DISTRIBUTED REAL-TIME RAILWAY SIMULATOR

Mihai Hulea, Camelia Avram, Tiberiu Letia, Dana Muresan, Sergiu Radu
Technical University of Cluj-Napoca, C. Daicoviciu street, nr. 15, Cluj-Napoca, Romania
{mihai.hulea, camelia.avram, tiberiu.letia}@aut.utcluj.ro

KEYWORDS
Railway traffic, real time simulation, distributed systems.

ABSTRACT

The problem of control and management of railway transportation is a complex task with major outcomes in a modern society. Trains are suitable for transporting peoples and goods with a good trade-off between cost and rapidity. A railway system consists of a network of tracks, list of stations, safety devices (signals, sensors, etc) and a set of trains. Trains moves from one station to other along the network. Distributed applications are applicable in case of large systems.

INTRODUCTION

Railways are perceived as the most efficient means of mass passenger transportation. In the case of land based freight movements, railways are usually at a competitive advantage relative to road transport for the non-urban medium to long distances, bulk and containerized tasks. The operations of railway systems involve multi-disciplinary practices, ranging from business to transport operations and engineering (Ho et al 2004).

Many railway systems are still state-owned, but the privatization of various extents has been going on in many countries (Zimmermann et al 2003). The newly evolved private companies assume different roles within the operational chart of a railway system. Indeed, more than one company may take on the same role and compete with each other, which is one of the supposed advantages of privatization. A number of non-privatized railway lines are also contemplating decentralization in some way so that local authorities or contracting companies are running the rail services.

As a result, there are many parties, such as track owners and service providers, working together (collaborating and/or competing) to provide such services that the overall business and engineering objectives are considered and balanced, as well as fulfilling their own interests and duties. In order to study the behavior of this system with multiple, interactive and autonomous parties, agent-oriented technology offers the framework for modeling. Each party is represented by an independent agent who is equipped with its objectives, intelligence and autonomy (Ho et al 2004; Cuppari et al 1999; Faber et al 2006).

The increase demand of short-term train schedules by Transport Operators highlights the necessity of automated tools for train traffic decision support. When the number of trains running on a railway line and the availability of tracks are known on day by day basis, decision support systems can help in maximizing the demand granting and optimizing the traffic flow. (Cuppari et al 1999)

The issue in railway simulation is to let trains move through the network based on their time table and to check if deadlocks appears. Also the simulator can be used to check the performances of different dynamic control algorithms before implement them in the real field. The simulators can be used also for training the railway operators.

As stated in (Ferschea 2005) the fundamental approaches for simulating systems are continuous vs. discrete; event and time driven vs. event driven. In continuous event simulation the state change continuously in time, while in a discrete event simulation the events happen instantly at a fixed moment in time. The proposed approach of this paper is a time driven simulation in which the simulator update the state of the system at fixed point in time based on a given pulse clock. For solving very large railway simulation problems a distributed architecture approach must be considered. The main reasons for this are: scalability, performance and reliability.

The paper proposes a distributed time driven generic simulation system suitable for testing various routing and planning algorithms. By generic is meant that the simulator is able to take a configuration as input in form of the XML structures therefore not being tied by a static railway structure. The simulator updates time in discrete steps. One basic requirement needed to be fulfilled by the time update method is to synchronize it's time update interval with other simulator instances. It should not be possible for one simulator to progress faster than other simulators. Taking into account the discrete nature of the proposed simulator, each entity in the system should update his state on each time step. Tow possible approaches can be identified to accomplish this. This first approach is a parallel update where all entities update states concurrently. With this method extra care should be taken in order to synchronize updating threads and to prevent data inconsistency. The second approach is a sequential method where all entities are updated one after another. The proposed simulation architecture uses the sequential method.

The simulator provides a structure communication module through which other applications (like

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

controller, monitoring and information and advisory system) can send commands to the railway objects (switches, signals, etc.) and read the current state of the different railway objects (sensors, signals, trains, etc). The system also integrates a GPS emulator which can be queried in order to obtain current train coordinates and speed. The simulator instances communicate through a communication and coordination module.
The issue of railway simulation is to virtually let trains run through the network and to check whether the timetable is verified or stable (H. Schlenker 2005).

