The Architecture and Components of LIBROS: Strengths, Limitations, and Plans

Huang, Y; Seck, MD; Verbraeck, A

Publication date
2010

Document Version
Accepted author manuscript

Published in
Proceedings of the 24th Annual European Simulation and Modelling Conference (ESM 2010)

Citation (APA)
Huang, Y., Seck, MD., & Verbraeck, A. (2010). The Architecture and Components of LIBROS: Strengths, Limitations, and Plans. In G. K. Janssen, K. Ramaekers, & A. Caris (Eds.), *Proceedings of the 24th Annual European Simulation and Modelling Conference (ESM 2010)* (pp. 80-87). Eurosis-ETI.

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

THE ARCHITECTURE AND COMPONENTS OF LIBROS: STRENGTHS, LIMITATIONS, AND PLANS

Yilin Huang, Mamadou D. Seck and Alexander Verbraeck
Systems Engineering Group
Faculty of Technology, Policy and Management
Delft University of Technology
PO Box 5015, NL-2600GA Delft
The Netherlands
E-mail: {y.huang,m.d.seck,a.verbraeck}@tudelft.nl

KEYWORDS
Railway Simulation, Simulation Library

ABSTRACT

Railway systems have long life spans, during which changes take place that lead to new issues to study. These changes can ask for the construction or alteration of simulation models for an analysis of the rail system. LIBROS is an open source java package that supports distributed microscopic multi-formalism simulation of heavy and light rail operations. Since its development, the library has been applied for simulations that successfully assisted decision making for the rail infrastructures design in a couple of projects. Each project focused on one specific part of a rail-based network. During the past year, LIBROS has been updated and extended as new simulation requirements emerged. This paper addresses the strengths and limitations of LIBROS by discussing its structural design, model components, functionality, and applications. Further research of using the DEVS formalism in LIBROS is proposed to transform the library for the future challenges of rail-based network design and simulation.

INTRODUCTION

Rail transport, as one of the major forms of public transport, plays a vital role that affects our daily life (Button and Hensher, 2001). In order to increase public transport's share compared to private transport modes and to maintain and improve its competitiveness, more reliable services should be offered (Tahmassseby, 2009). Modeling and simulation of transport systems have had important developments since the mid 1970s, and now received better recognition by transport designers in decision support (Ortzar and Willumsen, 2001). A microscopic rail network model is deemed not only suitable, but also mandatory, for exact running time calculation, timetable construction, and conflict detection and resolution (Hansen and Pachl, 2008). For large and complex rail-based networks, the planning and design of the infrastructure and operation are cumbersome and time-consuming; so is the modeling of the networks. Inevitably, working with complex infrastructure networks (total or partial) increasingly becomes a standard approach (Hansen and Pachl, 2008). A medium sized urban light rail operation, for example, may already concern a dozen lines and hundreds of vehicles and stops. A typical microscopic model contains all tracks of the routes, which have to be modeled at a high resolution at stations and junctions where two or more routes converge, diverge or cross-over. Each rail-based network is unique in terms of its technical specifications and available services (Ho et al., 2002). Different aspects of a network, such as the infrastructure, signaling control, and timetables at various locations and/or time periods, are full of variety. This further increases the complexity of rail-based network modeling. Moreover, because of the inherent long life span of rail infrastructure and services, new issues and questions often come up during the lifetime of the infrastructure. Many of these are about improving or at least maintaining the quality of service after the changes, e.g. by the adaptation of timetables, acquiring new equipment, and infrastructure changes and alterations. As such, model construction and reconstruction of rail-based networks particularly require flexible model composition and configuration in order to enhance reusability and reduce time and human resource investment in this regard.

LIBROS (Library for Rail Operations Simulation) is an open source java package that supports distributed microscopic multi-formalism simulation of heavy and light rail operations. It is designed for development of rail simulation models. The library development started at the request of HTM (The Urban Public Transport, The Hague, The Netherlands), as the rail simulation tools in existence could not meet the specific needs of urban tramway and light rail operation design. Since then LIBROS has been applied for simulations that successfully assist decision making for the design of rail-based infrastructures in a number of projects (Kanacilo and Verbraeck, 2006, 2007; Kanacilo and Oort, 2008; Huang et al., 2010). Each project focused on one specific part

