ICS: An Interactive Control System for Simulating the Probability of Car Accidents with Object Oriented Paradigm and Cellular Automaton

A. Madani and N. Moussa
Département de Mathématiques & Informatique
Faculté des Sciences, El Jadida, Morocco
madaniabdellah@gmail.com, najemmoussa@yahoo.fr

***Abstract*— As the number of vehicles continues to grow, more accidents occur most likely due to careless driving of the drivers who do not maintain the safety distance. To assist managers in their tasks of traffic planning, a system for decision support becomes essential. This article aims to describe our interactive control system (ICS), which falls into this category. Such a system should allow, through graphical tools, to view the current state of the traffic to make decisions to avoid congestion problems that are essentially the cause of car accidents.**

Keywords—Car traffic, car accidents, cellular automaton models, Object Oriented Simulation, UML

I. INTRODUCTION

In recent years, most industrialized societies have started to notice the limits of the growth of urban traffic. The traffic demand in cities has largely exceeded the roads capacity and we face the problem of increasing pollution, consumption, waiting time and growing frequency of car accidents. To circumvent the problem, constructing additional roadways could be seen as an intuitive solution, but this is undesirable due to political objections or impractical because of the high costs of construction. Faced with this complex problem, many researchers are beginning to think seriously to find solutions. They are motivated by two main elements:

On the one hand, a number of traffic models have been proposed to investigate the dynamical behavior of the traffic flow, including fluid dynamical models [1], gas-kinetic models [2,3], car following models [4] and cellular automata models [5,6]. Compared with others traffic models, cellular automata models are conceptually simpler and can be easily implemented on computers for numerical investigations. The related research has been developed very quickly in the last decade after the first basic cellular automata model was proposed by Nagel and Schreckenberg [5], describing a single-lane traffic flow.

On the other hand, when a traffic management system involves many thousands of vehicles using hundreds of streets and highways, it can be difficult or impossible to tell whether the network is flowing smoothly and to predict how modifications to dynamic control parameters will affect the system. For such systems a management tool has become more and more important for evaluating, planning, controlling and predicting the effects of changes in parameters of the control system.

The aim of this paper is to describe an Interactive Control System (ICS) whose purpose is to help human controllers (with ergonomic and easy to use Human Machine Interface) to avoid having congestion problems (deadlocks, bottlenecks, jams, …) in traffic if they have not yet materialized, and disabling them quickly if they have. These problems in traffic can be caused by car incidents like collision, accidents or car failures. They can also be caused by deterioration in traffic network flow like jams. Congestion problems can be used as basis for the ICS to visualize the current state of the network allowing then the planners to explore through different graphical tools (curves, histograms …) how avoiding such problems.

Our work is divided into two distinct parts. The first one concerns the simulation of car accidents. This section begins by recalling some models used to investigate the dynamical behavior of the traffic flow (section II.1) and those used for the simulations of car accidents (section II.2). The second part meanwhile, deals with the architecture of our system. We begin in Section III.1 by recalling the benefits (advantages) of a modular architecture. Section III.2 presents the UML modeling of the ICS. Results of micro-simulation, considering the probability of the occurrence of car accidents, are shown in Section III.3 and finally, in Section IV are presented the conclusions and possible future work.

II. MODELS AND THEORETICAL TECHNIQUES

In the cellular automata models a lane is represented by one dimensional lattice of L sites. Each of the lattice sites can be either empty or occupied by at most one vehicle. If periodic boundary conditions are imposed, the density ρ of the vehicles is N/L where N ($\leq$L) is the total number of vehicles. Throughout this paper we shall follow the convention that the vehicles move from left to right.

A. Car traffic models

The evolution of complex urban traffic has posed significant challenges to city planners in terms of optimizing traffic flows in a normally congested traffic network. When designing traffic control systems of a big city, the software engineer must model both mobile entities (cars, pedestrians, vehicular flow, and so on) and fixed entities (roads, traffic lights and so on). The task is not obvious, so the need of models. A good model should avoid incidents that may arise such as accidents, collisions and car failures, in order to solve congestion problems in traffic like deadlocks, bottlenecks and jams.

