*Proceedings of the 2005 Winter Simulation Conference*
*M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.*

# A DISTRIBUTED MULTI-FORMALISM SIMULATION TO SUPPORT RAIL INFRASTRUCTURE CONTROL DESIGN

Elisangela Mieko Kanacilo
Alexander Verbraeck

Systems Engineering Group
Faculty of Technology, Policy and Management
Delft University of Technology
Jaffalaan, 5, 2628BX, Delft, THE NETHERLANDS

## ABSTRACT

In this study we use simulation as a method of inquiry to support rail infrastructure control designers in making more effective decisions during the design process. Limitations encountered in commercial simulation tools when modeling rail system elements, are related to the choice of just one formalism (discrete or continuous) to model the element behavior. When supporting the design of rail system control, rail controllers and rail control designers might be in different locations. Therefore distribution of the simulation model is a required feature which is usually not possible in current simulation environments. In order to more accurately represent rail systems behavior and improve the effectiveness of control design, we propose a simulation library where different formalisms can be integrated in one single model and where simulation components are accessible by users in different locations.

## 1 INTRODUCTION

Designing rail infrastructure control is a challenging task. Simply stated, rail control should be designed in such a way that the system performance stays stable and at an acceptable level for as long as possible, also when disturbances occur. This also includes strategies that will give flexibility to the controller to adjust the control during operation.

When designing control, requirements from different stakeholders (rail company, local authorities, ministry of transport, etc.) have to be satisfied. These requirements might conflict to each other and control designers have to find a balance among all of them and not just consider each one independently.

For example, the decision of placing a station just before a crossing between tracks and a street, is more convenient and safer for passengers (as they do not need to cross the street to arrive at the station), if it is seen only from the rail company side, which has passenger's satisfaction as one of the requirements. Considering the situation where rail vehicles have priority over cars at crossings, the traffic light would be red for car drivers while the rail vehicle is still stopped at the station to collect passengers. This would delay the car traffic in the neighborhood and this scenario contradicts one of the requirements of the municipality, which is to avoid traffic jams. In this case, what might occur in reality is that car drivers would cross the red light if the tram is still at the station. This would increase the chances of accidents, what would end in an unsafe and non desirable situation for both sides: the company and municipality.

In addition, these requirements have to be satisfied taking into account the whole system conditions, like infrastructure capacity, vehicle lines priorities, control signs and etc. This is quite a complex system to design and control designers need technological support in order to understand the effects of their decisions on the performance of the whole system to make effective decisions.

In the complex environment of rail systems, simulation can play an important role in designing control strategies. It offers the possibility to test different strategies and to consider different scenarios only virtually, i.e., in a safer, quicker and cheaper way than in real situation experimentation (Versteegt and Verbraeck 2002; Bessey 2003). Simulation is therefore a useful instrument to test innovative control policies as it prevents many mistakes being made by assessing strategies first and apply only the ones that will bring more benefit to the infrastructure. Testing on beforehand is even more important because infrastructure control systems are very expensive, and because services might be disrupted during the implementation of changed control systems.

The promises of modularity and reusability has made Object Orientation very usual in simulation and modeling field (Verbraeck 2004). The complex structure of a system element[1] can be "hidden" from the outside world through encapsulation. Encapsulation (Sommerville, 2004) makes the object models less complex, it makes the system more modular and it increases the reusability of components in different situations.

With encapsulation it will be easy (Szypersky et al., 2002) to enable the interaction among rail system elements which present different types of behavior. The element behavior can be modeled in the formalism that best represents it and then it is encapsulated. Objects communicate with each other through interfaces, without looking into the details of the formalism in which a certain object was modeled. Rail infrastructures present many elements with a different kind of behavior. For example, vehicles movement continuously change its state while traffic lights change state in discrete steps.

Most commercial simulation tools, eg. eM-Plant or Automod, do not allow for the easy integration of multiple formalisms. Interoperability problems appear when mixing simulation components modeled in different formalisms in the same model. The modeler has to choose for instance for a continuous or a discrete representation, and one formalism within the discrete or continuous world.

In hybrid systems like rail infrastructures where both types of behavior are present, an environment where more formalisms can be used would provide a more accurate representation and a more elegant way of implementing. Multi-formalism does not mean that only discrete and continuous formalisms are provided but it also offers the possibility to the modeler to use the formalism that better represents a certain system element. For example, control designers could choose differential equations to model driving behavior and solve them using different solvers.

Although there are some commercial simulation tools that allow the integration of multiple formalism, to distribute the use of model components is difficult due to interoperability problems. When supporting the design of rail control, distributing the use of simulation component is important, because users (rail controllers and rail control designers) are normally placed in different locations and they need to share the same understanding of the system to take consistent decisions.

