Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

SIMULATION SERVICES TO SUPPORT THE CONTROL DESIGN OF RAIL INFRASTRUCTURES

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

The design of rail infrastructure is a difficult task. Many parties are involved, and the tasks range from stakeholder issues to very detailed technical questions, such as control design. Simulation studies are often applied during infrastructure control system design, but the application of simulation is quite hard. One of the problems is the lack of flexibility in linking to information systems and databases. Another problem is that there are many potential users of the models, while most simulation systems can only be used by one user at a time. In addition, the tightly coupled structure of models makes model reuse and model maintenance hard. To overcome these problems, a service oriented simulation architecture is proposed for rail infrastructure modeling. The object-oriented simulation libraries that have been created within this architecture have been tested in a real project to estimate rail infrastructure capacity, and proved to work well.

1 INTRODUCTION

When designing control systems for rail infrastructures (Harris and Godward 1992, Sussman 2000), designers aim at control strategies that will achieve a cost-effective and appropriate system to transfer passengers safely and reliably. Infrastructure capacity influences the quality of the transport service, not only in terms of reliability, with an infrastructure that provides enough trams / trains to satisfy the demand, but also in terms of safety, where the transport of the required number of passengers can be done in a safe way.

Estimating infrastructure capacity can be very complex, as many factors influence the throughput of vehicles. Depending on the control strategies in use, the number of trams / trains per hour that the infrastructure can support in a rail network can vary a lot. For example, giving priority to a tram line at crossings, or changing the speed limit along the rail network, will influence the travel time and consequently, it will influence the number of vehicles sharing the tracks during a certain time interval. Therefore, the rail infrastructure capacity will depend on the control strategies applied.

Given the complexity of the task, simulation is an adequate method to support this what-if analysis during the control design of rail infrastructures, and to help rail control designers in finding an appropriate set of control strategies that will allow the rail company to increase profit by increasing the efficiency in using the infrastructure resources in relation to infrastructure capacity.

Many existing simulation models which are used to support the control design of rail infrastructures, are not adequately supporting this task. The problems encountered can be divided into two types according to the user level: the end user (rail control designer) and the model builder. For the end user, the problems are related to the fact that simulation models are normally stand alone, single user applications. Usually, several designers, potentially from different disciplines, are involved in rail control design. Of course, installing the simulation tool on more computers is possible, but this will lead to extra costs in buying software licenses in addition to the work of installing the simulation package. Furthermore, it is hard to guarantee that the designers will all work with the same version of the model and of the control system.

Another problem is the difficulty to link to other information systems, such as database, geographical information systems, and others. During control design, historical data is often useful, either to compare system performance with historical performance, or to retrieve data about the infrastructure, such as the layout of the physical network, control strategies in use, etc.

For the model builder's perspective, the problem is that models are normally implemented with too many details, not allowing for the reusability of the model in other projects.

Another issue is that the simulation model is usually structured in a tightly coupled way, which makes the model hard to maintain or extend at a later stage of the project.

To solve these problems, we propose a Service Oriented Simulation Architecture for rail infrastructures. Using the concept of service orientation could help in creating a much more clear structure of the model, and to give more flexibility to both the model builder in developing a simulation model and to the end user in making use of the architecture to support his / her work.

In Section 2 we provide some information about the control design of rail infrastructures and the supporting tools used for this task. Section 3 describes the proposed Service Oriented Simulation Architecture. Section 4 contains the explanation of how we applied the proposed architecture to a real case, and in Section 5 we draw conclusions based on the use of the architecture and the implementation.

2 RAIL INFRASTRUCTURE CONTROL AND SUPPORTING TOOLS

In Harris and Godward (1992), the authors describe how important is to plan a rail infrastructure. Control strategies to operate the infrastructure should be developed to achieve a cost-effective and appropriate system to transfer passengers safely and reliably.