of an urban tramway and light rail network, e.g. the modeling and analysis of a crossing where the infrastructure would be extended and design alternatives should be evaluated and compared. The results show that LIBROS is suitable to help forecast the operations, and it enables prediction of the quality of service of a certain urban rail infrastructure configuration (Kanacilo and Oort, 2008). This paper addresses the strengths and limitations of LIBROS. Its structure design, model components, functionality, and applications are discussed in relation with the strengths and limitations. LIBROS' underlying simulation environment DSOL (Distributed Simulation Object Library) (Jacobs, 2005) is briefly explained. Recent research (Seck and Verbraeck, 2009) added the DEVS (Discrete Event System Specification) formalism to DSOL which could facilitate the building and simulation of DEVS models for LIBROS. This would especially benefit LIBROS in terms of modularity and providing a hierarchical structure of model components. Given the challenges and the complexity of network simulation for rail infrastructure, an extension of LIBROS with the DEVS formalism is one of the main additions planned.

The remainder of this paper is organized as follows. In the next section, the motivation of developing the LIBROS simulation tool is explained. It is followed by a description of how rail simulations can be carried out with LIBROS. Some encountered challenges are stated. Section 4 describes DSOL and DEVS, and how DEVS-DSOL would benefit LIBROS. In Section 5, the architecture and main components of LIBROS are discussed, as well as some opportunities for enhancements of the simulation library.

THE NEED FOR LIBROS

In the field of rail transport network planning and design (or transport in general), a number of simulation tools have been developed, e.g. simulation models of stations or terminals (Carey and Lockwood, 1995; Carey and Carville, 2002, 2003; Rizzoli et al., 2002), and train network simulators, such as Simon/TTS (Wahlborg, 1996), TOPSim (Sandblad et al., 2000), SIMONE (Middelkoop and Bouwman, 2001), OpenTrack (Nash and Huerlimann, 2004), VirtuOS (Kavicka and Klima, 2000), RailSys (Bendfeldt et al., 2000), UX-SIMU (Kaas, 2000), Multi-train simulator (Ho et al., 2002), SimMETRO (Koutsopoulos and Wang, 2007).

The motivation of developing a new rail-based simulation tool is multi-faceted. First, many railway models and simulators are designed to assess a limited number of aspects (e.g. timetabling, signaling control) of rail operations or to study a particular part (e.g. a station, a junction) of the rail network (Ho et al., 2002). It is impractical to carry out various studies with different simulation tools. Some models have a high abstraction level (Vromans et al., 2006), which may cause a significant difference between model outcomes and real operations on a lower abstraction level (Ferreira, 1997). Although rail operations can be decomposed into different aspects, all should be taken into consideration in a self-contained simulation package for analyzing the rail network on the micro level (Krueger et al., 2000). Second, very few simulation tools support tramway or light rail operations. To the authors' knowledge, one of the few is RailSys (Rudolph, 2000). In comparison with heavy rail operations, many differences occur when simulating light rail operations. Heavy rail vehicles drive in signalled blocks, while light rail vehicles also "drive on sight" (Overton, 1989). Given the large number of cities with metro and tramway systems, there is a growing need for tools that specifically aim at light rail simulation. Third, tools designed for diverse transport agencies are very specific to individual agencies' needs, often making the tools less suitable for other agencies or transport operators. This asks for generic tools that can be applied for different situations and allow for analysis from different viewpoints. Fourth, commercial rail simulators generally have good performance, but they raise proprietary issues. Concerning inter-operability, they are difficult to be modified or linked with other tools or information systems such as databases or GIS (Kanacilo and Verbraeck, 2006). Cost is obviously another concern.

The research team thus decided to develop an open source rail simulation library. On the one hand, it is tailored for light rail simulation, considering the combined impact of different aspects of the infrastructure design in one self-contained simulation package. On the other hand efforts are taken to make the library also suitable for heavy rail simulation. The fact that LIBROS is developed as an open-source project provides a unique possibility to improve the package, and adds flexibility for further research. The package is available for any party to conduct research.

RAIL SIMULATION WITH LIBROS

The use of LIBROS is straightforward, as shown in Fig. 1. Users need to specify the simulation model and its parameters in XML format. They can define the rail infrastructure, control measures, timetables, and change options such as if animation and data visualization are needed. The model generator of LIBROS verifies the XML input, then creates and initializes the simulation model using the available model components in the package. If needed, other data sources such as timetables or GIS maps can be added. Experiments are generated according to the user configuration. In the model, the vehicle movement is simulated continuously and the other components such as the traffic lights and switches are simulated using event scheduling. The results can be animated, plotted, and (t-v, t-x, x-v) graphs are generated. Data files recording the vehicle movements, waiting times, etc., are generated and categorized.