Based on that, we propose a simulation-based decision support environment for control design of rail infrastructures, where these issues are addressed. The belief is that the integration of multiple formalism will provide a more accurate and elegant way of modeling and the distribution will increase the effectiveness of the control design process.

[1] Rail system elements are all elements encountered in a rail system whose behavior is relevant to the scenario being studied, for example, sensors, tracks, trams/trains, traffic lights, speed signs, etc

This paper is structured as follows: Section 2 gives an overview of simulation tools used to support control design; Section 3 describes a case performed at a Dutch transportation company, Section 4 describes the proposed solution and Section 5 we draw conclusions.

## 2 EXISTING SIMULATION TOOLS TO SUPPORT CONTROL DESIGN CHALLENGES

Simulation is an appropriate approach to help people in understanding how complex systems work, where many interaction among system elements occur and these interactions affect the overall system behavior. Examples of simulation applications for the control design of logistic systems can be found at (Meer 2000; Ebben 2001; Versteegt and Verbraeck, 2002) and for transportation systems at (Manivannan 1998; Janssen 2005).

In the early years of simulation, simulation was already used to support the design of control system. Recent works, for instance (Harrel and Hicks 1998; Saanen 2004), show that simulation can be used during the whole system life, extending the life cycle of simulation models. Developing simulation components which are accurate enough to give a realistic representation of the system elements, gives the possibility to extend the usage of these components to real time operation to support rail controllers in taking more effective operational decisions.

Problems with accurate representation of models are related to the formalism used. As mentioned in (Neuendorffer 2004), control systems are usually designed using continuous formalism, following a set of ordinary differential equations and analytically solving these for continuous-time control signals. Unfortunately, not all systems are easy to describe using only differential equations. On the other hand, describing a control system using only discrete techniques is not a good approach, as many continuous aspects are neglected, making the system representation expensive and hard to understand as many states are required to abstract the dynamics with enough level of details.

A hybrid approach combining both discrete and continuous techniques captures the important aspects of the system and have been developed for the analysis and system control (Lygeros 2004).

Several papers describe the use of a hybrid system formalism. Liu et al. (1999) and Elker et al. (2001) applied a hybrid formalism to model embedded systems, Ludvig et al. (2002) applied this to high performance data acquisition systems and Liu (1998) applied a hybrid system formalism to modeling heterogeneous electronic systems containing analog and digital components.

Applying a multi-formalism approach to rail control design would improve the representation accuracy of simulation components, as distinct rail system elements present a different type of behavior. For instance, the tram driving moves continuously, while traffic lights change state in a discrete way.

Another requirement to better support the design of rail control is enabling the simulation to be distributed over different locations.

In rail infrastructures, control decisions are made based on a combination of requirements and constraints coming from departments responsible for the control of different parts of the system, such as physical infrastructure department, vehicles department, vehicle schedule department, etc.. Therefore, enabling the distribution of the use of simulation components among these departments would be appropriate for the control design process in this multi-actor environment.

The challenge in using a distributed multi-formalism approach is related to interoperability problems. Elements represented in different formalisms do not have a standard interface and therefore they cannot interact with each other.

In the next section we describe a case performed at a Dutch Transport Company and the problems encountered in a real rail setting.

## 3 HTM CASE

HTM Personenvervoer NV is a Dutch company which offers collective passenger transportation services through buses, trams and light rail vehicles. It is located in The Hague and it is responsible for part of the public transport in the city and surrounding regions. In 2003, 126 million passengers used the transportation services of HTM[2].

The case focuses on the rail network of the company and the objective is to study a real rail setting, to understand how the control is designed, to identify bottlenecks and to propose a solution. In this paper, we narrow down the focus of the case to study how tram schedules are made and assessed before commissioning them.

Currently, trams schedules are made with the help of analytical tools and people expertise. Timetables are mainly made based on employee's experiences and based on historical data gathered during vehicle journeys. These data are studied with the help of analytical tools, eg. TRITAPT (Verweij, 1991), and the outcomes of this analysis are used to make adjustments to timetables. After that, the adjusted timetables are applied within the rail network again. In this process, changes take long to be applied and it requires many (or endless) iterations.

To support HTM's control designers in accelerating this process and increase the effectiveness of the control decisions, we started developing a library of simulation components that can be used as a decision support system to help HTM in designing control, including the assessment of tram timetables.

In the course of the case, we identified the fact that in the rail environment many objects interact with each other. Looking at the dynamics of these objects, it was observed that they have their own particular behavior, but that the behavior could be classified in two main groups: discrete and continuous. From the modeling perspective, inside each group, many options to formally represent the behavior are available. For example, different types of vehicles could be modeled using different equation solvers. The communication among these objects should not be a problem as they can be part of the same model.

