TRAFFIC MODELING AND SIMULATION: A DEVS LIBRARY

Gabriel Wainer

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa, ON, Canada
gwainer@sce.carleton.ca

ABSTRACT

The Cell-DEVS methodology is formal modeling technique that permits defining each cell in a cell space as individual independent entity. We used Cell-DEVS to build a library that allows defining different models of traffic. We show how to model cell spaces with emerging behavior using this methodology. We present basic models and visualization tools based on 3D models in Maya.

Keywords: DEVS models, Cellular Automata, Cell-DEVS models, traffic simulation.

1 INTRODUCTION

Transit volume is growing up, and it is necessary to control, redirect and optimize it. As transit behavior is very complex to study, the application of simulation allows analyzing different scenarios for transit control strategies. Gridlock occurs on a daily basis on major routes into and out of a city and traffic jams happen all the time within major centers. Traffic engineers need to determine where the traffic is going to build up and how soon and what would be the optimal route to follow to avoid gridlocks.

A well-established method to determine such results is using modeling and simulation. If we can predict the amount of traffic flowing into a specified city section and can predict how individual drivers are going to react on city streets, then we can predict how the city traffic is going to develop in advance. A simulation can give city planners the ability to make modifications to an existing city map or even create their own city section and see the results based on parameters that they set that determine the amount of traffic that will be entering the city section and the normal destined route for the traffic when the get to the city section. Traffic simulations are useful to test traffic policies, signals, measuring the consequences of collisions or men at work, controlling pollution, avoiding traffic jams, etc. Due to the complex characteristics of these systems, models reflecting a higher number of features can provide more accurate results.

An important aspect of the transit flow models is the structure chosen to represent the streets, highways or transit lanes; this structure determines the kind of movements of vehicles. It permits to distinguish the difference between simple models that represent transit flow on one-lane roads, and the more complex that model bi-directional multi-lane roads with street intersections. The later models represent a more complete behavior, modeling for example, the exchange of vehicles between lanes and turning around a crossroad. Another important aspect to consider are the special characteristics that affect the movement of vehicles. This includes, for example, the representation of different kinds of vehicles, control signals, deviations and accidents. When a higher number of features of the traffic is included in the model, the simulated behavior is more accurate, and the results more precise.

Cellular Automata (CA) is a formalism well suited to describe this problem. A model built using CA uses discrete variables for time, space and system states (Burks and Von Neumann 1970). The states in the

lattice are updated according with a local rule in a simultaneous and synchronous way. The cell states change in discrete time steps according with a local computing function. The function considers the present cell's state and a finite set of nearby cells (called the cell's neighborhood). Several works have been proposed using cellular automata for traffic simulations.

CA are synchronous, a fact that poses several constraints in the timing precision for the models, and a discrete-time approach might be not effective. The Cell-DEVS formalism was proposed to solve this kind of problems. Cell-DEVS (Wainer 2009) extends CA providing Discrete Events Systems Specifications based on DEVS (Zeigler et al. 2000), a formalism for modeling discrete-event dynamic systems that allows for hierarchical decomposition of the model by defining a way to couple existing DEVS models. Cellular models can be described as discrete event models with explicit transport and inertial delays to model speed of the vehicle movement accurately.

Here we present new constructions for a library for traffic analysis implemented as DEVS and Cell-DEVS models. The library contains numerous models, including a one-lane one-direction traffic cellular model, a bidirectional traffic cellular model, crossings with traffic lights or stop signs and pedestrian control systems, traffic monitoring models, roundabouts, controller for a bridge with alternating traffic lights, highway tolls, highway interchanges, etc. The models are public domain and the tool and library are available at http://cell-devs.sce.carleton.ca. We also introduced advanced visualization using Maya for 3D models.

2 BACKGROUND

Nowadays, most existing techniques are based on microscopic models, which describe both the system entities and their interactions at a high level of detail (for example, a lane change could consider nearby cars, as well as detailed driver decisions). Different modeling techniques have been used to create traffic simulations, including queuing networks (Schmidt 2000), Cellular Automata (Treiber et al. 2000), software agents (Balmer et al. 2004), and other approaches, including Game Theory (Chen and Ben-Akiva 1998), Petri Nets (Tolba et al. 2005), up to fluid or electrical flow models.