1. The NaSch Model

In recent years it has turned out that cellular automata are excellent tools for the simulation of large scale traffic networks. The most prominent example for this kind of models, the NaSch model, has been introduced by Nagel and Schreckenberg [5].

For sake of completeness we briefly recall the definition of the NaSch model [5]. The NaSch model is a discrete model for traffic flow. The road is divided into L cells of equal size numbered by i= 1, 2, … L. Each cell can be either empty or occupied by exactly one car. The cars have a velocity v = 0, 1, …, v_{max}, where v_{max} represents the speed limit. The cars move from the left to the right on a lane with periodic boundary conditions (the number of cars remains constant). Let x(i,t) and v(i,t) denote the position and the velocity of the ith car at time t, respectively. The number of empty cells in front of the ith vehicle is denoted by g(i,t) = x(i+1,t) – x(i,t) - 1. g(i,t) is the gap between the ith car and the preceding one. The system update is performed in parallel for all cars according to the following rules:

1. Acceleration : $v(i,t+1) \leftarrow \min(v(i,t) + 1, v_{max})$
2. Deceleration : $v(i,t+1) \leftarrow \min(v(i,t+1), gap(i,t))$
3. Noise : $v(i,t+1) \leftarrow \max(v(i,t+1) - 1, 0)$, with probability p
4. Motion: $x(i,t+1) \leftarrow x(i,t) + v(i,t+1)$

These four rules can be interpreted as follows: the driver tends to drive as fast as possible without exceeding the maximum velocity (rule 1), but he must decelerate to avoid collision (rule 2). The rule 3 called also the randomization rule, takes into account the different behavioral patterns of the individual drivers, especially nondeterministic acceleration and overreaction while slowing down. The rule 4 moves the vehicle according to its current velocity.

2. The NaSch Model with the «Slow-To-Stop » rule

Clarridge et al. [7] has proposed a variant of the basic NaSch model. This amelioration concerns the deceleration of vehicles by adding a "slow-to-stop" rule. This rule allows drivers to look farther ahead and slow down gradually earlier when their cars come closer to the jam. In this new version of the NaSch model, the car's velocities are adjusted at each time step according to the following rules, where v_{next} represents the velocity of the car ahead (predecessor car):

1. Acceleration: v(i, t+1) ← min(v(i, t) + 1, v_{max})
2. Deceleration (when the predecessor car is nearer): if gap(i, t) ≤ v(i, t+1) and either v(i, t+1) < v_{next} or v(i, t+1) ≤ 2 then v(i, t+1) ← gap(i, t) – 1. Otherwise, if gap(i, t) ≤ v(i, t+1), v(i, t+1) ≥ v_{next} and v(i, t+1) > 2 then v(i, t+1) ← min(gap(i, t) – 1, v(i, t+1) – 2)
3. Deceleration (when the next car is far): if v(i, t+1) < gap(i, t) ≤ 2v(i, t+1), then if v(i, t+1) ≥ v_{next} + 4 then v(i, t+1) ← v(i, t+1) -2. Otherwise, if v_{next} + 2 ≤ v(i, t+1) ≤ v_{next} + 3 then v(i, t+1) ← v(i, t+1) – 1
4. Randomization: v(i, t+1) ← max(v(i, t+1) – 1, 0), with probability p.
5. Motion: x(i, t+1) ← x(i, t) + v(i, t+1)

B. Car accidents Models

Most recently, the occurrence of car accidents has been studied within the framework of cellular automata models with both periodic boundary conditions [8-11], as well as with open boundary conditions [12].

In the basic NaSch model, car accidents will not occur, because the second rule is designed to avoid accidents. Thus the safety distance between drivers is respected. However, in real car traffic, accidents occur most likely due to careless driving of the drivers who do not maintain the safety distance. More precisely, if the car ahead is moving, expecting it to be moving at the next time step, a careless driver has a tendency to drive as fast as possible and increases the safety velocity by one unit. At the next time step, it will arrive at the position of the moving car ahead. If the moving car ahead suddenly stops, or make a great deceleration a collision between the two cars takes place. These situations are called Dangerous Situations, DS [8, 9].