Another important factor that was identified is that due to the distributed nature of rail systems, control designers and rail controllers might be located in different places and therefore the simulation library should be accessible by many users placed in different locations.

Summarizing, the three main requirements for the domain-specific simulation library to be useful to rail companies are:

- Possibility to integrate multiple formalisms in one model
- Possibility to distribute the use of simulation components over different geographical locations.
- Possibility to link to other information systems, as some decisions need to taken based on the analysis of historical data.

## 4 DISTRIBUTED MULTI-FORMALISM SIMULATION

As already mentioned in a previous section, most commercial simulation tools present limitations when used to support the design of rail infrastructure control. Reasons can be cited as most available tools do not support the use of multiple formalism in the same model and the ones that do support this, do not allow the distribution of simulation components.

We chose to develop the library of components using DSOL – Distributed Simulation Object Language (Jacobs et al. 2002), a Java-based simulation suite that offers simulation services for the development of a fully distributed simulation model.

Features of a fully distributed simulation environment can be described as in (Jacobs et al. 2002): possibility to link with information systems (e.g., database) to automatically and more accurately specify simulation models; separation of simulation and visualization (this allows the simulation to be deployed on a more powerful computer (server) and to distribute the visualization among clients); and separation of the core simulation services from the model components that use it. With these features, the three requirements set for the proposed solution are fulfilled.

[2] Source http://www.htm.net, accessed July, 10th, 2005.

Using the simulation services provided by DSOL, a library of components representing rail system elements has been developed.

The control logic of each element is also implemented in a package that is embedded in the library of components. In this way, system elements can have different types of control logic, for example, traffic lights may be of different types, and they can be modeled using a different formalism, usually a discrete one.

Using the advantages of Object Oriented modeling, interoperability problems among objects can be overcome. The structure of each component is encapsulated and it communicates with other components through an interface where all public functionalities of this component are listed.

With the idea of encapsulation, elements can be modeled using any formalism and easily communicate with other elements. It is worth to mention here that this is possible, because the DSOL simulator (Jacobs et al., 2002) follows the idea of (Sol, 1982; Zeigler, 1976; Zeigler et al, 2000) where the DSOL simulation environment gives the freedom to modelers to use any formalism to conceptualize and specify a system.

DSOL simulator allows to simulate the DEVDESS formalism (Ziegler, 2002), a combined discrete-continuous formalism.

Figure 1 describes an interaction of elements modeled in different formalisms that commonly occurs in a rail system.

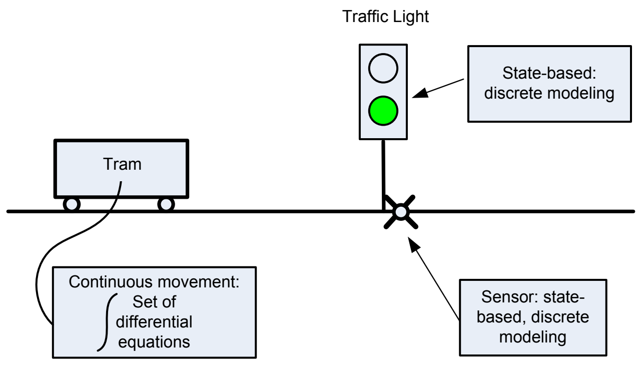
Traffic Light
State-based: discrete modeling
Tram
Continuous movement: Set of differential equations
Sensor: state-based, discrete modeling

Figure 1: interaction among elements of different nature

In Figure 1, it is shown that trams move continuously and an appropriate representation would be a set of differential equations. Traffic lights, as well as sensors, are state-based and a discrete representation would be preferable.

This example shows that objects modeled in different formalisms can and need to interact with each other, to produce the actual system dynamics. In this situation, the tram movement depends on the state of the traffic light, i.e., if a green light is given the driver can keep on driving. On the other hand, if a red light is shown, the tram should stop. More detailed explanation about how this interaction occurs is given in sub-section 4.1.

If necessary, different formalisms can also be combined to model the structure of only one object. For example, to model a tram, we created a class named `VehiclePhysical` (see Figure 2 for the class diagram). It contains all required operations to represent a rail vehicle. `VehiclePositioner` class, invoked by `VehiclePhysical`, models only the movement of the tram. In `VehiclePositioner` is where the differential equations and the correspondent solver is set. It extends `DifferentialEquation` class from DSOL, which is an implementation of DESS formalism (Differential Equations System Specification).