In this study, we focus the aspect of capacity measurement, which influence safety and reliability in rail infrastructure design. According to the set of control strategies applied to the infrastructure, capacity can vary a lot. Examples are vehicles with different physical characteristics (weight, acceleration and deceleration rate, length), different speed limits along the rail network, and priorities for tram lines at certain locations. These are all examples of control strategies that should be carefully analyzed before commissioning them. For each choice, the infrastructure capacity will probably vary and alternatives have to be analyzed in terms of safety and reliability. There are many possible combinations of control measures that lead to a good system configuration, and designers have to find out the good ones by exploring different scenarios. Therefore, simulation tools seem very appropriate to support this what-if analysis.

There are many simulation tools that are used to support the control design of rail systems. SIMONE is a Dutch simulation package used to assess timetables of large scale rail networks. In Middelkoop and Bouwman (2000, 2001), the authors apply SIMONE to evaluate and compare many traffic scenarios in The Netherlands. SIMON (Wahlborg 1996, Bergmark 1996) is a Swedish software package used to simulate the whole train network, and UX-SIMU (Kaas 2000) is used in Denmark to simulate rail systems traffic. RailSys is a supporting tool to create timetables for rail networks where the block safety mechanism is in use. The works of Rudolph and Demitz (2003) and Demitz et al. (2004) describe the application of RailSys to a rail network and how it can improve the timetable of a system. There are more supporting tools used to simulate rail infrastructures traffic, and the ones mentioned above are only examples.

Although existing simulation tools are very useful to test one aspect of the control design in isolation, like for example timetable assessment, delay propagation, or traffic analysis, they do not provide support for, or it becomes hard to analyze all these aspects at the same time in the model. For example, it might be that control strategies used to improve timetables and the ones used to decrease delay propagation will not produce a good system performance, when they are combined in the same scenario. Existing tools provide insufficient flexibility to the modeler to build a model where the rail control designer can have a clear understanding of the system operation.

In addition, commercial simulation models normally present interoperability problems (Taylor et al. 2003), as the wrappers to enable the link to other information systems, can lead to unstable code and models are hard to maintain. The problem with maintenance is that some wrappers contains proprietary code in addition to the model (Verbraeck 2004).

To support the control design of rail infrastructures, it is important to enable a link to databases, as historical data are often used to make performance comparisons, or to retrieve information about the infrastructure, such as network layout, vehicle information, control strategies in use, etc.

To address these problems, we propose a service oriented simulation architecture to support the control design of rail infrastructure more efficiently, as a service oriented simulation architecture (Lang, Jacobs, and Verbraeck 2003) *does* take into account interoperability, linkage to databases, and multiple users in different locations.

3 SERVICE ORIENTED SIMULATION ARCHITECTURE

The proposed architecture consists of a library of simulation components organized in a service oriented way. Papazoglou (2003) defines service orientation as the computing methodology that uses services as fundamental elements of an application. When building a service model, the separation between the interface and the implementation is fundamental.

Services are similar to class objects and components (Sprott and Wilkes 2004) and they can be seen as building blocks that combine information and behavior, hide their internal working from the outside world, and present simple interfaces. A difference is that while objects and components are organized into classes or hierarchies of inherited behavior, services are published and consumed singly or as hierarchies (Sprott and Wilkes 2004).

Based on these concepts, we built our service oriented architecture. The architecture consists of a li-

brary of simulation components to represent rail elements and their behavior, generate statistics with the purpose to support output analysis, to check and validate input data, to animate the simulation, and to visualize statistics. The simulation components are implemented in the Java programming language on top of DSOL, the Distributed Simulation Object Library (Lang, Jacobs, and Verbraeck 2003), an open source simulation suite (available at <http://sourceforge.net/projects/dsol>).