Moussa [10] has introduced conditions for the occurrence of car accidents in the NaSch model [5]. These conditions are based on the delayed reaction time of drivers. He distinguished two types of conditions according to the behavior of the car ahead: sudden stop or abrupt deceleration.

1. Car accidents caused by stopped cars

We assume here that a careless driver (whose driving is not careful enough) who see the car ahead moving, he excepts it to move again at the next time step, and tends to drive as fast possible. If the leading car stop suddenly, the successor driver decelerate to avoid collision, but his reaction is carried out after a reaction time τ. This reaction time is defined as the time passed between the instant where the brake light of the predecessor car is switched on and the one where the successor car begins his braking maneuver.

The dangerous situation (DS) between two neighborhood cars k and k+1 will exists at time t+1 if the following equations are accomplished [10]:

(i) τ v(k, t) > g(k, t)

(ii) v(k+1, t)> 0

(iii) v(k+1, t+1) = 0

These equations can be interpreted as follows: The distance required to cover by the kth car during the time τ is superior to its gap (i), the (k+1)th car is moving at time t (ii) and the (k+1)th car will stop at the next time step t+1 (iii).

2. Car accidents caused by great deceleration

The second conditions for the occurrence of car accidents concern vehicles moving at high speed. It is obvious in this case that a great deceleration of a car can cause an accident with its successor. Based on the delayed reaction time τ of the successor vehicle and the unexpected abrupt deceleration of the predecessor vehicle, the conditions for the occurrence of a DS are expressed [10]. When the vehicle ahead with velocity v(k+1, t) at time t do an abrupt deceleration at time t+1, its velocity will be reduced to v(k+1, t+1). Thus, if the covered distance during the delayed reaction time τ of the successor vehicle is enough to reach the next time position of the vehicle ahead, DS happens most likely on the road. Hence, the conditions for the occurrence of car accidents with respect to abrupt deceleration of the vehicle ahead are as follows:

(i) τ v(k, t) > g(k, t) + v(k + 1, t + 1)

(ii) v(k + 1, t) – v(k + 1, t + 1) ≥ v_d

Where g(k, t) is the number of empty cells in front of the vehicle k at time t. The parameter v_d is the deceleration limit beyond which a risk of the occurrence of DS exists.

III. ICS: System Architecture

The programming of the simulation application itself is more complex. On the one hand, there are several different objects that both operate independently and interact with each other directly: there is road, vehicles…, all changing their state as the simulation proceeds. Each of these objects has its own characteristics and behaves differently. On the other hand, this simulation must allow the planners to see all facets of the traffic under different constraints. To allow such experiments, the user interface design needs to be considered.

A. *Object Oriented Simulation*

Object Oriented Simulation (OOS) is a new paradigm that has been applied to simulation problems in many fields: biology, engineering, economics and sociology. An object-oriented simulation consists of a set of objects that interact with each other over time. An object can be described by an entity that holds both the descriptive attributes of the object as well as defines its behavior. For example, suppose you are modeling a car traffic application, objects that should immediately come to mind include the vehicles, the pedestrians, the traffic lights, etc. For a vehicle, the position and the velocity are examples of attributes, while the acceleration, the deceleration and the motion describe the behavior.

The major difference between classical simulation and OOS is in the viewpoint of simulation system. A classical simulation takes system as complex with the global rules and static description. In this case, model of system is characterized by global mathematic formulas as statistical function. In contrast, OOS is focusing on system entities that form the system. The entities are described as autonomous units with its states and behaviors. Thus simulation is based on interaction among entities (Objects) and its environment.

The paradigm of Object Oriented (and hence the Object Oriented Simulation) presents many advantages like easier mapping of application concepts into language constructs, more readable code, and large program acting like a collection of very small programs.