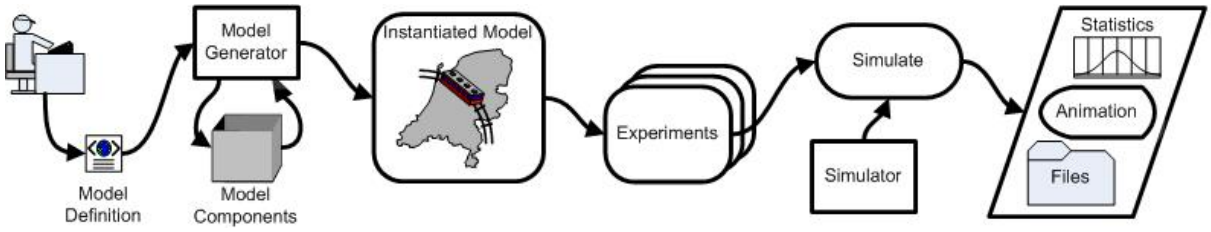

Figure 1: Rail Simulation with LIBROS

As stated earlier, LIBROS has been used in a number of projects that showed good results. Some challenges we encountered include the following. For example, the infrastructure configuration is XML-based, making the task difficult for non-experts. Here a GUI could help by providing drag-and-drop model components which has become quite common in commercial simulation tools. Modeling by means of drag-and-drop requires the relation between model components to be cut-and-dried. The DEVS formalism (Zeigler et al., 2000) provides a modeling theory of such modularity and hierarchical structure which the LIBROS model lacks. Some transformation of the library is therefore desired. The DEVS formalism could also benefit the library design in terms of information exchange between the components and model state saving. These issues are discussed in relation with the library design in the following sections.

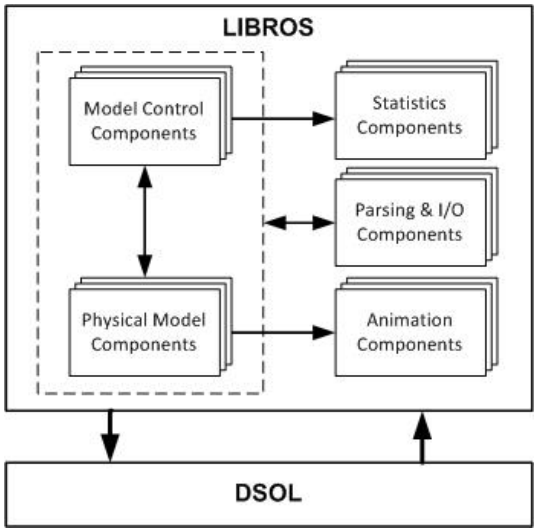

Figure 2: The LIBROS Architecture

DSOL AND DEVS

LIBROS is built as an extension of DSOL (Jacobs, 2005), an open source java simulation library that supports discrete and continuous formalisms, and provides generic simulation services such as various simulators, specification of experiments, event scheduling, and probability distributions. As a recent development (Seck and Verbraeck, 2009), DSOL ES-DEVS implements the parallel DEVS formalism on top of the DSOL library. DEVS (Zeigler et al., 2000) is a modeling and simulation formalism that allows for formal specifications of systems. Two levels of specifications are possible. The atomic DEVS formalism consists of (input/output/state) sets, and functions on the sets allowing complete and unambiguous specification of systems according to the discrete event abstraction. The coupled formalism consists of input, output, components (either atomic or coupled), and coupling relation sets. Along with the formalism, modularity is guaranteed and the closure under coupling property (Zeigler et al., 2000) allows for the construction of hierarchical models. The ES-DEVS simulation protocol is based on the event-scheduling worldview wherein executions of the internal transition function are scheduled according to the specified time advance function and unscheduled at the reception of external events. The ES-DEVS implementation strictly follows the DEVS formal specification, and the separation of concerns between models and simulators is respected. Dynamic structure DEVS is also implemented so that components and coupling relations can be added and removed dynamically during simulation runtime. This feature can be especially useful in dynamic systems such as transport systems in general and rail transport in particular, where the relations between the simulation components (e.g. a vehicle and a traffic light) are temporary and subject to change.