STATE OF THE ART

Modeling behavioral characteristics is a key issue of Rail way Intelligent Safety Guarantee System (RISGS) presented in (Cai et al 2006). A multi-agent simulation system in RISGS is considered and the proposed model is evaluated in a simulated experiment. This model incorporates the philosophy of object oriented concepts, available Petri-nets analysis tools and multi-agent techniques. The analysis results indicate this method can provide effective judgments without deadlocks, and improve complicated characteristics description about intelligence, autonomy and security.
Using parallel processing reduce the computing time and therefore is suitable for building real-time simulators and present different issues related to solving a power distribution system with parallel computing based on a multiple-CPU server and they concentrate, in particular, on the speed-up performance (Fun et al 2000). The model presented in [Bon, 2000] was initially based on the track analyzer method developed by Volpe National Transportation Systems Center, but was significantly enhanced in order to provide accurate predictions over a wider variety of vehicle behavior and to fulfill the real time constraints. Several models are described in literature and some programming constraints for the routing and scheduling of trains running through a junction are taken into account (Rodriguez 2005). The model uses input data from relevant time events of train runs calculated by a simulator. The model can be integrated into a decision support system used by operators who make decisions to change train routes or orders to avoid conflicts and delays.
Hybrid Petri nets can be used to simulate the traffic and to evaluate the performance of the control system. (Kaakai et al 2007) propose a simulation model based on hybrid Petri nets able to carry out performance evaluation procedures in order to increase the traffic safety.

DESIGN OF THE SIMULATOR

Railway structure modeling

XML Schema is used to describe the structure of the document containing the railway network structure. In this section the elements used to describe the network are presented.
The root of the XML document is the *railmap* element which contains all other elements which define a railway network.
The following types of railway elements are defined: gate, connectionGate, segment, signal, switch and sensor. Each element has a unique id and is stored in the element ID attribute. Also all elements have tow attributes (posX and posY) which store the location of the elements and are used for displaying them inside the simulator graphical user interface.

A gate represents a connection point between tow segments. A special type of *gate* is a *connectionGate* which is used to define connectivity between segments located in the networks managed by different simulators. This type of gates are located at the border of the railway network, and when the train pass a *connectionGate* it will be transferred to the remote network in a position which correspond to the remote *connectionGate* identified by the local *connectionGate*.
A segment represents a railway track and is delimited by tow gates named *gate1* and *gate2*. A convention is maid that the moving direction from *gate1* to *gate2* is the positive direction and is coded with the value 1 while moving from *gate2* to *gate1* is considered negative direction and is noted with -1.
The railway switch is modeled by *switch* element. A *switch* element is composed of a set of gates and has at a moment of time one active connection which allows a train to pass from one segment to another segment. The switch can be commuted in order to change the connected segments.

The *signal* element models a railway semaphore. The signals can have 2 possible states: green state and red state. Signals are attached to segments and are identified by the gate where are located and by the direction from which they can be seen.
The *sensor* element models a presence sensor which can be installed on any position on the road network. The sensor is activated when the train passes over it.
Based on the presented XSD schema, the XML files are created representing various railway network structures. When a simulator instance is launched it will load a railway network located in a XML file.

System Architecture

The paper proposes a distributed simulator architecture in which each station in a road network is handled by one simulator. The simulators are interconnected and communicate between each others. The UML component diagrams in Figure 1 presents the general simulator software architecture.

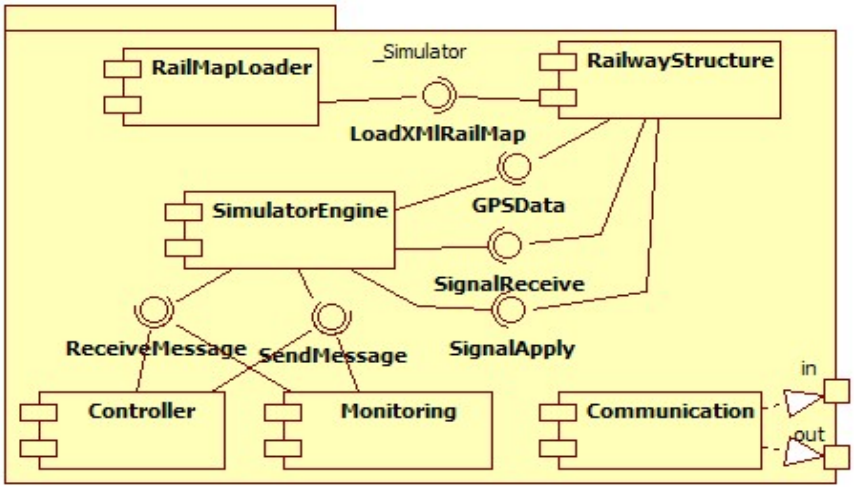

Figure 1: Simulator general software architecture.