Methods like `getCurrentSpeedandProgression` and `getCurrentAcceleration` in `VehiclePhysical` class are examples of discrete operations of a rail vehicle. Therefore, as `VehiclePhysical` needs to deal with discrete and continuous implementations, a DEVDESS (Ziegler, 2002) simulator becomes necessary.

In next sub-section, the publish-subscribe mechanism used to enable the interaction between a continuous and a discrete element is explained in more details.

## 4.1 Publish-Subscribe Mechanism

The publish-subscribe mechanism is an asynchronous communication method implemented in DSOL, which is based on generic publish-subscribe methods present in programming languages like Java. With this mechanism (Jacobs, 2005) the disproportionate communication traffic between objects is avoided and it makes the components loosely coupled. This feature will be of a big value for the distribution of components over a network when necessary.

The mechanism works as follows. Some objects are event producers. Event-producer objects trigger events that are relevant for the action of other objects. Objects interested in a certain event can add themselves to the subscribe list of an event-producer object and when the event occurs, all objects in the list will be notified.

Human behavior modeling is out of the scope of this project, but components were developed in such a way that the influence of humans action can easily be considered. This can be viewed by the situation illustrated in Figure 1. Some meters before each control element, like a traffic light, a speed sign, etc, there is a sensor to indicate that from that point on, the control element is visible to the driver. The publish-subscribe mechanism is used to communicate a relevant state change within the system (e.g. a traffic light change) to the driver, who can react on this state change.