Figure 1 gives an overview of the proposed architecture, showing the simulation components clustered by the type of services they offer. The explanation of each service type is given below:

- **Physical Layer**: classes representing the physical elements of a rail infrastructure, such as tracks, vehicles, sensors, stations, traffic lights, etc. This layer provides services to create a physical rail infrastructure with rail elements displayed at specified locations. Classes of these layers create only the physical structure of a rail element and the behavior intrinsic to the element, if it is a moving object. For example, classes implementing vehicles are part of this layer and when a vehicle is instantiated, its engine, including the acceleration and deceleration rate, are assigned to the vehicle. The way these vehicles will be operated, like how fast or how slow they can be driven, are seen as driver behavior and these are part of the control layer.
- **Control Layer**: it contains classes representing the control logic of elements of the physical layer. Services from this layer define different types/behavior for objects. For example, a three light traffic light might have different logic to change its state. It can be time fixed or according to the use of the infrastructure it is guarding. In this case, just one class is defined in the physical layer, but two classes are necessary in the control layer to implement both logics. With this structure, it is easier to extend the library by adding more object types, when necessary.
- **Statistics Layer**: classes to generate and to support the generation of important statistics for a rail system are implemented in this layer. Statistics include graphs and additional classes to calculate key performance indicators (KPI) of a rail infrastructure. KPIs for a rail infrastructure are for instance, average travel time, speed average and average waiting time at crossings or in front of stations.
- **Input Processing**: this contains classes to support the customization of the model. Users provide the infrastructure configuration, control measures, and schedules in an XML file. The data in the XML file are checked and validated by an XML parser. In addition, users can input data manually or retrieved data automatically from a database, from excel sheets, or from other data sources.
- **Output Processing**: classes to enable the animation of the scenario during simulation and additional classes to support the analysis of statistics by visualization of graphs, or figures of the specified performance indicators.

When input data, defined by the user and/or retrieved from a database, is checked and validated in the input processing, data is passed to the physical, control and statistics layers. The services of the corresponding layers are invoked and the physical infrastructure, the control part and the additional components to generate statistics are then created. After this, the model is loaded and the simulation can start. After pushing the "play" button, users can optionally use the services from the output processing to visualize the animation of the scenario, graphs and other statistics.

With the use of Java, an Object-Oriented language (Sommerville 2004), it is easy to separate object information from its services. Information and internal behavior of an element are implemented in a class and it can be kept hidden from the outside world. The services the element provides are made public and are listed in an interface. This creates a loosely coupled structure for the architecture and makes it easy to extend it through the use of the extension concept from Object Orientation.

Interactions among other objects are done through the invocation of public services. In our architecture, we use the concept of the publish-subscribe mechanism, to invoke services from other objects automatically when a certain event or state change occurs. To explain how we apply this mechanism, we give an example of a tram interacting with a traffic light.

In a rail system, the state of a traffic light indicates whether the tram should keep on moving or should brake. The state of a traffic light is given by the color of the light, red for stop, green for clear. *ChangeState* is the event that influences the tram dynamics, therefore, we say a traffic light is an *EventProducer* and the tram is an *EventListener* for this publish / subscribe event (which is a different event from a simulation event).

We abstract the distance from which a driver can see the state of a traffic light, by using a *visibleSensor*. See Figure 2(a). When the tram triggers this sensor, it means that the vehicle can take action.

When the tram passes over the *visibleSensor*, depending on the state of the traffic light at that moment, it will either accelerate or decelerate, but it will also subscribe to the event list (*Change-State List*) of the traffic light. See Figure 2(b). By subscribing to the *ChangeState* event of the traffic light, the tram is "aware" of the traffic light state and anytime it is