THE LIBROS SIMULATION LIBRARY

In the development process of LIBROS, an effort is made to overcome the issues addressed in Sect. and to design, develop and test a component-based, loosely-coupled rail simulation package. LIBROS uses part of the DSOL services, and extends them to define more rail specific simulation components. Fig. 2 illustrates a simplified component view of its architecture. The library contains two major groups of components: those that form the building blocks of the simulation model, which is the core of LIBROS; and those that offer peripheral simulation services such as parsing model definition files, generating statistics, animation, and outputs. The rail model structure can be divided into two layers. The first layer is a collection of the physical model components.

Most of them represent the (physical) rail and road infrastructure elements, such as tracks, stations, vehicles and intersections. But there can also be some virtual elements needed by the simulation, e.g., the locations where vehicles enter and exit the (simulation) system, or where data shall be collected for statistics. Instantiation of a model component creates the representation of a physical or virtual element and its initial state. The second layer is composed of the model control components, which define the control logic (i.e. state transition) of the physical model components in the first layer. The control logic can be rule-based or dependent on the interactions between different model components, e.g. the rules that define how several traffic lights shall coordinate their signals, how priorities are given to vehicles at junctions, the acceleration or deceleration of a vehicle according to the speed limits or based on obstacles. The separation of physical model and model control components simplifies the setup by which a physical model component can implement different model control strategies. A traffic light, for example, may have fixed time intervals for signal changes, or change signals depending on the traffic conditions. In this way, components can be easily extended and updated. The communications between different model components are also handled by the model control components using the publish-subscribe interaction scheme, which is discussed later. LIBROS provides XML schema for model configurations. The XML definition is parsed and verified by the input (processing) components, which then creates models and simulation replications. A physical model or model control component is associated with one or more statistics components or animation components (if needed). The statistics components collect important model states, and generate graphs of the key performance indicators for the rail network operation. The animation components are able to plot GIS data as background maps, and display the signalling changes, vehicle movements, etc. Output components generate and save the simulation results including graphs. Fig. 3 shows a simplified diagram of the model components. (Some interfaces and classes are omitted for clarity.) As mentioned earlier, the model structure has a physical layer and a control layer.

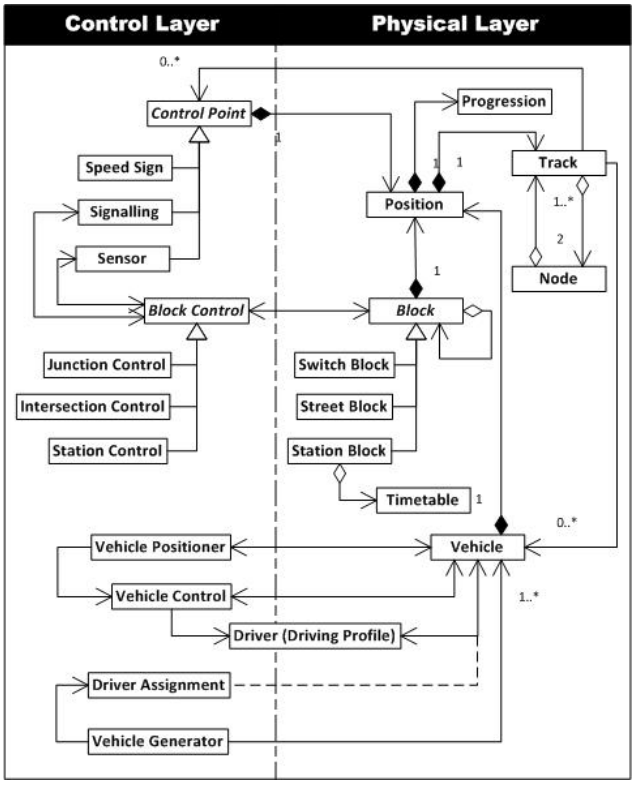

Figure 3: Overview of the LIBROS Components

Infrastructure Modeling

An accurate infrastructure model is important for rail simulation, as it is the basis for all calculations. In LIBROS, a combination of link-oriented and node-oriented approaches is chosen, as both have advantages and disadvantages (Hansen and Pachl, 2008). A rail network is a directed graph $N = (V, T)$ following Bang-Jensen and Gutin (2009). V is a set of vertices (or *node*s). T is a set of directed track segments (or *track*s). Each track is an ordered pair of distinct nodes (i.e. the two ends of the rail axis). The location (real world coordinates) of a track is saved in its starting and ending nodes. A rail switch is located at a node $v \in V$ with an out-degree $d_N^-(v) = 2$. A (switch) node contains information of which direction (i.e. the next track) a vehicle shall move to. The *position* of the other infrastructure elements (e.g. stops, traffic lights, sensors, speed limits) are saved in association with tracks. The *progression* is defined as the distance between the element and the starting node of the track.