In Figure 3a, 80m before the traffic light `TL1`, there is a `visible_sensor` to represent the fact that in that location, `TL1` becomes visible to the tram driver. As there is not a driver class in our library, the tram `T1` subscribes to the change_state event list of `TL1` (Figure 3b). This list is called the *subscribe list* and when TL1 changes its state, all objects in the list will be notified. According to the new state, all interested objects will perform an action independently. For example, if TL1 changes its state to green, the tram will increase speed or keep it constant to pass through the traffic light, respecting always the maximum allowed speed. On the other hand, if the new state of TL1 is red, then the tram might start braking and stop before the traffic light and wait until TL1 shows a green light again.

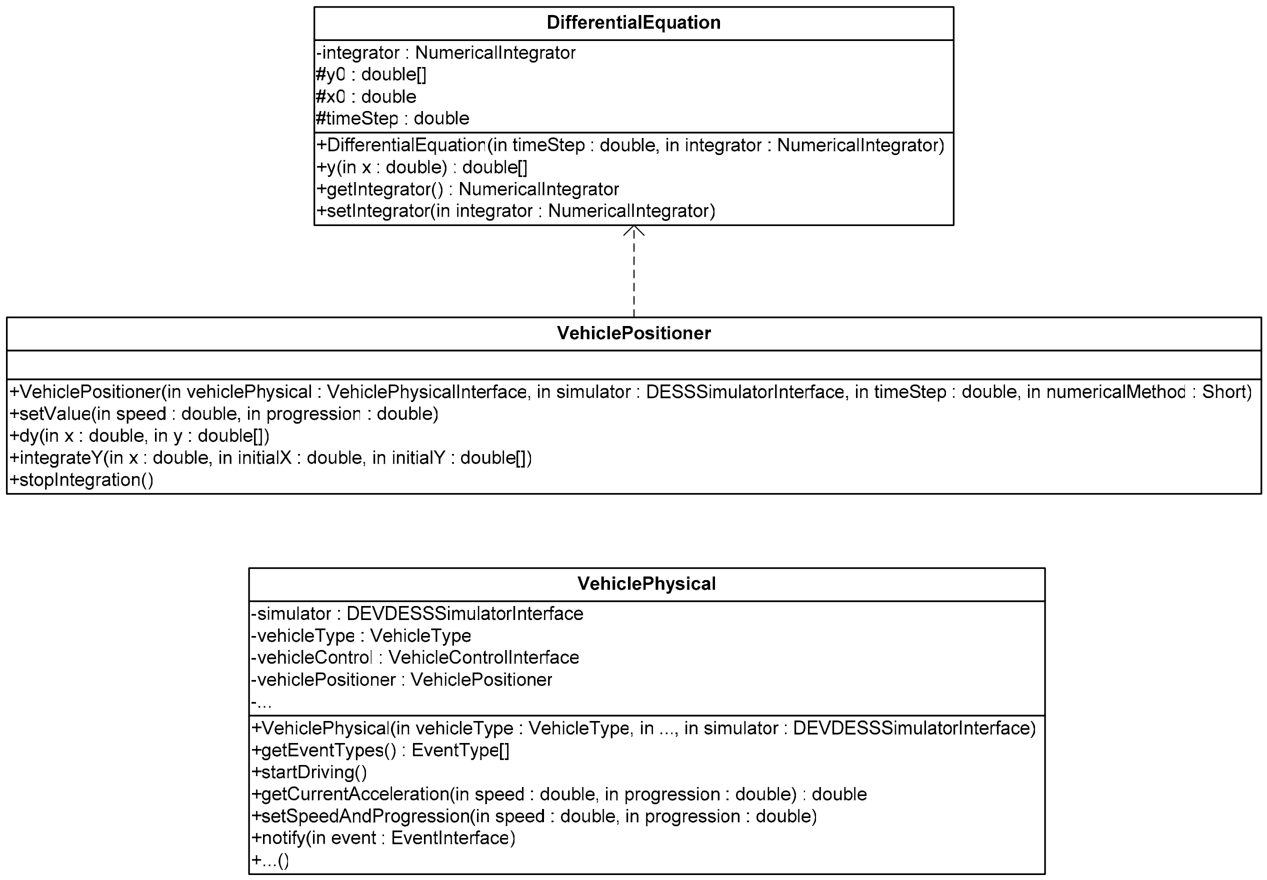
DifferentialEquation
-integrator : NumericalIntegrator
#y0 : double[]
#x0 : double
#timeStep : double
+DifferentialEquation(in timeStep : double, in integrator : NumericalIntegrator)
+y(in x : double) : double[]
+getIntegrator() : NumericalIntegrator
+setIntegrator(in integrator : NumericalIntegrator)
VehiclePositioner
+VehiclePositioner(in vehiclePhysical : VehiclePhysicalInterface, in simulator : DESSSimulatorInterface, in timeStep : double, in numericalMethod : Short)
+setValue(in speed : double, in progression : double)
+dy(in x : double, in y : double[])
+integrateY(in x : double, in initialX : double, in initialY : double[])
+stopIntegration()
VehiclePhysical
-simulator : DEVDESSSimulatorInterface
-vehicleType : VehicleType
-vehicleControl : VehicleControlInterface
-vehiclePositioner : VehiclePositioner
-...
+VehiclePhysical(in vehicleType : VehicleType, in ..., in simulator : DEVDESSSimulatorInterface)
+getEventTypes() : EventType[]
+startDriving()
+getCurrentAcceleration(in speed : double, in progression : double) : double
+setSpeedAndProgression(in speed : double, in progression : double)
+notify(in event : EventInterface)
+...()