Cellular Automata (CA) is a popular technique widely used for defining these kinds of models (Maniezzo 2004; Nagel et al. 2000; Nagel 2002; Esser and Schreckenberg 1997; Marinosson 2002; Rickert et al. 1996). CA define a grid of cells using discrete variables for time, space and system states (Chopard and Droz 1998, Wolfram 2002). Cells are updated synchronously and in parallel for every cell in the space according with a local rule using a finite set of nearby cells (the neighborhood). Cellular models represent a quite intuitive way of analyzing the traffic flow in detail, and they enable good visualization of the results. Nonetheless, CA are synchronous, a fact that poses precision constraints and extra computing time. The Cell-DEVS formalism (Wainer and Giambiasi 2002) was proposed to solve these problems by defining cell spaces as DEVS (Discrete Events systems Specifications) models (Zeigler et al. 2000). Using Cell-DEVS, a cell space is described as a discrete event model in which explicit delays can be used to accurately model the cell timing properties.

Cell-DEVS is an extension of DEVS that can be used for modeling and simulation of systems in a cell space. It is a combination of DEVS and cellular automata (CA) with explicit timing delays (Wainer 2009). Each cell becomes an individual atomic model and the cell space becomes a coupled model where all the cells are linked to their neighbors through input and output ports, as in regular DEVS models. A cell-DEVS model can be described by the conceptual diagram presented in Figure 1.

Each cell receives N inputs, usually from neighboring cells but they could also be provided by a regular DEVS model. When a cell receives these inputs, it triggers τ, the local computing function that will determine the next state, s'. At this moment, if the cell's future state s' is different than its current state s, it will schedule an output of its new state following a transportation delay specified by d. Whenever a cell changes state, its new state s' and scheduled time for transition are added to a local queue. Cells with transport delays will always output their new state, provided a state change has occurred. If a cell's state

does not change following the local computing function τ, it becomes passive and waits for further external events. Cells can also use inertial delays which allow pre-emption of the cell's state transition. A cell with inertial delay that does not manage to keep its new state until the next scheduled time elapses is pre-empted and foregoes its output phase for the pre-empted state transition (Wainer 2009).

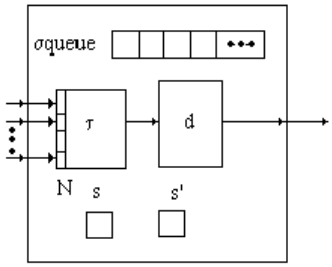

Figure 1: Cell-DEVS atomic model definition (with transport delay).

The CD++ tool implements the DEVS theory. It allows defining models according to the specifications introduced in the previous section (Wainer et al. 2001, Rodríguez and Wainer 1999). A set of independent applications related with the tool allows the user to have a complete toolkit to be applied in the development of simulation models.

The tool is built as a hierarchy of models, each of them related with a simulation entity. Atomic models can be programmed and incorporated onto a basic class hierarchy programmed in C++. A specification language allows defining the model's coupling, including the initial values and external events.

CD++ tool includes an interpreter for a specification language that allows describing the behavior of each cell of a cellular model, including its delay and neighborhood. In addition, it allows to define the size of the cell space and their connection with other DEVS models, the border and the initial state of each cell. To do so, the theoretical definitions of the Cell-DEVS formalism were used. The behavior specification for a cell is defined using a set of rules, each indicating the value for the cell's state if a condition is satisfied. The output of the model should be delayed by using a specified time. If the condition is not valid, the next rules in the list are evaluated until a rule is satisfied or there are no more rules. The main operators available include Boolean, Comparison, Arithmetic, Number types, Neighborhood values, Time, Conditionals, Angle conversion, Pseudo-random numbers, Error rounding, Predefined constants (pi, e, gravity, etc.), and many others.

3 TRAFFIC MODELING, SIMULATION AND VISUALIZATION

In this section we introduce examples of our library, built using CD++, and discuss various case studies in the field of traffic modeling and simulation. We first show an example of a discrete-event model of a crossing with traffic lights and a speeding camera, and then various Cell-DEVS models showing how to define microscopic cellular models.

Driving through an intersection after the light has turned red is probably one of the most dangerous traffic violations. According to the Insurance Institute for Highway Safety, every year, these accidents kill about 800 people and rack up an estimated $7 billion dollars in property damage, medical bills, lost productivity and insurance hikes. Collisions resulting from red-light running tend to be more severe than other intersection collisions because they usually involve at least one vehicle travelling very quickly. In the most red-light running collisions, the vehicles hit each other at right angles causing severe injuries and often leading to death. Public awareness of red-light cameras improves driving behavior.