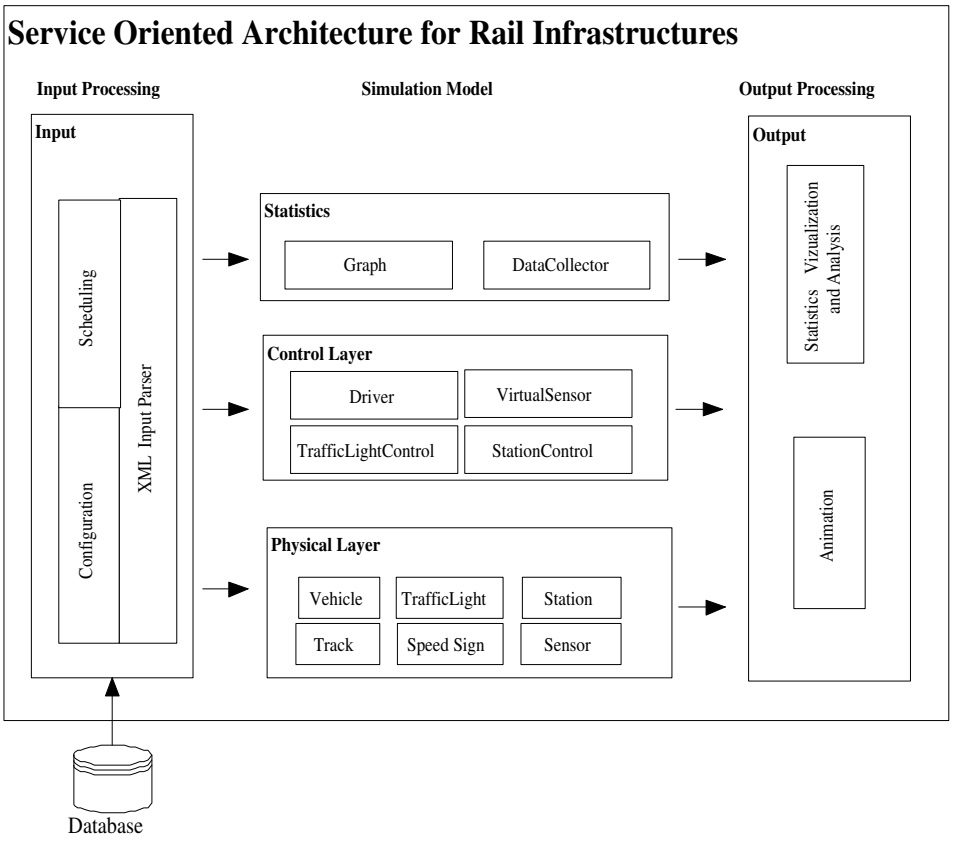

Figure 1: Overview of the Service Oriented Architecture

changed, all objects in the subscription list will be notified and each one of them can take the appropriate action. When the vehicle passes the traffic light, it removes itself from the subscription list, as the state of the traffic light does not influence the dynamics of the tram anymore.

The publish subscribe mechanism helps the interaction among objects in an asynchronous way. This avoids overload of message exchange among objects. Although objects are not directly connected, the messages will still reach the target objects.

In the next section, we apply simulation services to a real problem where the control system of a rail infrastructure has to be redesigned.

4 APPLYING SIMULATION SERVICES TO SUPPORT THE CONTROL DESIGN OF RAIL INFRASTRUCTURES

HTM Personenvervoer NV is a Dutch transport company offering collective transport services through buses and trams. The company is currently extending its network in order to introduce light rail transport services. This study focuses on one specific part of this network extension, where the new tracks will join existing tracks. With this change, in some

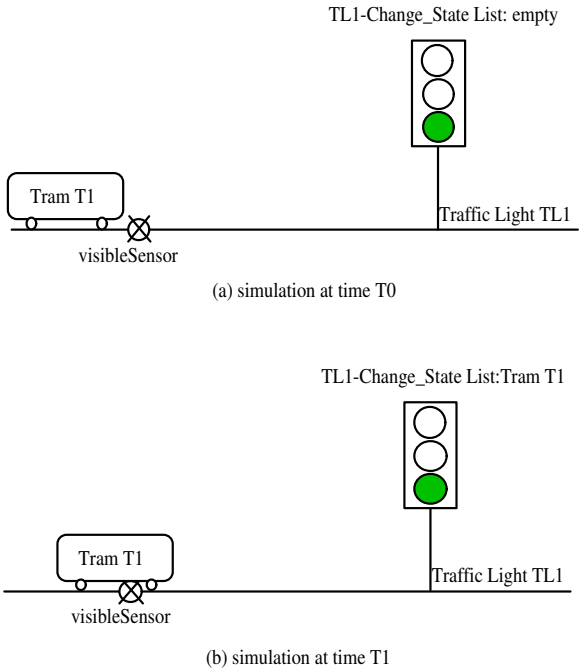

Figure 2: Publish-Subscribe Mechanism

parts of the rail infrastructure, the tracks will be shared by both heavy and light vehicles. See Figure 3.