Figure 2: class diagram of a continuous element

It is important to mention that if the tram does not have enough distance to brake, the simulation library can provide some warnings to the control designer that some measurements needs to be taken: control the tram speed by placing a speed sign with lower value for maximum allowed speed in that region or increase the distance before the traffic light to place the visible_sensor. This last choice is not always possible, as in reality the traffic light is visible from a certain point onward and this is hardly ever changed. There might be some high buildings blocking it, and the rail company cannot do anything in this sense. In case a speed limit needs to be changed, the simulation model supports the choice for the appropriate one.

After tram T1 passes the traffic light, it removes itself from the subscribe list of TL1 and as soon as it becomes necessary, it adds itself to the subscribe list of other objects.

To enable this interaction, VehiclePhysical class implements the EventListenerInterface from DSOL, and the TrafficLight class implements the EventProducerInterface from DSOL. Figure 4 shows the class diagram.

As an event producer, the TrafficLight class can add and remove listeners to its subscribe list. VehiclePhysical, as event listener, may overwrite the method notify with the code implementing the required action when the target object (traffic light, in this case) trigger the event of interest (change state, in this case).

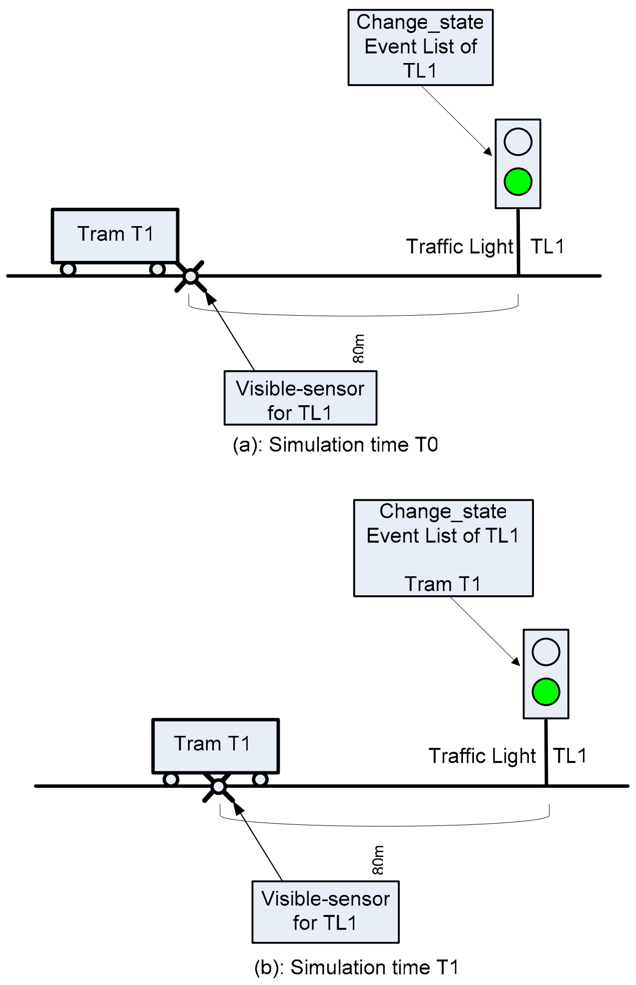

Figure 3: Subscription-notification mechanism

The publish-subscribe mechanism does not only add value in enabling the interaction among components modeled using different modeling formalisms, but it also enables the interaction among distributed objects. When simulation is run in a distributed setting, objects can keep track of other objects' dynamics, even they are run in remote computers. The asynchronous type of communication avoids the disproportionate traffic through the network.

### 4.2 Triggering Sensors in Multi-formalism Simulation

As described in the previous sub-section, using additional sensors helps in modeling relevant human behavior and produces good results. But triggering a sensor, which presents a discrete behavior, by a tram that has continuous dynamic shows some problems.

If we consider discrete steps in time to update the tram position (the time step of the differential equation is solved), the actual triggering sensor time will always be between a certain simulator time interval. No matter in which part of the interval the triggering sensor time is (whether it is at the beginning or at the end of the interval), the sensor will be effectively triggered only at the end of this interval as this is the time when the differential equation will be aware of the discrete space change. This will cause a delay in the action of triggering a sensor. Considering the sensor an event-producer object, this delay is propagated throughout the system, as many other actions are dependent on that.

We used a set of differential equations which are solved by a Runge-Kutta-4 method. For the application to a rail system control, the problem is solved. But we identified that for control systems that requires more accurate control, in terms of milliseconds, this method still present a problem. Even if it is a continuous formalism, the actual solving of each iteration on a computer occurs in very tiny discrete time intervals.

We conclude that for rail control where the time accuracy does not need to be so precise, this formalism is applicable and produces good results. In other cases, the time step to evaluate the differential equations should be decreased to give a more precise result or choose for another solver method.

## 5 CONCLUSIONS

In this paper we describe a distributed multi-formalism simulation approach to support the design of rail system control. Integration of multiple formalism in the same model provides a more accurate and more intuitive way of modeling because the representation is more similar to the real behavior. Allowing the simulation components to be used in a distributed setting adds value to the design of rail control in a multi actor environment.

Having a more realistic scenario reflects on the accuracy of the results and in the models acceptance. Changing the tram's behavior from discrete to continuous made the coding more clear and more concise. Code was reduced in hundreds of lines.

With a more realistic representation of rail system elements, there is an expectation to extend the usage of the simulation components to real time operation. As rail infrastructures are often influenced by disturbances (bad weather, accidents, machines breakdown, etc.), supporting operators to take decisions in real time would add value to the system control.

The belief is that testing alternatives of control strategies before applying them to the real infrastructure, can improve the effectiveness of the system as controllers can have the chance to choose for the strategies which have bigger probability to increase the rail system performance.