We built a library to simulate red-light running detection, including three components built as DEVS models: electromagnetic detectors, embedded in the pavement just before the limit line, cameras and controller. Electromagnetic detectors, triggered by speeding vehicle, send signal to the controller that (if

the traffic light is red) activates the Camera which catches red-light runner. By using the speed measurements, the system predicts if a particular vehicle will not be able to stop before entering the intersection, and activates the camera. The first photo shows the rear view of the vehicle just before it enters the intersection, with the light showing red, and the second photo, taken a second or two later, shows the vehicle when it is in the intersection. The DEVS modes built for this kind of traffic modeling includes a Red Light Runner Detection (RLRD), a coupled model. RLRDs consist of two main components: an atomic Electromagnetic Detector (ED) and a coupled Camera System (CS). ED is in charge of detecting speeding vehicles and sending an activation signal. CS model is composed of two simple atomic models – Camera Controller (CC) and Camera (C). CC is responsible for detection input signals "RED On" and "ED activated" and producing the "Camera On" output signal when both input signals are detected. C, activated by an input signal from CC, captures video evidence and then sends it to a receiver. We also use a Counter for the number of violations to be sent to a Data Center model for analysis. Counter is an atomic model, responsible for counting number of violations and compute statistics. The camera system is supposed to record the vehicle license plate, the date and time, the location, the vehicle speed, and the amount of time elapsed since the light turned red and the vehicle passed into the intersection. The events can be captured as a series of photographs or a video.

Figure 2 shows the structure of a case study in which we employed the library components to build a model of an intersection between two 2-lane streets with traffic lights and the detection system. Video evidence shows the vehicle before it enters the intersection on a red light signal and its progress through the intersection. Then, the data and images are sent wirelessly to the relevant law enforcement agency. We also include a Traffic Lights (TL) model, a coupled model consisting of two connected atomic models that create "RED On" signals.

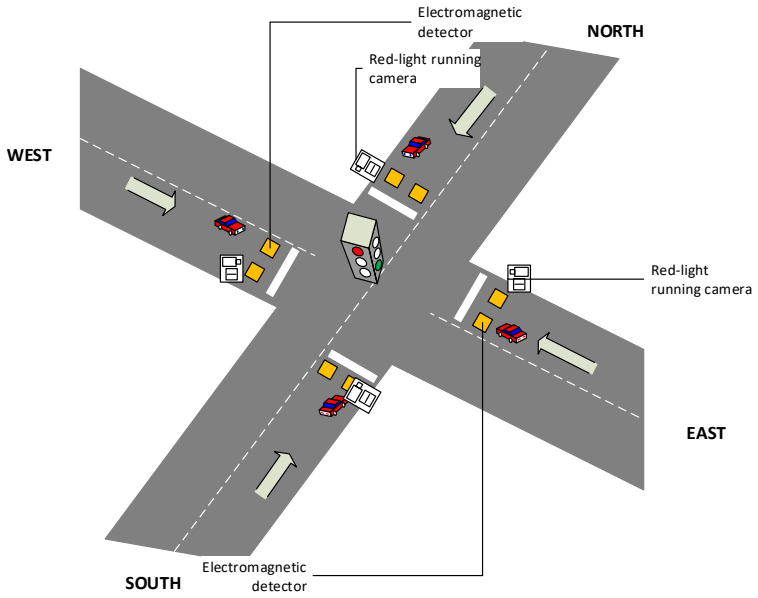

Figure 2. Intersection equipped by the Red-Light Running Detection System.

The top-level model, presented in Figure 3 is a coupled model composed of six basic components: the Traffic Lights model (TL), four identical Red Light Runner Detection (RLRD) models (one for each direction) and Counter. The four RLRDs represent four directions of intersection (NORTH, SOUTH, EAST, and WEST) respectively.

