A. Suna, "CLAIM: A Computational Language for Autonomous, Intelligent and Mobile Agents", *In ProMAS (AAMAS 2003)*, Melbourne-Australia, 14-18 July 2003.

[10] I. Kallel, M. Jmaiel and M.A. Alimi, "A Multi-Agent Approach for Genetic Algorithm Implementation", *In IEEE SMC'02*, Hammamet-Tunisia, October 2002.

[11] J. Ferber and O. Gutknecht, "A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems", *In Proceedings of the 3rd ICMAS*, pp.125-135, 1998.

[12] J. Odell, H.V.D. Parunak and B. Bauer, "Extending UML for Agents", *In Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence*, Gerd Wagner, Yves Lesperance, and Eric Yu (eds.), Austin, TX, pp. 3-17, 2000.

[13] B. Bauer and J. Odell, "UML 2.0 and agents: how to build agent-based systems with the new UML standard", *Journal of Engineering Applications of Artificial Intelligence*, Volume 18, Issue 2 , pp. 141-157, March 2005.

[14] J. Ferber, O. Gutknecht and F. Michel, "MadKit Development Kit", version 4.0, 2004. [Online] Available: http://www.madkit.org

[15] B. Chaib-draa and F. Dignum, "Trends in Agent Communication Language", *Computational Intelligence*, 18(2), pp. 89-101, 2002.

[16] M.D. Graças, B. Marietto, N. David, J.S. Sichman, H. Coelho, "Requirements Analysis of Multi-Agent-Based Simulation Platforms: State of the Art and New Prospects", *In Proceedings Third International Workshop on Multi-Agent Systems and Agent-Based Simulation*, In Jaime S. Sichman, François Bousquet and Paul Davidsson (eds.), Lecture Notes in Artificial Intelligence, Springer-Verlag, 2002.

[17] F. Michel, "Introduction to TurtleKit: A Platform for Building Logo Based Multi-Agent Simulations with MadKit", RR LIRMM 002215, June 2002.