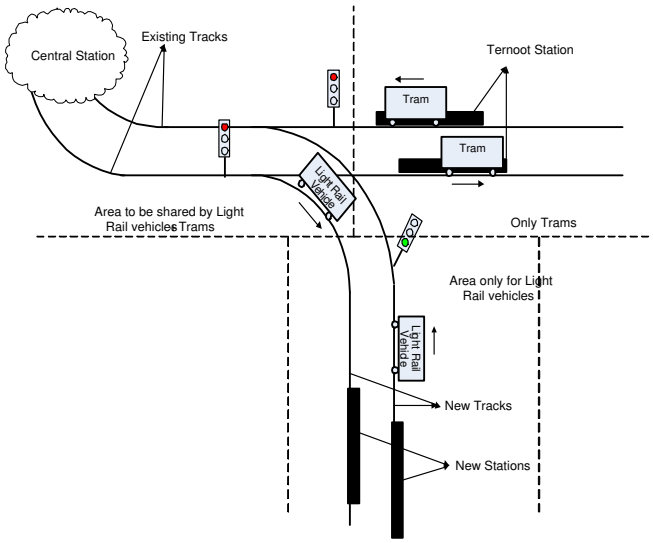

Figure 3: Extension of the Rail Network

The lines operated by light rail vehicles will serve more distant regions, therefore these lines are expected to have a higher passenger demand than existing lines.

The junction showed in Figure 3 is the connection point for light vehicles arriving at the Central Station. The company raised the possibility to give priority to light vehicles at the crossing in order to give these vehicles a quicker access to the city center.

This decision is not simple. Giving priority to light vehicles might indeed make the vehicles reach the Central Station quicker, but on the other hand, it might cause a long queue of vehicles with secondary priority waiting at the crossing and this is not desirable.

In addition, the whole control system will probably change. It might be necessary to shift the location of control signalling objects, change their type and/or add more signalling objects, in order to adapt the control system to the new traffic volume. Depending on the control strategies applied, the infrastructure capacity will vary. Therefore, measuring infrastructure capacity also means designing the control system.

We applied our service oriented simulation architecture to support the design of the control system for this area and to measure the infrastructure capacity as well. The experiments performed and result analysis are described in Section 4.1.

The link to databases is part of this research project but it has not been implemented yet. In this case, the input data was defined in an XML file. In this data file, we set the configuration of the physical infrastructure by specifying the types of control objects (such as traffic light types, sensor types, speed signs and others) and where to place them, we set the types of vehicle in use, etc. Input data also includes information about the control strategies chosen, like for example timetable and priority settings per tram line.

The XML parser takes the XML data as input and checks whether all information is given and if they are in the correct format. When data is valid, services from the physical, control and statistics layers are invoked and all necessary objects are instantiated. At this moment, the model is loaded. When the simulation starts, services from the output processing unit can be invoked to provide the animation of the scenario and a visualization of the statistics gathered.

To give more details on how this instantiation process occurs in our service oriented simulation architecture, we use the example of a traffic light. A real traffic light is represented by a class named *TrafficLight*. *TrafficLight* implements *TrafficLightInterface*. In the *TrafficLightInterface* we define all services this rail element provides. See Figure 4.

In the control layer, the control logic of this traffic light is created by an instance of *TrafficLightControl* which implements a *TrafficLightControlInterface*. These objects publish their services to the outside world through the interfaces. Any service listed in an interface is reached by any other object.

For the services of the statistics layer, suppose the user wants to calculate the average of time vehicles of a certain line have to stop because of a red light. Traffic lights are the location where sample data need to be collected. Therefore, the *DataCollector* class is interested in the state change of the traffic light. For this purpose, in the constructor of a *DataCollector* it adds itself as a listener to the *ChangeState* event of the traffic light (see in Figure 4 in Statistics Layer). Some lines below, one can see that if the *ChangeState* event of the traffic light is fired, the code in the method notify will be executed.