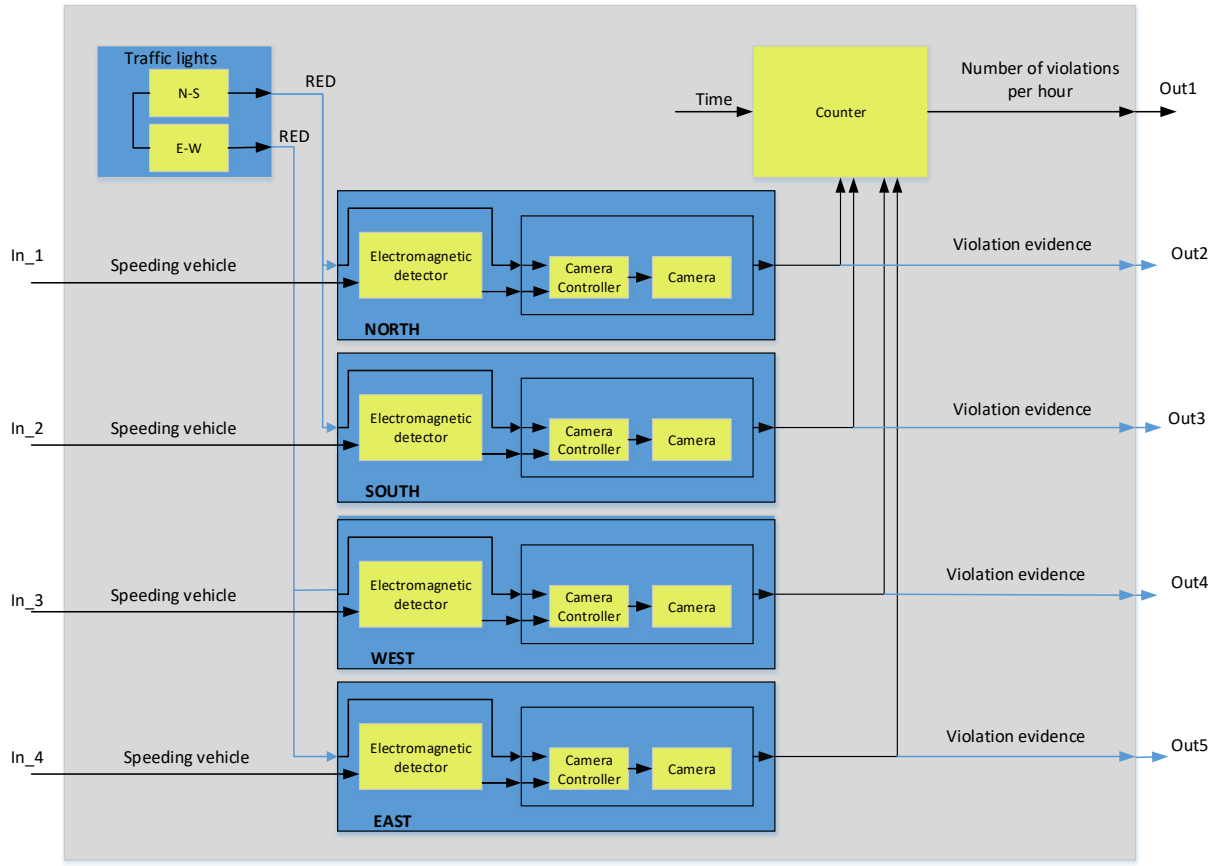

Figure 3. Structured model of the Red-Light Running Detection System.

The library contains a number of atomic and coupled models that allow defining this model. We will show the details of one of the Atomic Models, in this case, the Traffic Lights Atomic Model, which represents two traffic lights working together on the intersection. The combination of traffic light cycles and the signal phase duration is represented in the following Table 1. As we can see, from the table duration of the Green signal is 60 sec, duration of the Yellow signal – 30 sec. Duration of the Red signal equal to the sum of Green and Red, 10 sec.

Table 1. Light cycles and signal phase durations.

State	Direction		Time, sec	Outputs	
	EW	NS		redEW	redNS
1	Green	Red	60	0	1
2	Yellow	Red	30	0	1
3	Red	Green	60	1	0
4	Red	Yellow	30	1	0

This model can be formally defined as

$$Trafficlights=<X, S, Y, \delta int, \delta ext, \lambda, ta>$$

$X=\Phi$;