The railway network structure is loaded from an XML file by the configuration module. The core of the configuration module is the RailMapLoader which uses a XML parser for interpreting the structure XML file and creating a structure of objects representing the railway network. The UML diagram presented in the Figure 2 describes the structure of objects used to model the railway network.

be sent to the corresponding remote simulator as stated in the network structure connection rules (this rules are set in the connectionGate elements presented in the previous section).

Moving trains and controlling structure states inside the network are accomplished by the simulator engine using the TrainMove and CollisionDetection modules. On each step the next position of the trains will be

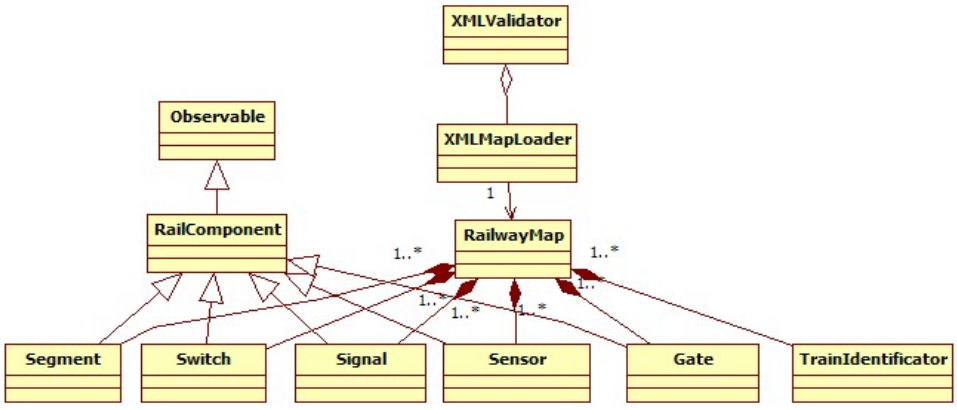

Figure 2: Railway structure modeling classes

The RailwayMap class is a container class which stores the structure of the railway network and the current state. The simulator engine accesses the internal data and alters the state of the network through a set of function defined in the following modules: StructureQuer, DataTransformtion, SingalReceive and SignalApply.

The communication between simulator instances is realized through CommServer and CommClient modules which use TCP\IP protocol in order to exchange messages. The serialization mechanism is used for sending messages. When a train reach the border of the network controlled by a simulator it will

calculated taking into account the following aspects: the dynamic train characteristics and the current status of the railway objects (the state of signals, semaphores and switches). The CollisionDetection implements an algorithm for detecting collision between trains which are moving on the same segment.

The communication model

The communications between distributed simulators are implemented at tow levels. At the first level it is implemented a communication layer between simulator engines. When a train leaves a zone controlled by a simulator it will be sent through the simulator engine connection to the corresponding neighbor simulator. At

the second level it is implemented a communication layer between controllers. At this level messages are exchanged between controllers in order to make path reservations. In Figure 3 the communication between simulators is presented.

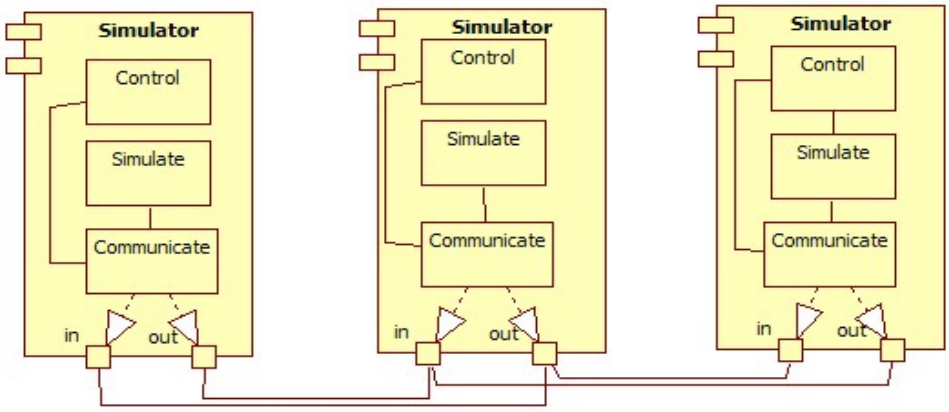

Figure 3. Simulators communication.