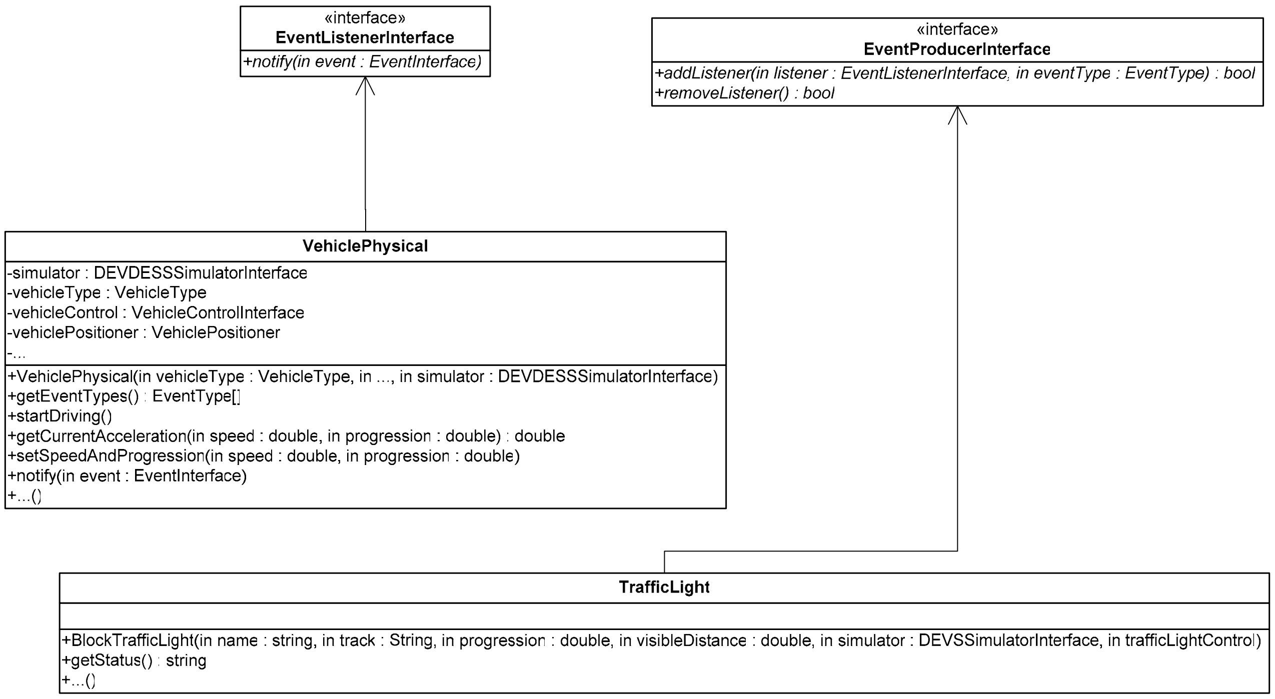
«interface»
EventListenerInterface
+notify(in event : EventInterface)
«interface»
EventProducerInterface
+addListener(in listener : EventListenerInterface, in eventType : EventType) : bool
+removeListener() : bool
VehiclePhysical
-simulator : DEVDESSSimulatorInterface
-vehicleType : VehicleType
-vehicleControl : VehicleControlInterface
-vehiclePositioner : VehiclePositioner
-...
+VehiclePhysical(in vehicleType : VehicleType, in ..., in simulator : DEVDESSSimulatorInterface)
+getEventTypes() : EventType[]
+startDriving()
+getCurrentAcceleration(in speed : double, in progression : double) : double
+setSpeedAndProgression(in speed : double, in progression : double)
+notify(in event : EventInterface)
+...()
TrafficLight
+BlockTrafficLight(in name : string, in track : String, in progression : double, in visibleDistance : double, in simulator : DEVSSimulatorInterface, in trafficLightControl)
+getStatus() : string
+...()

Figure 4: Event producer and Event listener objects

## ACKNOWLEDGEMENTS

We acknowledge the support of the HTM Personenvervoer NV for this research. Special thanks to Mr. Willard Kamerling (email: w.kamerling@htm.net), project leader on the side of HTM.

The research project has been funded by the BSIK-NGI program.

## REFERENCES

Bessey, T. 2003. On-Line Simulation: Towards New Statistical Approaches. In *Proceedings of SCS'03 Summer Computer Simulation Conference*, ed. A. Buzzone, M. Itmi, 35 (3): 453-458, Society for Computer Simulation, San Diego, CA, USA.

Ebben, M. 2001. *Logistic Control in Automated Transportation Networks*. Doctoral Dissertation, Twente University . Enschede, The Netherlands.

Elker, J., C. Fong, J. W. Janneck, and J. Liu. 2001. Design and Simulation of Heterogeneous Control Systems Using Ptolemy II. In *Proceedings of the Conference on New Technologies for Computer Control,* n 1220 Elsevier.

Harrell, C. R. and D. A. Hicks. 1998. Simulation Software Component Architecture for Simulation Based Enterprise Applications. In *Proceedings of Winter Simulation Conference*, ed. D. J. Medeiros, E. F. Watson, J. S. Carson and M. S. Manivannan, 1717-1721, IEEE Inc, Piscataway, NJ, USA. Available via http://www.informs-cs.org/wsc98papers/236.PDF [accessed April, 15, 2005].

Jacobs, P. H. M. 2005. *The DSOL Simulation Suite: Enabling Multi-Formalism Simulation in a Distributed Context.* Doctoral Dissertation, Delft University of Technology, Delft, The Netherlands (forthcoming).