Y= {redEW, redNS,};

```
S={ EW_GREEN_NS_RED, EW_YELLOW_NS_RED, EW_RED_NS_GREEN,
        EW_RED_NS_YELLOW };

δint:
    case EW_GREEN_NS_RED:
        this->holdIn(active, greenTime); break;
    case EW_YELLOW_NS_RED:
        this->holdIn(active, yellowTime); break;
    case EW_RED_NS_GREEN:
        this->holdIn(active, greenTime); break;
    case EW_RED_NS_YELLOW:
        this->holdIn(active, yellowTime);

λ:
if (curLightState == EW_RED_NS_YELLOW)
        curLightState = EW_GREEN_NS_RED;
    else
        curLightState++;

    switch(curLightState)
        case EW_GREEN_NS_RED:
            sendOutput( msg.time(), redEW, 0 );
            sendOutput( msg.time(), redNS, 1 );
            break;
        case EW_YELLOW_NS_RED:
            sendOutput( msg.time(), redEW, 0 );
            sendOutput( msg.time(), redNS, 1 );
            break;
        case EW_RED_NS_GREEN:
            sendOutput( msg.time(), redEW, 1 );
            sendOutput( msg.time(), redNS, 0 );
            break;
        case EW_RED_NS_YELLOW:
            sendOutput( msg.time(), redEW, 1 );
            sendOutput( msg.time(), redNS, 0 );
                    break;
}

ta (GreenTime) = 60 sec.
ta (YellowTime) = 30 sec
```

After implementing the model in CD++, we can use it for different scenarios. The following Table 2 shows a simulation scenario to test the library, and the simulation results.

Table 2. Testing TrafficLights

Time	Direction		Outputs	
	EW	NS	redEW	redNS
00:00:00:000	Yellow	Red	0	1
00:00:03:000	Red	Green	1	0
00:00:10:000	Red	Yellow	1	0
00:00:13:000	Green	Red	0	1
00:00:20:000	Yellow	Red	0	1
00:00:23:000	Red	Green	1	0

```
TrafficLights outputFunction
Timestamp: 00:00:10:000 curLightState: 3
EW_RED_NS_YELLOW, redEW: 1, redNS 0
TrafficLights internalFunction
Timestamp: 00:00:10:000 redew 1 redns 0
TrafficLights outputFunction
Timestamp: 00:00:13:000
curLightState: 0 EW_GREEN_NS_RED, redEW: 0, redNS 1
TrafficLights internalFunction
Timestamp: 00:00:13:000  redew 0  redns 1
TrafficLights outputFunction
Timestamp: 00:00:20:000 curLightState: 1
EW_YELLOW_NS_RED, redEW: 0, redNS 1
TrafficLights internalFunction
Timestamp: 00:00:20:000 redew 0   redns 1
TrafficLights outputFunction
Timestamp: 00:00:23:000  curLightState: 2
EW_RED_NS_GREEN, redEW: 1, redNS 0
TrafficLights internalFunction
Timestamp: 00:00:23:000 redew 1  redns 0
```

As we can see, the model reproduces the case study according to the required specifications, allowing one conducting studies on the speed of traffic light configuration. Such formal model and its simulation results could be better analyzed using 3D visualization. In this case, we built a Maya extension to allow 3D visualization of the models. Maya is a powerful application for three dimensional modeling and animation. To do that, we built Python script to interface a JASON file extracted from CD++. When the user developed python script is run, it should automatically execute the files and start animation. Below is Figure 4, which represents the software stack used.

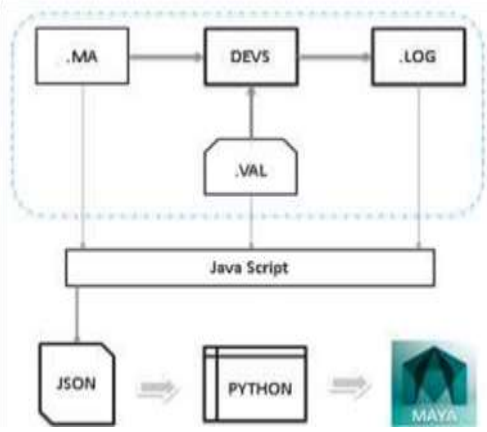

Figure 4: Software Architecture and tools for 3D visualization

MASH is a procedural node-based toolset that allows designers and artists to combine nodes to quickly create unique animations and effects. Its intuitive and easy-to-use procedural instancing and animation toolset is ideal for creating complex motion graphics, UI design, environments, animation, and effects. Maya now comes with Arnold, one of the best renderers available, to help make the rendering process faster and deliver better results. The animation workflow in Maya has seen some big changes the past few releases, and now, with a new Time Editor plus an improved Graph Editor, animating in Maya is faster and more artist-friendly. Workspaces define the configuration of windows and panels, providing more flexibility to reconfigure panel layouts; you can open, close, and move windows and panels, dragging and dropping to dock and undock, almost anywhere in the interface. We used these tools to build the following components:

- **Python parser**: we built an interface from JSON to Maya, which converts a JSON - XML like file generated by CD++ into a language Maya can understand. Maya has inbuilt objects which can be pulled in when required into the scene. If we need any new objects, we can generate them using the basic objects given by default in the tool. For our project, we created a city landscape, used to initialize MAYA file will initially have objects, running the python script, and converting the JASON file into timeline **keyframes**. Maya keyframes are used for animation (we also refer to them as "keys"). Keyframes are basically a marker used to specify an object's position and attributes at a given point in time.

- **Traffic flow**: To illustrate a micro scaled model of a DEVS traffic flow model, we show an example in Figure 5. Traffic flows from the left to right, and it will be stopped to let the right-hand side traffic to pass as per traffic rules.

Figure 5: Traffic execution in a Cell-DEVS model

Based on this basic behavior, we define a 3D visualization including trees, plants, street vendor shops and high rise buildings to simulate a look and feel of a suburban city.

Over imposing the cellular models into the 3D visualization results is simple; the formalization of the DEVS and Cell-DEVS models allows the software integration to be easily done and the analysis of the results in an immersive environment achievable. This can be seen in Figure 6.

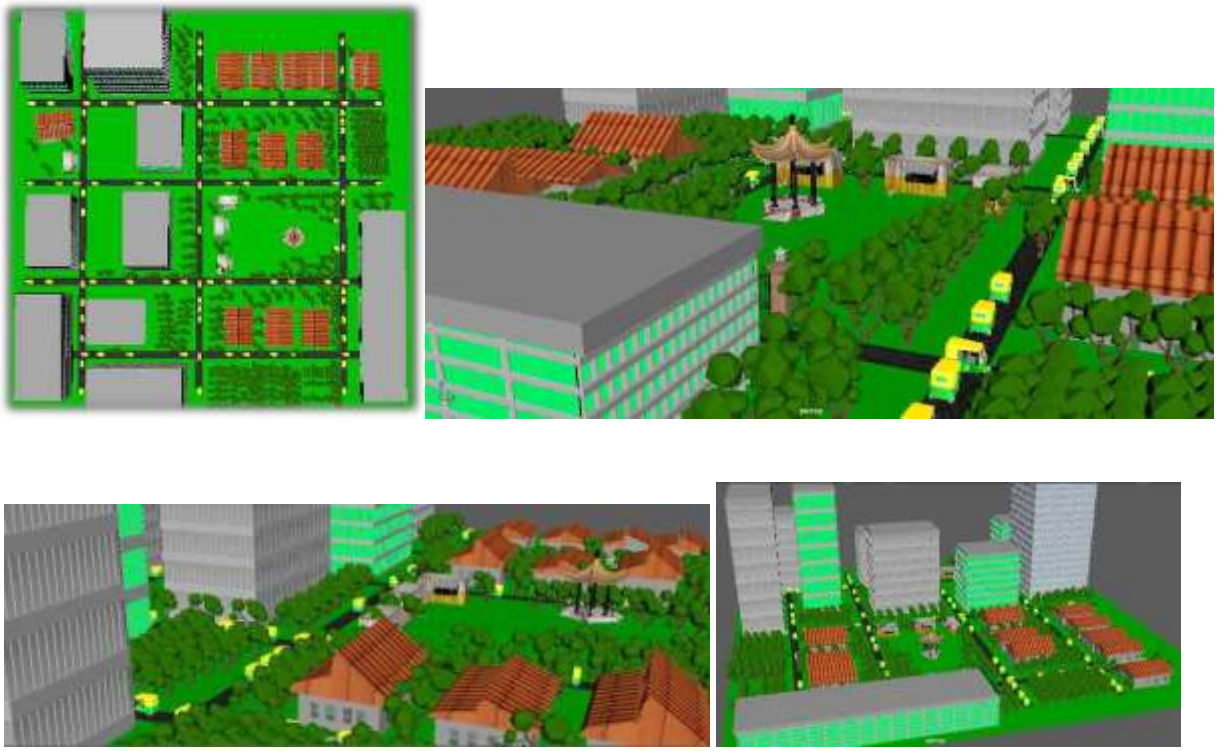

Figure 6: City Visualization view

4 CONCLUSION

We introduced examples of a DEVS library which provides an application-oriented specification language, which allows the definition of complex traffic behavior using simple rules for a modeler. The models are formally specified, avoiding a high number of errors in the application, thus reducing the problem solving time. The high level specification of the problem to be modeled reduces the developing efforts, as the techniques presented permit to automatically build the structure for coupled models, and to generate rules for atomic models. In this way, changes in the system specification can be done in a simple fashion, without spending time in coding or testing every proposed solution to existing problems. In this way, a traffic analyzer can focus in the problem solving task, avoiding implementation or low level details.

The constructions are mapped into DEVS and Cell-DEVS. These translated models are formally specified, and its correctness was proved, avoiding errors in their definition. Using this approach we could obtain:

- Efficiency: by describing a high level specification of the problem to be modeled, we have reduced the effort needed in developing the application. The models execute using a discrete-event approach, which provide higher precision and speedups than the discrete time approaches. Likewise, the proposal automatically builds the structure for coupled models, generates rules for atomic models. In this way, changes in the system specification can be done in a simple fashion, without spending time in coding or testing every proposed solution to existing problems.