B. *ICS: UML Modeling*

The object-oriented approach in software development is becoming increasingly popular. A significant number of object-oriented methods [13-18] have been introduced during the past several years. Since the 90's the UML [19] is accepted as the standard for the Object Oriented Modeling. Within Object-Oriented analysis and design, different UML models are employed for modeling the system from different viewpoints and at different levels of abstractions.

The simulation information we consider are related to two categories: system structure and behavior. Structural design information is an explicit description of the structure of the base-level objects. This includes the number of attributes and their data type. Behavioral design information describes the computations and the communications carried out by the base-level objects. It includes objects behavior, collaboration between objects, and the state of the objects. Structure and behavior of the system are modeled by diagrams like class diagram, sequence diagram and state diagram.

Although many diagrams exist, the two most popular are generally acknowledged to be class diagram to describe the structure and interrelationships among the classes, and sequence diagram to describe sequences of method calls among objects that together describe important usage scenarios.

Class diagrams list the fields and methods in each class, and show which classes use other classes, which inherit from others, and multiplicities of classes in various relationships with other classes. Sequence diagrams are commonly used for partially documenting the behavior of the software system as interaction patterns among objects.

In figure 1, we show, through a class diagram, the static structure of our system. It consists of a main class, the "Controller" which is the entry point to the system. The controller creates a "Road", generates "Vehicles" and starts the "Model". The outputs of the model are stored in "Data", which send them in turn to "HMI" (Human Machine Interface) to be visualized.

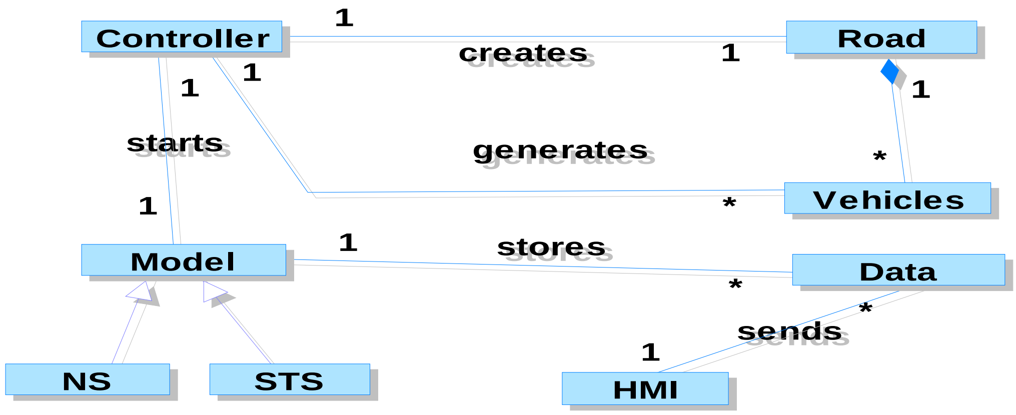

Figure 1. Class diagram representing the statistical structure of ICS. Two models are considered, the NaSch model (NS) and the NaSch with the "slow-to-stop" rule (STS)

The communication between objects in our system is illustrated with a sequence diagram of the figure 2. The interaction among objects is performed by communication called "message passing." One object sends a message to another and the receiving object then responds.

Once the user starts the system, a road is created and N cars are randomly generates (with random positions and velocities values). The vehicles are then moved according to the model rules. Output information (density ρ, average velocity <v>, flow j …) are first stored in the Data object before sending them to the HMI object to be visualized. Figure 3 and figure 4 show two examples of HMI of the ICS simulator.