Jacobs, P. H. M., N. A. Lang and A. Verbraeck. 2002. D-SOL: A Distributed Java Based Discrete Event Simulation Architecture. In *Proceedings of Winter Simulation Conference,* ed. E. Yucesan, C.-H. Chen, J. L. Snowdon and J. M. Charnes, 793-800, IEEE Inc., Psicataway, NJ, USA. Available via <http://www.informs-sim.org/wsc02papers/102.pdf> [accessed April, 15, 2005].

Janssen, M. F. W. H. A., Verbraeck, A. 2005. Evaluating the Information Architecture of an Electronic Intermediary. *Journal of Organizational Computing and Electronic Commerce* 15 (1), 35-60.

Liu, J. 1998. Continuous Time and Mixed-Signal Simulation in Ptolemy II. Memo M98/74, UCB/ERL, EECS UC Berkeley, CA, USA.

Liu, J., X. Liu, T. J. Koo, B. Sinopoli, S. Sastry, and E. A. Lee, 1999. A Hierarchical Hybrid System Model and its Simulation. In *Proceedings of the Conference on Decision and Control*, 3508-3513, IEEE Inc., Psicataway, NJ, USA.

Ludvig, J., J. McCarthy, S. Neuendorffer, and S. R. Sachs. 2002. Reprogrammable Platforms for High-Speed Data Acquisition. *Journal of Design Automation fro Embedded Systems* 7 (4): 341-364.

Lygeros, J. 2004. Lecture Notes on Hybrid Systems. Patras, Greece, Department of Electrical and Computer Engineering, University of Patras.

Manivannan, M. S. 1998. Simulation of Logistics and Transportation Systems. In J. Banks (ed.), *Handbook of Simulation: principles, methodology, advances, applications and practices.* Wiley & Sons, New York, USA.

Meer, R. v. d. 2000. *Operational Control of Internal Transport.* Doctoral Dissertation, Delft University of Technology, Delft, The Netherlands.

Neuendorffer, S. 2004. Modeling Real-World Control Systems: Beyond Hybrid Systems. In *Proceedings of Winter Simulation Conference*, ed. R. G. Ingalls, M. D. Rossetti, J. S. Smith and B. A. Peters, 240-248, Madison, OmniPress. Available via < http://www.informs-sim.org/wsc04papers/028.pdf> [accessed April, 15, 2005]

Saanen, Y. 2004. *An Approach for Designing Robotized Marine Container Terminals.* Doctoral Dissertation, Delft University of Technology, Delft, The Netherlands.

Sommerville, I. 2004, *Software Engineering.* 7th edition, Pearson/Addison Wesley, Harlow, United Kingdom.

Szypersky, C., G. Dominik, S. Murer. 2002. Component Software: Beyond Object Oriented Programming. 2nd edition, Addison-Wesley, London, United Kingdom.

Verbraeck, A. 2004. Component-Based Distributed Simulations. The Way Forward? In *Proceeding of 18th Workshop on Parallel and Distributed Simulation (PADS'04)*, eds. S. Kawada, 141-148, IEEE Inc., Los Alamitos, California, USA.

Versteegt, C. and A. Verbraeck 2002. The Extended Use of Simulation in Evaluating Real-Time Control Systems of AGVs and Automated Material Handling Systems. In *Proceedings of 2002 Winter Simulation Conference,* ed. E. Yucesan, C.-H. Chen, J. L. Snowdon and J. M. Charnes, 1659-1666, IEEE Inc., Psicataway, NJ, USA. Available via http://www.informs-sim.org/wsc02papers/227.pdf [accessed April, 15, 2005]

Verweij, C.A. 1991. *Tritapt-Programmapakket. Installatie, Gebruik en Werking.* VK 4505.309, Delft, The Netherlands (in Dutch).

## AUTHORS BIOGRAPHIES

**ELISANGELA M. KANACILO** is a Ph.D. candidate at Delft University of Technology. She got her Masters degree in Software Engineering by Universidade Federal de Uberlandia, in Brazil. Her PhD research is focused on developing a simulation environment to support the design of rail infrastructure control. Her email address is <e.m.kanacilo@tbm.tudelft.nl> and her web page is <www.tbm.tudelft.nl/webstaf/elisangelak>.

**ALEXANDER VERBRAECK** is chair of the Systems Engineering Group of the Faculty of Technology, Policy and Management of Delft University of Technology, and a part-time full professor in supply chain management at the R.H. Smith School of Business of the University of Maryland. He is a specialist in discrete event simulation for real-time control of complex transportation systems and for modeling business systems. His current research focus is on development of generic libraries of object oriented simulation building blocks in C++ and Java. His e-mail address is <a.verbraeck@tbm.tudelft.nl>, and his web page is <www.tbm.tudelft.nl/webstaf/alexandv>.