In this paper an asynchronous message passing model for implementing the distributed communication mechanism is proposed. This approach as been used both for implementing the communication at the simulator engine level but also at the controller level. The components will exchanges messages in an asynchronous manner.

In order to model task and threads the following stereotypes have been defined:

- <<PeriodicThread>>, which represent activities which are executed with a given time period;
- <SporadicThread>> , which represents activities which are executed each time an event occurs;

For communication between threads the following stereotypes have been defined:

- <<PriorityQueue>>, this is an unbounded queue in which elements are ordered based on an priority attribute associated with each element;
- <<Buffer>>, this is a simple buffer on which data are ordered based on the FIFO rule;

Based on the stereotypes defined before, the UML object communication diagram for the simulator system is presented in Figure 4.

The messages are passed between threads using unbounded buffers and queues. For send commands to the railway structure elements, and for receiving data about the state of the railway structure elements FIFOBuffers are used. The PriorityQyeye queue are used for exchanging messages between working threads (simulator SimulatorEngine and Controller) and communication threads (Sender and Receiver). Messages between distributed controllers are exchanged using CSender and CRecevier threads. When a new message must be sent, it must be added in the COutQueue queue or the corresponding SOutQueue, from where is get by the CSender or SSender and is sent to the destination. When a new message is received by SReceiver threads or the corresponding CReceiver, it is put in the CInQueue or SInQueue queue, from where is read by the Controller or Simulator Engine thread.

Time modeling

The method chosen to update time in the simulators is to use discrete time steps. This means that time is not updated continuously but in blocks of a certain interval. The time updates are executed by calling a tick method in the simulator engine. Each entity new state is calculated based on (e.g. a train updates its position, velocity and acceleration) the previous state and the time passed since last tick. The component responsible for calling the tick method is the Timer.

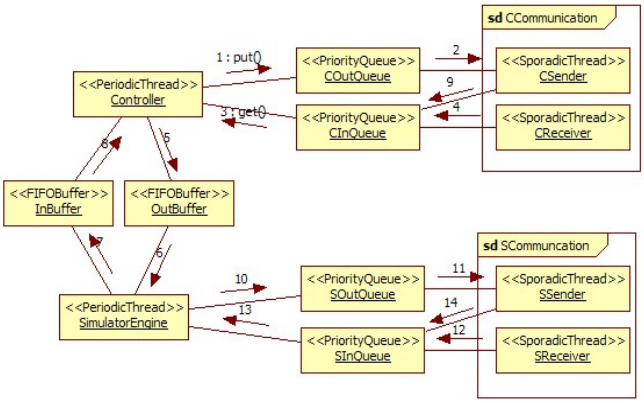

Figure 4. Object communication diagram.

In order to synchronize time of tow or more simulator instance, a time server coordinate all the simulators timers and notify each simulator timer when the simulator must advance at the next step.

SIMULATOR IMPLEMENTATION

The simulator application has been implemented in Java language using JDK v1.6.0. The implementation has been tested on Windows XP and Linux RedHat platforms.

In Figure 5 a screenshot of the simulator graphical user interface is presented. The railway network managed by the simulator is presented into a window. We are using color codes to represent states of structure elements. For example a free segment is drawn in green while a segment on which is at least one train is drawn in red. A manual control interface is available to the operator from which he can change the state of structure elements, and also can control trains.
The screenshot in Figure 6 represents the structure view window. In this window the details of the simulated structure are presented to the operator. For each structure element a separate table is displayed containing the attributes values.

The simulator architecture has been designed with the goal to be used for testing various controls and routing algorithms. In order to provide this functionality the simulator implements tow interfaces through which other modules like controller, monitoring or information systems can have access to the simulation data or can interact with the simulated railway structures.