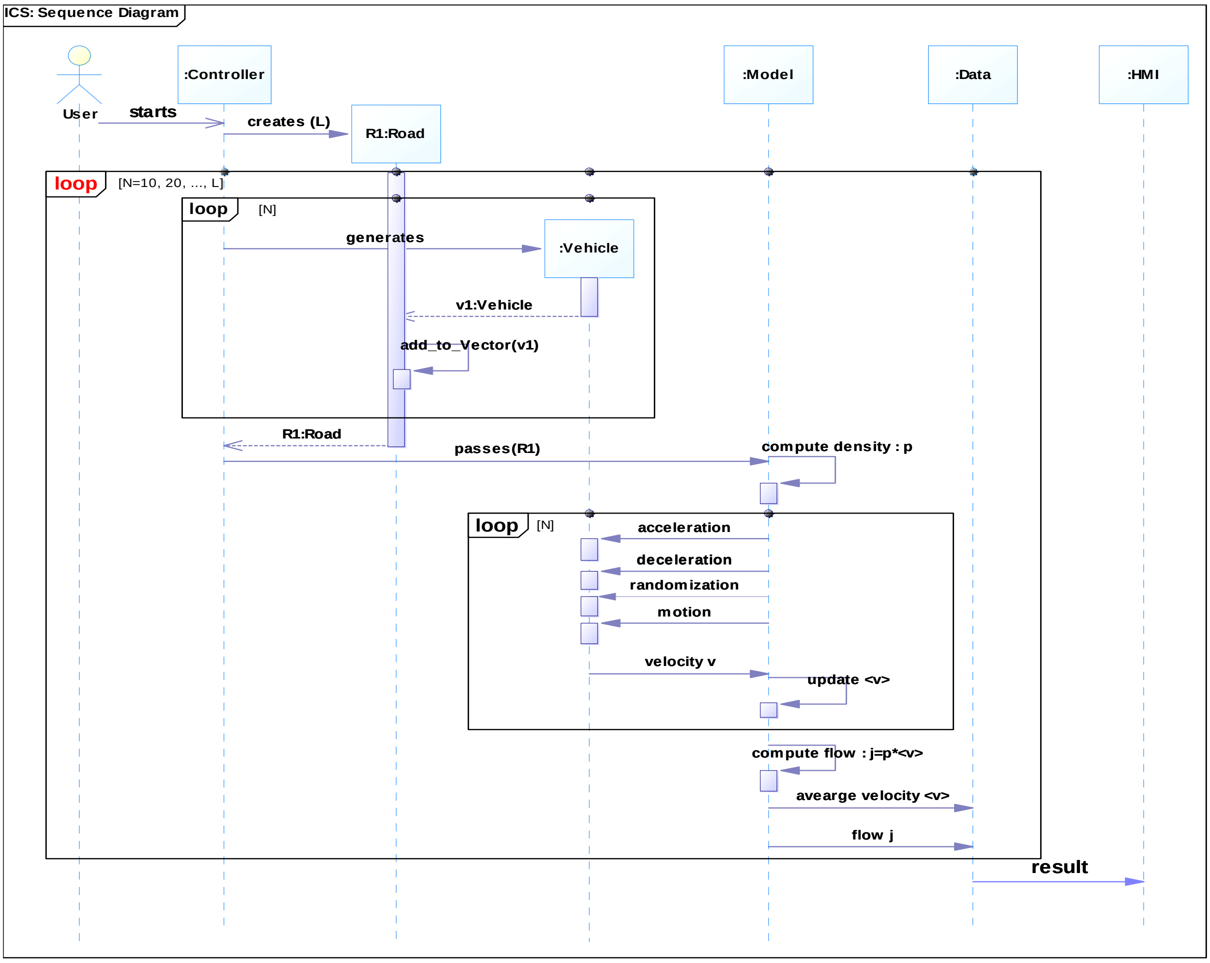

Figure 2. Sequence diagram of ICS. The diagram represents the behavioral aspect of the system in term of communications between objects of the ICS

C. Simulation Results

We use the ICS to simulate the probability of car accidents using two models, the basic NaSch model (NS) and the NaSch model with the “slow-to-stop” rule (STS). Before going on, we would like to describe our standard simulation set-up for the following observations. We simulated a system of length L=1000 cells with closed boundary conditions, i.e. the traffic is running in a loop. We start with random initial conditions. N cars are randomly distributed on the lane around the complete loop with an initial speed taking a discrete random value between 0 and v_{max}. Since the system is closed, the average density ρ=N/L remains constant with time. Next, we update the individual car velocities and positions in accordance with rules of the model (NaSch model and NaSch model with the “slow-to-stop” rule). Each density is simulated for T=2000 time steps, of which the first half (1000) were discarded to let transients die out and for the system to reach its asymptotic steady state. The table 1 shows the parameters used in this paper.