Strengths: The infrastructure model is microscopic. This enables precise calculation of running times and evaluation of control strategies. High quality animation is possible; an example is shown in Fig. 4.

Limitations: The infrastructure configuration is time-consuming. Each node is defined by coordinates. Track lengths are calculated by end nodes and curvatures. Detailed geographical data are required for model setup.

Plans: The model should allow for simpler track definition, e.g. with only lengths and speed limits. This will reduce the complexity of the infrastructure configuration. Methods can be added for track information refinement if more detailed visualization of the infrastructure is desired.

Block System Modeling

In railway control, a block system is a signaling system that provides safe spacings for vehicles (Hansen and Pachl, 2008). A block section is a section of track where

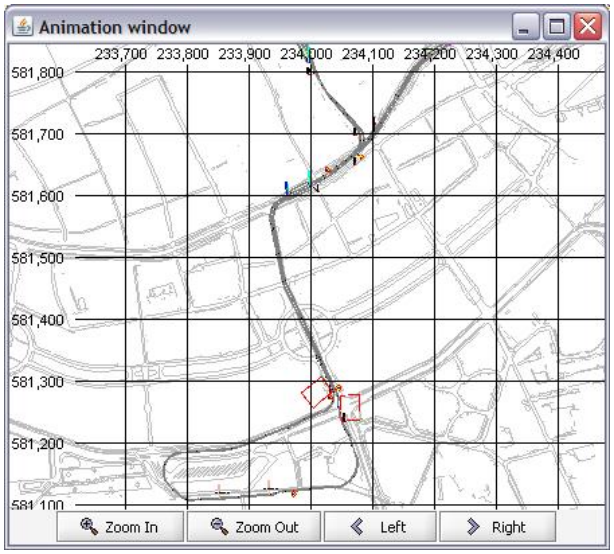

Figure 4: An Example of the Animation

a vehicle may only enter when the section is not occupied by other vehicles (Pachl, 2002); each section is guarded by a signal at the entrance. In the model, the concept is extended. Interlocking (at e.g. junctions and for single-tracks) also uses the concept of *block* and *block control*, because from a modeling perspective, block systems and interlocking operate in a very similar way, i.e. vehicles may not enter a locked section, and the signalling logic of different traffic lights in one area is often interdependent. The block control components define the diverse signalling logic. It can be traffic dependent. At major intersections in a city, trams may not have priority over the other street traffic; hence users can configure the vehicles' waiting time distributions. Alternatively, the signalling logic can depend on the occupation state of another block. In this case, users can define the dependencies between different blocks using sub-blocks. A simple example is shown in Fig. 5. The crossing is guided by 3 traffic lights (thus 3 blocks); 4 driving directions are possible. Three sub-blocks are defined in the example. Passing through direction 1 needs sub-block 1; direction 2 needs sub-blocks 2 and 1; directions 3 and 4 need sub-blocks 3 and 2. The logic is very intuitive: when a sub-block is occupied by a vehicle, it can not be used by another. Thus the state of the traffic light depends on the availability of the required sub-blocks. The release time of a sub-block is identified precisely, e.g. once a vehicle at direction 2 releases sub-block 2 (i.e. the vehicle's tail left sub-block 2, but sub-block 1 is still occupied), the access of direction 3 or 4 is granted if there is any request. A vehicle requests the access of a block by triggering a request sensor placed in front of the traffic light. If the access is granted, the required sub-blocks are set to be locked immediately. The *sensor*s can be of different types. The most common ones are request sensors and release sensors. The former are

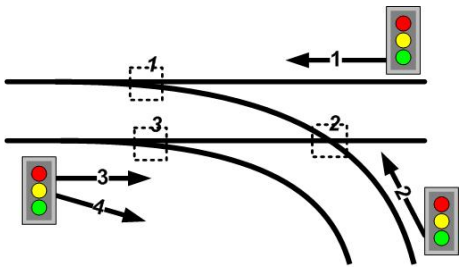

Figure 5: Block Control using Sub-blocks

triggered by a vehicle's head, and the latter by a vehicle's tail. If there are conflicts at a crossing and requests are queued, the access is granted to the vehicle that has the highest priority.