With the publish subscribe mechanism it is not necessary to keep on checking if the state of the traffic light has changed. By subscribing the DataCollector to the target event, the DataCollector object will be notified and can take the appropriate action. The publish-subscribe mechanism avoids an excess of communication traffic among objects and it works well in the loosely coupled structure. Once subscribed to the target events, objects will be notified of the event firing wherever they are. It is very appropriate for distributed or multi-user applications of the same model.

To support the control design, users can vary the values of the input parameters, like for example, varying the timetables (increasing the number of vehicles per hour) and monitor how the performance indicators vary accordingly. The animation and the statistics analysis calculate important performance indicators to support the decision of an appropriate control system. Switching the priority on and

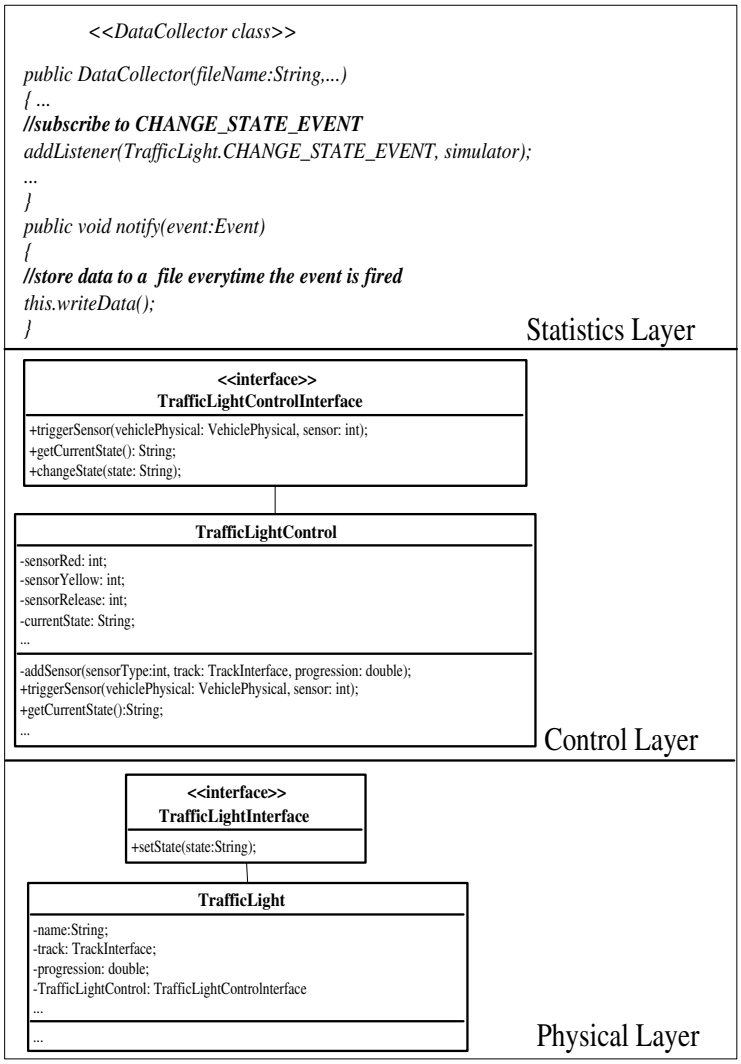

Figure 4: Layers of Services Related to a Traffic Light

off through the XML file, users can also see the variance in the performance indicators and also monitor the system behavior through the animation services.

In the following section, we describe the experiments we performed for this case and analyze the results.

4.1 Experiments and Result Analysis

The experiments were set with the purpose of answering two questions: (i) How many vehicles per hour does the infrastructure support? and (ii) What would be the impact in the system performance if light rail vehicles have priority to access the crossing?