CONCLUSIONS

This paper highlights the benefits of applying distributed techniques to rail transportation system modeling.
The proposed simulator enables the handling of more complex problems than the existing technology can handle. In the case of railroads, this might include routing more trains over more tracks, whereas traditional movement planning systems are able to plan the movement of only one train at a time. Another advantage of this simulator is that it enables more rapid adaptation to alternative schedules because of changes in the environment (e.g., a track blocked by an accident) and does not require a total reassessment.
The proposed system architecture is distributed since several train stations are connected and on each of it a simulator railway is started.
The communication will be realize on each layer (control, monitoring, information, simulation) between two simulator railways and between different layers inside of a simulator railway.
Using a good scheduling algorithm the traffic can be increase and to test the result is better to use a proper distributed simulator.

REFERENCES

Bernaer, S.; E. Burke; P. De Causmaecker; G. Vanden Berghe; and T. Vermeulen, 2006. “A Multi Agent System to Control Complexity in Multi Modal Transport”, *The IEEE Simulation Tran*.

Bonaventura, C.S.; J.W. Palese; and A.M. Zarembki, 2000. “Intelligent system for real-time prediction of railway vehicle response to the interaction with track geometry”, *Railroad Conference, Proceedings of the 2000 ASME/IEEE* Joint Volume, Issue, 2000 Page(s):31 – 45.

Cai, G.; Z. Zhang; L. Jia; and Y.Ye, 2006. “A Multi-Agent Model of Railway Intelligent Safety Guarantee System, (PRC)”, *From Proceeding (523) Computational Intelligence*.

Cuppari, A.; P.L. Guida; M. Martelli; V. Mascardi; and F. Zini. 1999. “An Agent Based Prototype for Freight Trains Traffic Management”, *Proc. of FMERail Workshop*.

Faber J; and R. Meyer. 2006. “Model Cheking Data-Dependent Real-Time Properties of the European Train Control System”, *Proceedings of the Formal Methods in Computer Aided Design.*

Ferschea, A. 2005. “Parrallel and distributed simulation of discrete events systems”, *Handbook of Parallel and Distributed Computing.* McGraw-Hill.

Fung, Y.F.; T.K. Ho; and W.L. Cheung. 2000. “Real-time simulation for power systems based on parallel computing-an empirical study”, *International Conference on Advances in Power System Control,* Operation and Management, Volume 2.

Gambardella, L.M.; A.E. Rizzoli. 2005. “Agent-based Planning and Simulation of Combined Rail/Road Transport”, *Mathematical and Computer Modelling*.

Ho, T.K.; L. Ferreira; and K.H. Law. 2004. “Agent applications in rail transportation”, *Proc of International Conference on Intelligent Agents Web Technologies and Internet Commerce*, pp. 251-260, Vienna.

Kaakai, F.; S. Hayat; and A. El Moudni. 2007. “A hybrid Petri nets-based simulation model for evaluating the design of railway transit stations”, *Simulation Modeling Practice and Theory*, Science Direct.

Rodriguez, J. 2005. “A constraint programming model for real-time train scheduling at junctions”, *Transportation Research Part B 41* pp 231–245, Science Direct.

Schlenker, H. 2005. “Distributed Constraint Based Railway Simulation”, Springer Verlag Berlin Heidelberg 2005.

Zimmermann A.; and G. Hommel. 2003. “A Train Control System Case Study in Model-Based Real Time System Design”, *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS’03)*.