- Adaptation: new rules can be easily incorporated, as we showed with different examples here (traffic lights, truck behavior, potholes, etc.).
- Abstraction: the specifications were translated into executable models. In this way, a traffic analyzer can focus in the problem solving task, avoiding implementation or low level details.

REFERENCES

Burks, A.W. 1970. “Von Neumann’s self-reproducing automata”. In: Burks, A.W. (ed.) Essays on Cellular Automata, pp. 3–64. University of Illinois Press, Champaign.

Chen, O.; Ben-Akiva, M. 1998. "Game Theoretic formulation of the interaction between dynamic traffic control and dynamic traffic assignment". Technical Report, ITS, M.I.T.

Chopard, B.; Droz, M. 1998. “Cellular automata modeling of physical systems”. Cambridge University Press.

Chopard, B.; Dupuis, A.; Luthi, P. 1997. “A CA Model for Urban Traffic and its applications to the city of Genoa”. Proceedings of Traffic and Granular Flow. Duisburg, Germany.

Chopard, B.; Queloz, P. A.; Luthi, P. 1995. “Traffic Models of A 2d Road Network”. Proceedings of the 3rd CM Users’ Meeting. Parma. Italy.

Chopard, B.; Queloz, P. A.; Luthi, P. 1996. “CA Model of Car Traffic in Two-Dimensional Street Networks”. J. Phys. A, Vol. 29.

Davidson, A.; Wainer, G. a. 2000. "Specifying traffic signs in traffic models". A. Davidson, G. Wainer. In Proceedings of AI, Simulation and Planning in High Autonomous Systems, AIS'2000. Tucson, AZ.

Davidson, A.; Wainer, G. b. 2000. "Specifying truck movement in traffic models using Cell-DEVS". In Proceedings of the 33rd Annual Simulation Symposium. IEEE Press. Washington, DC.

Díaz, A; Vázquez, V.; Wainer, G. 2001. "Application of the ATLAS language in models of urban traffic". In Proceedings of 34th IEEE/SCS Annual Simulation Symposium. Seattle, WA.

Esser, J.; Schreckenberg, M. 1997. “Microscopic simulation of urban traffic based on CA”. Intl. Journal of Modern Physics C8, 1025-1036.

Lee, J.; Chi, S. 2005. “Using Symbolic DEVS Simulation to Generate Optimal Traffic Signal Timings”. SIMULATION. 81: 153-170.

Maniezzo, V. 2004. “CA and roundabout traffic simulation”. Proceedings of Sixth International conference on Cellular Automata for Research and Industry. Amsterdam, Netherlands.

Marinosson, S. 2002. “Simulation of the Autobahn Traffic in North Rhine-Westphalia”. Proceedings of 5th International Conference on Cellular Automata for Research and Industry. Geneva, Switzerland.

Nagel, K. 2002. “Cellular Automata models for transportation applications”. Proceedings of 5th International Conference on Cellular Automata for Research and Industry. Geneva, Switzerland.