As we were modeling a non-existing system there are no real data to compare with the simulated ones. For this reason, we set two basic scenarios to collect data to be used as reference. In these basic scenarios, except for the dwell time at stations, no other obstacle would delay the trip.

Performance was assessed by the analysis of the following indicators: average of travel time, speed average, irregularity and queue time. Irregularity is the difference of the actual arrival time at stations and the expected arrival time. Queue time is the time vehicles have to wait to enter in the system in order to respect the safety distance from its predecessor. The busier the infrastructure is, the longer the queue time will be. All performance indicators were measured per line and per direction.

Apart from the basic experiments, 8 more experiments were set. The experiments have: (1) 44 vehicles, (2) 88 vehicles, (3) 132 vehicles and (4) 176 vehicles. Based on these four, another four experiments were set with the same number of vehicles but with the priority rule in use. Because of confidentiality agreement, we cannot publish the numbers, but we give an overview of the result analysis.

In scenario 1, the values for the performance indicators did not exceed the reference data. This means that the system was running below its capacity, as no significant

delay was caused by the infrastructure usage. Scenario 2 showed an increase in some of the indicators, but was still considered acceptable. Most of the values did not exceed the reference data, therefore, the conclusion was that the infrastructure support 88 vehicles per hour.

The results of scenario 3 showed a bigger increase in all the performance indicators, as it was expected. But watching the animation of the experiment, one could notice that there was an overlapping in the timetable of tram lines, as vehicles coming from all directions approach the crossing almost at the same and have to compete for the crossing causing a delay. But right after vehicles leave the crossing, there was a time interval where no vehicle appear. Therefore, solving this conflict in the timetables would increase the performance. For scenario 4, measurements indicated a drop of performance in the system, with an increase of 20% of the queue time, which was enough reason to discard this scenario.

Regarding the priority aspect, the experiments did not show a big difference in the measurements if compared with the corresponding experiment where the priority rule was not applied. This is explained by the following: for safety reasons, vehicles have always to keep a safety distance from each other. Because of the fact that vehicles cannot overtake, even when the infrastructure is running over its capacity, it is only possible that at the maximum 3 vehicles (all three possible directions) per time compete for the crossing. Light rail vehicles only drive in two of three possible directions, where one direction is also shared with heavy vehicles. Therefore, the situations where the priority rule is the cause of the delay of other tram lines is very rare to occur and it does not influence much the waiting time.

Summarizing, the infrastructure supports 88 to 132 vehicles per hour, but vehicles departures should be better distributed over time to avoid the conflict at the crossing. Further analysis is necessary to find out what would be a good way to solve the conflict and produce an acceptable system performance. The priority aspect, differently from what was expected, did not improved the travel time of light rail vehicles and it increased the travel time of lines with secondary priority. Other types of priority could have a bigger impact in the system performance and make light rail vehicles arrive quicker at the central station. For example, granting the access to the crossing at farther distance would avoid vehicles to stop at the crossing, but this has to be tested and a more careful analysis is necessary.

Simulation showed to be a very good method for the what-if analysis when designing the control system for rail infrastructures. The performance can be assessed by the statistics and through animation. This assessment provides a clear understanding of the impact of the control strategies in the system behavior helping the control designer in making decisions about which strategies are more cost-effective.

5 CONCLUSIONS

We used a service oriented simulation architecture to support the control design of rail infrastructures. The service oriented architecture presented advantages in comparison with the structure of traditional simulation packages. The structure is more flexible and easier to maintain and extend. With the concept of publishing and consuming services, it is easier to link the simulation model to other information systems. Through the service oriented architecture, we can easily link to input files, databases, and spreadsheets.

For the users, the proposed architecture is more flexible in the sense that more aspects of a infrastructure control can be tested at the same time. In addition, the fact that it is Java based, makes it easy to distribute the model over a network, without any further costs for licences and software installation.

REFERENCES

Bergmark, R. 1996. Railroad capacity and traffic analysis using SIMON. In *Computers in Railways V*, ed. J. Allan, C. A. Brebbia, R. J. Hill, and G. Sciutto, 183-191. Southampton, United Kingdom : WIT Press.