Strengths: The block (and sub-block) components provide users the possibility to configure different signalling logic for e.g. crossings, stations, single tracks, and any combination of these infrastructures. The signalling logic for sophisticated infrastructure settings, e.g. where different light rail lines intersect in city centers and near train stations, has been modeled by this method.

Limitations: Configuration of block system models is hard for non-experts. The model components lack modularity. For example, to model a crossing, each block (sub-block), switch (direction), and sensor have to be configured individually.

Plans: Higher level components should be constructed. The authors are aware of the complexity of rail infrastructures. Because each element is unique, constructing components for all situations is impossible and unnecessary. But for some standard and often occurring situations, components can be designed to simplify the configuration. It is important to separate the functional definition of a component with its geoinformation.

Vehicle Modeling

The *vehicle* component saves the vehicle's state and other information used for simulation and statistics, e.g. its size, speed, position, and "visible" objects (speed signs, signals, or a vehicle in front). Vehicles are generated according to timetables.A probability distribution can be introduced to simulate the earlier and delayed departure. The *vehicle generator* also assigns each vehicle a *driver/driving profile* that determines how a vehicle accelerates, cruises, and brakes. The vehicle movement is calculated by *vehicle positioner* using the Runge-Kutta integrator. Given the speed of a vehicle at time t_n, it computes the vehicle's speed and position (distance) at time t_{n+1} based on the vehicle's acceleration changes during the integration time-step. The acceleration rates are determined by the *vehicle controller*. As stated before, the vehicles can "drive on sight", meaning that a vehicle "sees" also other objects besides traffic lights. The objects being considered in the model are

(1) traffic lights, (2) changes of infrastructure, e.g. stops or stations, curves, (3) speed signs, (4) obstacles, e.g. a vehicle in front, and (5) other objects, e.g. an intersection or a switch, that by regulation a vehicle shall pass through with reduced speed. In principle, at each integration time-step, the vehicle "checks" the tracks in front of it, whether there are any objects associated with the tracks. The vehicle controller then decides if the vehicle shall accelerate, cruise or brake.An object may change state, e.g. a traffic light or a vehicle. If so, the object notifies the approaching vehicle using the publish-subscribe interaction scheme.
Strengths: Driving profiles introduce randomness into the driving behavior. "Drive on sight" is important for modeling light rail operation. The calculation of vehicle movement is precise and in detail.
Limitations: Checking acceleration and integration at each time-step has a high computational cost. Profiling the simulation program shows that this part of the calculation takes 74% of the execution time. The performance decreases when simulating large scale networks with many vehicles.
Plans: More efficient algorithms can be developed. A vehicle can check as far as possible until an object of interest is found, e.g. a traffic light, a sharp curvature, or another vehicle. Once an object of interest is found, the vehicle decides if and when it shall accelerate or brake; and the object "informs" the vehicle if the state of the object has any change, based on which the vehicle may react.

Asynchronous Messaging

The object-to-object communication in LIBROS (and DSOL) is accomplished by the publish-subscribe (or event notification) interaction scheme. With systems based on this scheme, subscribers register their interest in an event, or a pattern of events, and are subsequently asynchronously notified of events generated by the publishers (Eugster et al., 2003). The strength of this event-based interaction style lies in the full decoupling in time and space between publishers and subscribers (Eugster et al., 2003). Furthermore, asynchronous communication prevents disproportionate polling between objects and enables well tailored communication between potentially distributed objects (Jacobs, 2005). The publish-subscribe scheme is implemented following the *observer pattern* (also known as the *publish-subscribe pattern*) which defines a non static one-to-many dependency between objects so that when one object changes state, all of its dependents are notified and updated automatically (Gamma et al., 1994). For example, when simulating a vehicle approaching a traffic light, the state change of the traffic light is obviously of interest to the vehicle. Thus the vehicle registers to the traffic light's *subscribers list* and becomes a listener of the state changes of this traffic light in order to be notified. Once the vehicle passed the traffic light, it deregisters itself from the list. In LIBROS, not only the communication between the model components uses such a scheme but also the simulation services (animation, statistics, file generation, etc). Concerning implementation of the observer pattern (or the publish-subscribe scheme in general), common issues include unexpected updates (Gamma et al., 1994), thread-safety, and lapsed listeners (Goetz, 2005). When these issues are treated with prudence, the publish-subscribe interaction scheme is an efficient and convenient method for asynchronous communication.

