Simulating SysML Transportation Models

Christos Kotronis, Anargyros Tsadimas, George-Dimitrios Kapos,
Vassilis Dalakas, Mara Nikolaidou, Dimosthenis Anagnostopoulos
Department of Informatics & Telematics
Harokopio University of Athens
70 El. Venizelou Str, 17671 Athens, GREECE.
{itp14110, tsadimas, gdkapos, vdalakas, mara, dimosthe}@hua.gr

***Abstract*—Model-based Systems Engineering (MBSE) promises efficient and effective systems development, by providing integrated system model views and streamlining the transition between different development activities. For instance, system testing with simulation should be provided as a simple facility for the performance evaluation of complex systems. Transportation systems are complex and their behavior is determined by dynamic factors. However, research efforts define simulation models for transportation systems, using custom or simulation-specific notation. Additionally, model-based approaches for transportation systems emphasize peripheral issues, such as safety conditions and environmental impact.**

In this work, a framework that enables seamless performance evaluation of Railway Transportation System (RTS) models via simulation is proposed. The de facto standard for MBSE modeling, Systems Modeling Language (SysML), is selected as the modeling infrastructure, while other standards, like Query/View/-Transformation (QVT), are used for the generation of executable simulation models. The latter can be simulated in Discrete Event System Specification (DEVS) simulators and the simulation results are returned in the RTS SysML model. Additionally, the application of the framework in the public RTS of Athens and the obtained simulation results are presented.

I. Introduction

The development process of a system comprises several activities, each addressing a different aspect. In the case of complex systems, transitioning from an activity to another can be tedious and require considerable effort for the re-generation of different models of the same system. Model-based Systems Engineering (MBSE) aims at developing and utilizing a common system model for all or most of the development activities. It, therefore, simplifies the transition between different activities and leads to an integrated, rigorous development process with minimum inconsistencies.

Transportation systems are complex and comprise of diverse sub-systems. In Railway Transportation System (RTSs), the complexity stems from and is continuously influenced by its constituent parts, coarsely identified as: stations, routes (lines), stops, moving trains and passengers. Thus, they can be described and analyzed, using a systems of systems approach. Systems Modeling Language (SysML) is appropriate for modeling the structure and behavior of such systems [1].

During the initial stages of the development process, testing, a common development activity, is performed virtually, via simulation. RTSs are usually analyzed as Discrete Event systems, where trains travel according to a fixed or dynamic schedule and passengers are generated using appropriate probability distributions [2].

Simulation can support decision making via the study of different parameters and scenarios. Furthermore, it can help uncover potential errors and problems, related to the operation of the system (e.g., problems stemming from large passenger load in a railway network). Thus, seamless simulation of design solutions can effectively improve the RTS development process. However, simulation models are often re-defined using custom or simulation-specific notation, diverging from the MBSE vision and its advantages.

The work presented in this paper focuses on the performance evaluation of RTSs, within a MBSE environment. To this end, a framework that can deliver such facilities is proposed. A SysML profile is proposed, for modeling RTSs, independently of any specific simulator. In the context of MBSE, the simulation-agnostic RTS model is used for the automated generation of Discrete Event System Specification (DEVS) simulation models and, eventually, executable DEVS-Java [3] simulation code that uses existing simulation library components. Automated simulation model generation from the RTS model is defined via a standards-based Query/View/-Transformation (QVT) [4] mapping. Additionally, incorporation of simulation results in the RTS model is provided, further reinforcing the role of the common system model.

The framework was developed, following the methodology presented in [5] and applied with success in the domain of Enterprise Information Systems (EIS) [6]. Domain-independent elements that have already been proposed in [6] are utilized in this work, while novel, domain-specific elements were developed to complete the framework for RTSs.

As a proof of concept the proposed framework was applied to the ATTIKO METRO [7] (underground) and Athens-Piraeus Electric Railways (ISAP) [8] (overground) RTSs, used for public transport in the city of Athens, Greece.

In the following section, an overview of the state of the art in transportation systems simulation and model-based transportation systems development are provided. In section III, the implemented framework for model-based simulation of RTSs is presented. Its application for the underground and overground public RTS used in the city of Athens, Greece, is presented, in section IV. Finally, our concluding remarks can be found in section V.