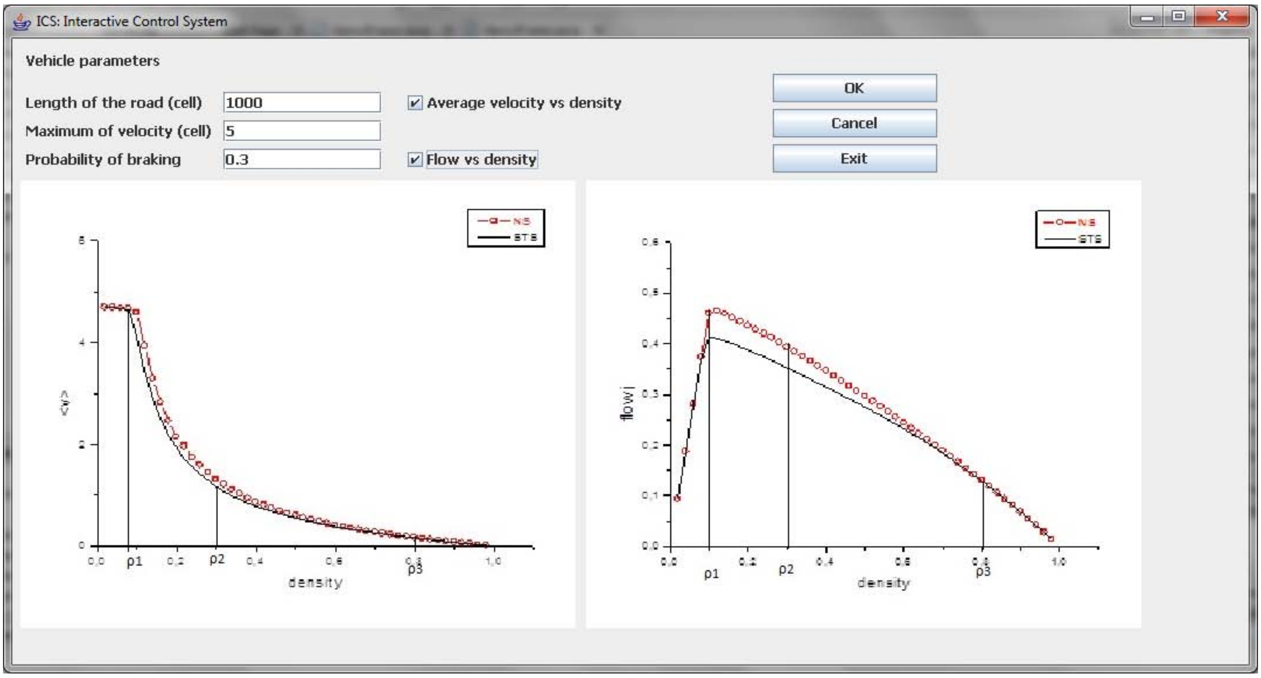

Figure 3. An example of a HMI of our system. On the left, we represent the average velocity vs the density, while on the right, we represent the fundamental diagram.