Data-Driven Simulation

In a recent paper (Huang and Verbraeck, 2009), the authors proposed a dynamic data-driven approach for rail transport simulation. The idea is to automatically perform model calibration and validation by comparing the model output with the rail operation data. The model states at each state transition are saved to compare with the available data. The model is duplicated so that different model calibrations can be simulated in parallel to evaluate which parameter configuration is better. In this regard, efficient model state save and component copy are of importance. With the current library, the model states are periodically written to output streams for state saving. With the DEVS formalism, state changes are formally defined by the internal and external transition functions ($\delta_{int}, \delta_{ext}$), which make it easier to trace the state transition and causality. The model can be saved or copied at each state transition. Therefore, transformation of LIBROS model components by using the DEVS formalism would also benefit the development of automatic model calibration and validation.

CONCLUSIONS

This paper discussed the architecture and model components of LIBROS, and the strengths and limitations of its design. The library supports distributed rail simulation that uses configurable components as model building blocks. Not only does the library support rail transport design from an engineering perspective, its advanced animation and visualization capabilities makes it an efficient means of communication and enforces common understanding between transit authorities, service providers, as well as other parties involved. The library has been used in projects that showed good results (Kanacilo and Verbraeck, 2006, 2007; Kanacilo and Oort, 2008; Huang et al., 2010). Microscopic modeling offers simulation experiments with high detail that is important for timetable construction and conflict detection and resolution. The library currently models the vehicle movement using differential equations solved by numerical integrators. This solution offers high simulation precision but comes with a high computational cost, which hinders software performance. Infrastruc-

ture configuration can become very complex for the configuration of large scale rail networks. Both limitations can be mitigated by using the DEVS system theoretical formalism which offers a formal specification for modular and hierarchical discrete event systems. The transformation of LIBROS using the DEVS formalism is underway. When LIBROS uses DEVS formalism, the communication between model components would naturally use message transmission through ports. The communication between the simulation services will remain using the publish-subscribe scheme. Our next step is to extend the library for data-driven simulation through which automatic model calibration and validation can be performed (Huang and Verbraeck, 2009). In this context, the DEVS formalism will also benefit the library design.

ACKNOWLEDGMENTS

The authors would like to acknowledge the support of HTM Urban Public Transport, the Netherlands.

REFERENCES

Bang-Jensen J. and Gutin G., 2009. *Digraphs: Theory, Algorithms and Applications*. Springer Monographs in Mathematics. Springer Science, 2nd ed.

Bendfeldt J.P.; Mohr U.; and Mller L., 2000. *RailSys, a system to plan future railway needs. Advances in Transport*, 7, 249–255.

Button K.J. and Hensher D.A. (Eds.), 2001. *Handbooks in Transport 3: Handbook of Transport Systems and Traffic Control*. Elsevier Science.

Carey M. and Carville S., 2002. *Testing schedule performance and reliability for train stations. Journal of the Operational Research Society*, 51, no. 6, 666–682.

Carey M. and Carville S., 2003. *Scheduling and platforming trains at busy complex stations. Transportation Research Part A: Policy and Practice*, 37, no. 3, 195–224.

Carey M. and Lockwood D., 1995. *A Model, Algorithms and Strategy for Train Pathing. The Journal of the Operational Research Society*, 46, no. 8, 988–1005.

Eugster P.T.; Felber P.A.; Guerraoui R.; and Kermarrec A.M., 2003. *The many faces of publish/subscribe. ACM Computing Surveys*, 35, no. 2, 114–131. ISSN 0360-0300. doi:http://doi.acm.org/10.1145/857076.857078.

Ferreira L., 1997. *Planning Australian freight rail operations: An overview. Transportation Research Part A: Policy and Practice*, 31, no. 4, 335–348.

Gamma E.; Helm R.; Johnson R.; and Vlissides J., 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Goetz B., 2005. *Java theory and practice: Be a good (event) listener*. IBM Java Technology Technical Library.

Hansen I.A. and Pachl J. (Eds.), 2008. *Railway Timetable & Traffic: Analysis-Modelling-Simulation*. Eurailpress.