Nagel, K.; Esser, J.; Rickert, M. 2000. “Large-scale traffic simulations for transport planning”. Annual Reviews of Computational Physics VII (World Scientific, Singapore, 2000). [22], 151-202.

Nagel, K.; Stretz, P.; Pieck, M.; Leckey, S.; Donnelly, T.; Barret, C. 1998. “TRANSIMS traffic flow characteristics”. *Transportation Research Board, 77th Annual Meeting.* Washington, DC.

Rickert, M.; Nagel, K.; Schreckenberg; M.; Latour, A. 1996. “Two Lane Traffic Simulations using CA”. Physica A 231, 534-550.

Sadoun, B. 2003. “An Efficient Simulation Methodology for the Design of Traffic Lights at Intersections in Urban Areas”. SIMULATION, 79: 243 - 251.

Schmidt, M. 2000. "Decomposition of a traffic flow model for a parallel simulation". In Proceedings of AI, Simulation and Planning in High Autonomous Systems, AIS'2000. Tucson, AZ.

Treiber, M.; Hennecke, A.; Helbing, D. 2000. “Congested Traffic States in Empirical Observations and Microscopic Simulations”. Physical Review E 62, 1805-1824.

Wagner, P.; Nagel, K.; Wolf, P. 1997. “Realistic Multi-line traffic rules for cellular automaton”. Physica A, 234:687.

Wainer, G. 2006. “ATLAS: a language to specify traffic models using Cell-DEVS”. In Simulation Modelling Practice and Theory. Vol. 14, pp. 313–337

Wainer, G. 2002. “CD++: a toolkit to define discrete-event models". *Software, Practice and Experience*. Wiley. Vol. 32, No.3. November 2002. pp. 1261-1306.

Wainer, G. 2009. “Discrete-Event Modeling and Simulation: a Practitioner’s approach”. CRC Press, Taylor and Francis

Wainer, G.; Borho, S.; Pittner J. 2004. "Defining and visualizing models of urban traffic". In Proceedings of the SCS 1st Mediterranean Multiconference on Modeling and Simulation. Genoa, Italy.

Wainer, G.; Giambiasi, N. 2002. "N-Dimensional Cell-DEVS". G. Wainer, N. Giambiasi. In Discrete Event Systems: Theory and Applications, Kluwer. Vol. 12, No. 1. pp. 135-157.

Wolfram, S. 2002. “A New Kind of Science”. Wolfram Media.

Zeigler, B.; Kim, T.; Praehofer, H. 2000. "Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems". Academic Press.

AUTHOR BIOGRAPHIES

GABRIEL A. WAINER, FSCS, SMIEEE, received the M.Sc. (1993) at the University of Buenos Aires, Argentina, and the Ph.D. (1998, with highest honors) at UBA/Université d’Aix-Marseille III, France. In July 2000, he joined the Department of Systems and Computer Engineering at Carleton University (Ottawa, ON, Canada), where he is now Full Professor and Associate Chair for Graduate Studies. He has held visiting positions at the University of Arizona; LSIS (CNRS), Université Paul Cézanne, University of Nice, INRIA Sophia-Antipolis, Université de Bordeaux (France); UCM, UPC (Spain), University of Buenos Aires, National University of Rosario (Argentina) and others. He has published around 400 research articles and five books in the field of Modeling and Simulation. He is one of the founders of the Symposium on Theory of Modeling and Simulation, SIMUTools and SimAUD. Prof. Wainer was Vice-President Conferences and Vice-President Publications, and is a member of the Board of Directors of the SCS. Prof. Wainer is the Special Issues Editor of SIMULATION, member of the Editorial Board of IEEE Computing in Science and Engineering, Wireless Networks (Elsevier), Journal of Defense Modeling and Simulation (SCS). He is the head of the Advanced Real-Time Simulation lab, located at Carleton University's Centre for advanced Simulation and Visualization (V-Sim). He has been the recipient of various awards, including the IBM Eclipse Innovation Award, SCS Leadership Award, and various Best Paper awards. He has been awarded Carleton University's Research Achievement Award (2005, 2014), the First Bernard P. Zeigler DEVS Modeling and Simulation Award, the SCS Outstanding Professional Award (2011), Carleton University’s Mentorship Award (2013), the SCS Distinguished Professional Award (2013), and the SCS Distinguished Service Award (2015). He is a Fellow of SCS.