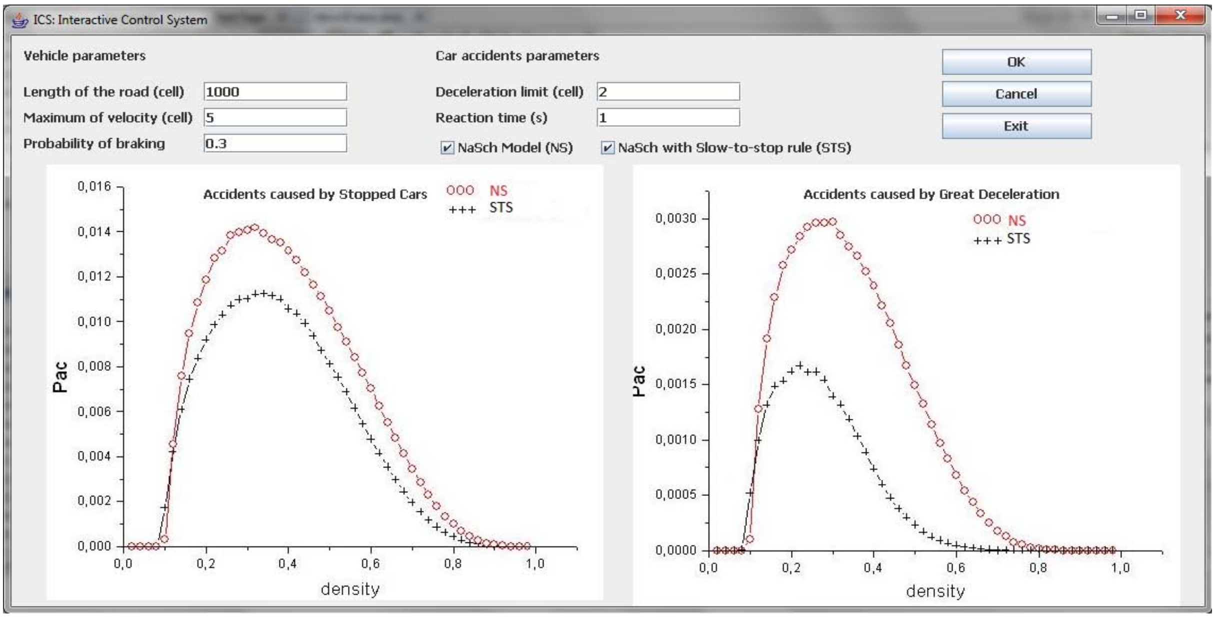

Figure 4. A HMI representing the probability, pac, of the occurrence of car accidents for the two conditions

Table 1: Parameters selected in the simulation

Length of each cell (m)	7.5
Length of each car (cell)	1
Total number of cells	1000
Maximum velocity (cell)	5
Randomization probability	0.3
Deceleration limit vd (cell)	2
Delayed reaction time τ (s)	1

Using our Interactive Control System simulator, we show an example of the experiment simulation for the study of the probability of car accident (Fig.4). The dependence of the probability of car accidents on the car density can be interpreted as follows:

In the free flow region (when the density < ρ_1), all vehicles move with the maximum velocity v_{max} as shown in figure 3. Since the density is low, the mean gap between vehicles is superior to v_{max}, thus there is no stopped cars exists. The probability of the occurrence of car accidents in this area is zero.

When the density $\rho > \rho_1$ we are in the congested traffic. In this region, we can distinguish two different regimes [20]. In the first one ($\rho_1 < \rho < \rho_2$), spontaneous jams are formed, and then dissolve after a while and the cars move with maximum velocity. The drivers tend to accelerate as much as possible. Once a jam is formed, careless drivers do not react quickly and car accidents occur frequently. In the second region of the congested traffic ($\rho_2 < \rho < \rho_3$), a super jam exists. The drivers are obliged to reduce at most their velocities. The probability in this case decreases rapidly.

Once the density exceed a certain value (when the density $\rho > \rho_3$), the velocities of most vehicles are zero in every time step. The probability of the occurrence of car accidents in this region is then zero.

Now let's move to see what effect has the "slow-to-stop" rule on the probability of the occurrence of car accidents. From figure 4, we can see clearly that the probability, pac, obtained with the STS model is very low compared to that obtained with the NaSch one. It's obvious, since the goal of the rule added to the basic model is to allow the drivers to look farther ahead and slow down gradually earlier when their cars come closer to the jam. Thus, this last kind of driving is highly recommended even though it causes a slight diminution of the total flow.

IV. Conclusion

Every system which helps to alleviate traffic jams needs information about the current traffic situation. Ideally, traffic data are collected in real-time by locally fixed detectors like inductive loops or cameras. Another way consists of using historical surveys to gather information about the traffic. Nevertheless, a lot of road networks are not adequately equipped with detection devices to gather information about the present traffic state in the whole network. A possible way to derive information for these regions is to use a simulation models, like the NaSch model or one of its variants.