Ho T.; Mao B.; Yuan Z.; Liu H.; and Fung Y., 2002. *Computer simulation and modeling in railway applications. Computer Physics Communications*, 143, no. 1, 1–10.

Huang Y. and Verbraeck A., 2009. *A Dynamic Data-Driven Approach For Rail Transport System Simulation*. In M.D. Rossetti; R.R. Hill; B. Johansson; A. Dunkin; and R.G. Ingalls (Eds.), *Proceedings of the 2009 Winter Simulation Conference*. IEEE, 2553–2562.

Huang Y.; Verbraeck A.; van Oort N.; and Veldhoen H., 2010. *Rail Transit Network Design Supported by an Open Source Simulation Library: Towards Reliability Improvement*. In *Transportation Research Board 89th Annual Meeting Compendium of Papers*. 10-0310.

Jacobs P.H.M., 2005. *The DSOL simulation suite - Enabling multi-formalism simulation in a distributed context*. Ph.D. thesis, Delft University of Technology, the Netherlands.

Kaas A., 2000. *Punctuality model for railways. Advances in Transport*, 7, 853–860.

Kanacilo E.M. and Oort N.v., 2008. *Using a rail simulation library to assess impacts of transit network planning on operational quality*. In *WIT Transactions on the Built Environment*, WIT Press, 103. 35–43.

Kanacilo E.M. and Verbraeck A., 2006. *Simulation services to support the control design of rail infrastructures*. In *Proceedings of the 2006 Winter Simulation Conference*. IEEE, 1372–1379.

Kanacilo E.M. and Verbraeck A., 2007. *Assessing tram schedules using a library of simulation components*. In *Proceedings of the 2007 Winter Simulation Conference*. IEEE, 1878–1886.

Kavicka A. and Klima V., 2000. *Simulation support for railway infrastructure design and planning processes. Advances in Transport*, 7, 447–456.

Koutsopoulos H. and Wang Z., 2007. *Simulation of Urban Rail Operations: Application Framework. Transportation Research Record*, 2006, 84–91.

Krueger H.; Vaillancourt E.; Drummie A.M.; Vucko S.J.; and Bekavac J., 2000. *Simulation within the Railroad Environment.* In *Proceedings of the 2000 Winter Simulation Conference*. 1191–1200.

Middelkoop D. and Bouwman M., 2001. *Simone: Large Scale Train Network Simulations.* In *Proceedings of the 2001 Winter Simulation Conference*. IEEE, 1042–1047.

Nash A. and Huerlimann D., 2004. *Railroad simulation using OpenTrack. Advances in Transport*, 15, 45–54.

Ortzar J. and Willumsen L., 2001. *Modelling Transport.* John Wiley & Sons, 3rd ed.

Overton D., 1989. *Traffic signal control of LRVs.* In *IEE Colloquium on Light Rapid Transit On-Street.* 9/1–9/3.

Pachl J., 2002. *Railway Operation and Control.* VTD Rail Publishing.

Rizzoli A.E.; Fornara N.; and Gambardella L.M., 2002. *A simulation tool for combined rail/road transport in intermodal terminals. Journal of Mathematics and Computers in Simulation*, 59, no. 1-3, 57–71.

Rudolph R., 2000. *Operational simulation of light rail systems.* In *Proceedings of the European Transport Conference*. 167-178.

Sandblad B.; Andersson A.; Jonsson K.E.; Hellstrm P.; Lindstrm P.; Rudolf J.; Storck J.; and Wahl-borg M., 2000. *A train traffic operation and planning Simulator. Advances in Transport*, 7, 241–248.

Seck M.D. and Verbraeck A., 2009. *DEVS in DSOL: Adding DEVS operational semantics to a generic Event-Scheduling Simulation Environment.* In *Proceedings of the 2009 Summer Computer Simulation Conference.*

Tahmassseby S., 2009. *Reliability in Urban Public Transport Network Assessment and Design.* Ph.D. thesis, Delft University of Technology, The Netherlands.

Vromans M.J.C.M.; Dekker R.; and Kroon L.G., 2006. *Reliability and heterogeneity of railway services. European Journal of Operational Research*, 172, no. 2, 647–665.

Wahlborg M., 1996. *Simulation models: Important aids for Banverket's planning process.* In *Computers in Railways*, WIT Press, vol. V. 175–181.

Zeigler B.P.; Praehofer H.; and Kim T.G., 2000. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems.* Elsevier/Academic Press, 2nd ed.