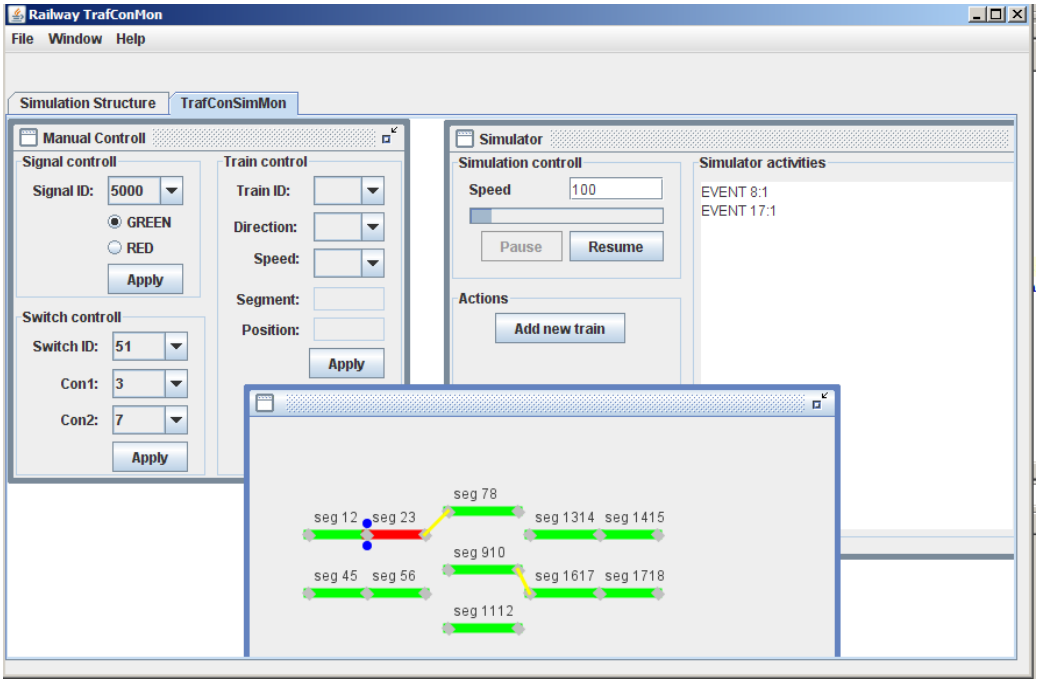

Figure 5. Simulator application main view.

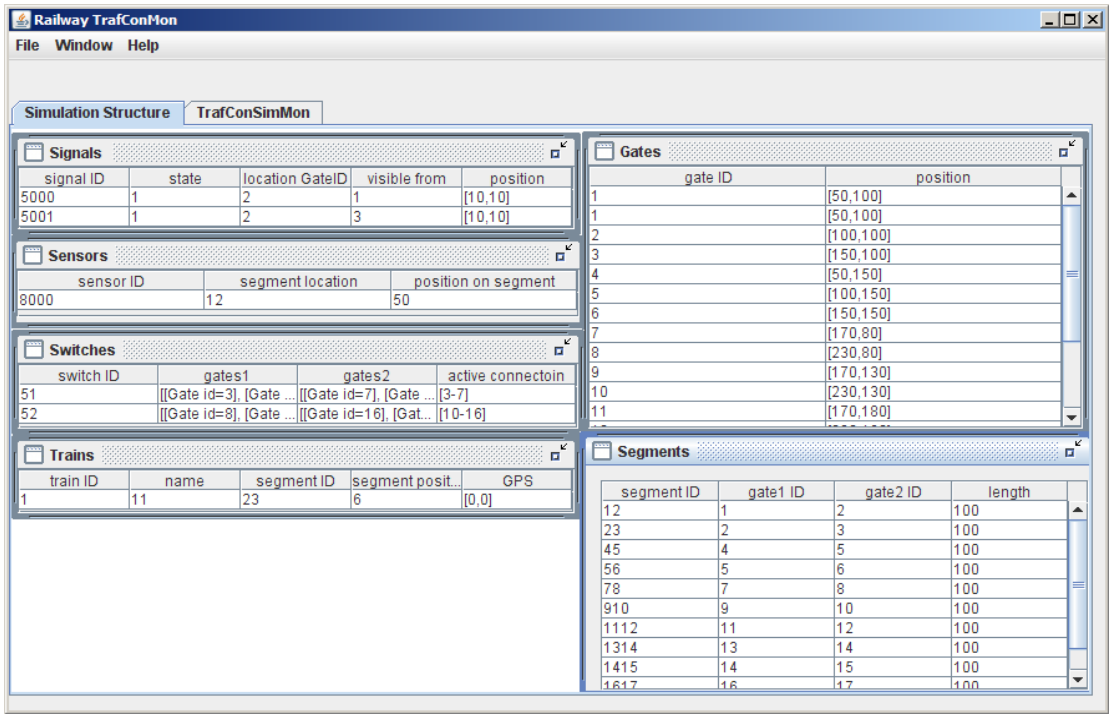

Figure 6. Simulator structure view.

AUTHOR BIOGRAPHIES

Camelia Avram works in the field of designing and implementing of real time applications and discrete events systems applied in communications protocols.

Mihai Hulea works in the filed of object oriented programming and real time applications.