Demitz, J., C. Hübschen, and C. Albrecht. 2004. Timetable stability – using simulation to ensure quality in a regular interval timetable. In *Computers in Railways IX*, ed. J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto, and S. Sone, 549-562. Southampton, United Kingdom : WIT Press.

Harris, N. G., and E. W. Godward. 1992. *Planning passenger railways*. Derbyshire, England : Transport Publishing Company Ltd.

Kaas, A. H. 2000. Punctuality model for railways. In *Computers in Railways VII*, ed. J. Allan, R. J. Hill, C. A. Brebbia, G. Sciutto, and S. Sone, 809-816. Southampton, United Kingdom : WIT Press.

Lang, N. A., P. H. M. Jacobs, and A. Verbraeck. 2003. Distributed open simulation model development with DSOL services. In *Proceedings of the 15th European Simulation Symposium 2003 – Simulation in Industry*, 210-218. Germany : SCS European Publishing House.

Middelkoop, D., and M. Bouwman. 2000. Train network simulator for support of network wide planning of infrastructure and timetables. In *Computers in Railways VII*, ed. J. Allan, R. J. Hill, C. A. Brebbia, G. Sciutto, S. Sone, 267-276. Southampton, United Kingdom : WIT Press.

Middelkoop, D., and M. Bouwman. 2001. SIMONE: large scale train network simulations. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B. Peters and J. Smith, 1042-1047. Available online via <http://www.informs-sim.org/

wsc01papers/140.PDF> [accessed March, 29th, 2006].

Papazoglou, M. P. 2003. Service-oriented computing: concepts, characteristics and directions. In *Proceedings of the 4th IEEE International Conference on Web Information Systems Engineering*, 3-12. IEEE Computer Society.

Rudolph, R., and J. Demitz. 2003. Simulation of large railway networks. Robustness test for the timetable 2003 in North Rhine Westphalia, Germany, In *Proceedings of the World Congress on Railway Research 2003*, 644-652. Edinburgh, United Kingdom.

Sommerville, I. 2004. *Software engineering*, 7th edition. Harlow, UK : Pearson/Addison Wesley.

Sprott, D., and L. Wilkes. 2004. *Understanding service oriented architecture*. Available online via <http://msdn.microsoft.com/architecture/soa/default.aspx?pull=/library/en-us/dnmaj/html/aj1soa.asp> [accessed April, 1st, 2006].

Sussman, J. 2000. *Introduction to transportation systems*. Norwood, MA : Artec House, Inc.

Taylor, S. J. E., B. P. Gan, S. Straβburger, and A. Verbraeck. 2003. HLA−CSPIF panel on commercial off-the-shelf distributed simulation. In *Proceedings of the 2003 Winter Simulation Conference*, ed. S. Chick, J. Sánchez, D. Ferrin, and J. Morrice, 881-887. Available online via <http://www.informs-sim.org/wsc03papers/107.pdf> [accessed March, 29th, 2006].

Verbraeck, A. 2004. Component-based distributed simulations. The way forward? In *Proceedings of the 18th Workshop on Parallel and Distributed Simulation (PADS'04)*, ed. S. Kawada, 141-148. Los Alamitos, CA : IEEE.

Wahlborg, M. 1996. Simulation models: important aids for Banverket's planning process. In *Computers in Railways V*, ed. J. Allan, C. A. Brebbia, R. J. Hill, 175-181, (1). Southampton, United Kingdom : WIT Press.

ACKNOWLEDGMENTS

We acknowledge the support of the HTM Personenvervoer NV for this research. Contact person: Mr. Willard Kamerling; email address <w.kamerling@htm.net>. This research project has been funded by the BSIK-NGI program.

AUTHOR BIOGRAPHIES

ELISANGELA MIEKO KANACILO is a Ph.D. candidate at the Systems Engineering Group of Delft University of Technology. Her Ph.D. research is focused on developing a simulation environment to support the control design of rail infrastructures. 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 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>.