The advantage of our simulation system (ICS) is on one hand, the use of the Object Oriented paradigm with the UML and Java, and on the other hand the use of traffic flow model. The model is necessary to perform the micro-simulations. Since a simulation system requires a high computation speed, a simple and efficient model is inevitable.

On the other hand, since car accidents cannot be directly measured, our ICS simulator can be an appropriate tool for the accidents analysis and the prevention strategy. In order to test the simulator, we gave some examples of the experiments simulation and we showed that "looking farther ahead" and "slowing down gradually" could be a driver's attitude that can cause a decrease of the risk of car accident.

References

[1] R. Kühne, N. Gartner, C.J. Messner and A.J. Rathi (eds.), "Traffic Flow Theory", Transportation Research Board (TRB) Special Report 165, 2nd ed. 1998.

[2] I. Prigogine and R. Herman, "Kinetic Theory of Vehicular Traffic", Elsevier, Amsterdam, 1971.

[3] Dirk Helbing, "Gas-kinetic derivation of Navier-Stokes-like traffic equations", Physical Review E 53, 2366-2381, 1996.

[4] C. Gazis, R. Herman and B. Potts, "Car-following theory of steady-state traffic flow", Operations research, vol. 7, No. 4, pp. 499-505, 1959.

[5] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic", Journal of physical I France, Vol. 2, pp. 2221-2229, 1992.

[6] A. Schadschneider and M. Schreckenberg, "Cellular automaton models and traffic flow", Physics A, (26):679{683, 1993.

[7] A. Clarridge and K. Salomaa, "Analysis of a Cellular Automaton Model for Car Traffic with a Slow-to-Stop Rule", S. Maneth (Ed.): CIAA 2009, LNCS 5642, pp. 44-53, 2009 © Springer-Verlog Berlin Heidelberg 2009

[8] N Boccara, H Fuks and Q Zeng, "Car accidents and number of stopped cars due to road blockage on a one-lane highway", Journal physics, A 30, 3329, 1997

[9] Y. Xian-Qing and M. Yu-Qiang, ”Car accidents determined by stopped cars and traffic flow”, Journal of Physics A: Mathematical and General, Vol. 35 No. 49, 2002.
[10] M. Najem, “Car accidents in cellular automata models for one-lane traffic flow”, Physical Review E 68, 036127, 2003.
[11] M. Najem, “Simulation study of traffic accidents in bidirectional traffic models”, International Journal of Modern Physics C, Vol. 21, No. 12, 2010.
[12] Y. Xian-Qing, M. Yu-Qiang and Z. Yue-Min,” Boundary effects on car accidents in a cellular automaton model”, Journal of Physics A: Mathematical and General, Vol. 37 pp. 4743–4752, 2004.
[13] I. Jacobson, “Object Oriented Development in an Industrial Environment “. In: Proc. OOPSLA. Orlando, Florida: ACM Press, pp. 183-191, 1987.
[14] G. Booch, “Object-Oriented Design with Applications”, the Benjamin/Cummings Publishing Company, Inc, 1991.
[15] D. de Champeaux, “Object-Oriented Analysis and Top-Down Software Development”, European Conference on Object-Oriented Programming, pp. 360-375, July 1991.
[16] P. Coad and E. Yourdon, “Object-Oriented Analysis”, 2nd edition, Yourdon Press Computing Series, Prentice-Hall, 1991.
[17] P. Coad and E. Yourdon, “Object-Oriented Design”, Yourdon Press Computing Series, Prentice-Hall, 1991.
[18] J. Rumbaugh et al., “Object-Oriented Modeling and Design”, Prentice-Hall, 1991.
[19] Grady Booch, James Rumbaugh, and Ivar Jacobson, “The Unified Modeling Language, User Guide”, Object Technology Series. Addison-Wesley, Reading, Massachusetts, third edition, February 1999.
[20] S. Cheybani, J. Kertesz and M. Schreckenberg, J. Phys. A31, 9787, 1998