II. Related Work

A review of the relevant literature reveals the ongoing and increased interest in studying RTSs behavior through simulation. In [2], a custom, event-driven simulator for multi-line metro systems and its application to the Santiago de Chile metropolitan rail network are presented. In [9], the DEVS formalism and a compliant simulator are used for the simulation of magnetically levitated train systems. SIMARAIL, a discrete event simulation environment is proposed in [10] to apply simulation-based timetable optimization for the Iranian railway network. Distributed constraint-based railway simulation is proposed in [11] to achieve efficient testing in large railway networks. In a similar context, a discrete event model for representing RTSs, including their dynamic aspects, is proposed in [12], aiming at reducing the computational cost of simulation execution. Simulation performance is also addressed in [13], where the movement of a group of trains on a single railway line can be studied with less iterations and CPU time. In [14], the SIMAN/ARENA Discrete Event Simulation Tool is used in the Operational Planning of a RTS.

In the referenced approaches simulation models are defined and presented using either custom or simulator-specific notation. On the contrary, we propose the use of a standard system modeling language, namely SysML, to define various aspects of the transportation system and derive executable simulation models for evaluating the performance of system design solutions.

Relevant to the notion of this paper, few approaches propose the development of transportation systems in a generic, model-based manner. Focus is given on specific aspects of the transportation system operational characteristics. In [15], a model-based safety architecture framework for capturing and sharing architectural knowledge of safety cases in safety-critical systems of systems is proposed, utilizing SysML diagrams. The framework is applied in the Dutch high speed train lines. In order to analyze urban transportation and its environmental impacts, a comprehensive, interdisciplinary approach is proposed in [1]. The wide range of spatial and temporal scales and processes involved, lead to a multi-tiered approach and a cascade of models to describe alternative urban development and transportation scenarios. Different simulation tools are linked within a common indicator framework, enabling the subsequent multi-criteria evaluation.

In the work presented in this paper, the need for simulating RTSs, without requiring the manual definition and development of simulation models or custom simulation code, is addressed. The system model is defined in terms of SysML and the Railway Transportation Profile. The system model is utilized to derive executable simulation models for the DEVS formalism. Furthermore, the evaluation activity is facilitated, by importing the simulation results in the original SysML transportation model.

III. An integrated approach for transportation systems simulation

A. Short Review

In this paper, we aim at providing advanced evaluation facilities via simulation, during model-based development of RTSs. The methodology, presented in [5], was followed in order to meet the challenges of such an endeavor. The methodology instructs the development of a framework that enables (a) effective modeling in the selected domain, (b) automated generation and execution of simulation models, based on the system model, and (c) the incorporation of simulation results, for further processing. Such a framework comprises several components:

- A domain-specific SysML profile, to define the system models appropriately. Additional tools (plugins) may be required to further facilitate the modeling activity.
- The selection of a simulation framework that is (a) appropriate for the domain and (b) provides the means for high-level representation, i.e. a Meta-Object Facility (MOF) meta-model.
- The selection or development of simulation library components that would be useful for the construction of simulation programs to test systems in the domain.
- The development of a high-level conceptual mapping of system model elements to simulation model elements and the respective executable model transformation in terms of QVT.
- The development of extensions in a SysML modeling tool, to enable the incorporation of simulation results in the system model.

In this context, a framework for public RTSs was developed, comprising of:

- The Railway Transportation Profile for SysML.
- A plugin for the MagicDraw modeling tool, to provide enhanced functionality, during transportation systems modeling, like the automated supplementation of useful model elements (e.g. SysML ports into the system model elements and the incorporation of simulation results in the system model).
- The DEVS simulation framework that is appropriate for the domain (usually simulated with discrete event simulation) and provides a MOF meta-model for DEVS [5].
- A set of simulation library components and auxiliary classes for simulation execution and results aggregation.
- The transportation system elements were conceptually mapped to respective simulation components. This mapping was imprinted in high-level transformation, defined as a set of QVT relations.

This framework enabled the definition of the public RTS of Athens, Greece, and the derivation of useful simulation results. The elements of the developed framework are discussed in the rest of the section, while the case study is presented in the next section.

B. Modeling Transportation Systems with SysML

According to the Unified Modeling Language (UML) extension mechanism [16], a profile is defined using stereotypes, tagged values and constraints that are applied to specific elements of a model, such as classes, attributes and operations. To create a complete, accurate and practical profile the following were defined:

- All the entities that represent the real components of a system. In this work, the domain is the metro and urban railway public transportation, so our entities represent *Lines*, *Stations* and *Stops*.
- The characteristics and attributes of the aforementioned entities. For example, a *Line* component has a unique *Line* ID, *Line* name, etc. Some attributes are used to characterize the entity, while others hold auxiliary parameters about behavioral aspects of the model entities.
- The meta-classes (e.g., block or flow-port) that may be extended by one or more stereotypes via an association/extension relationship.
- Specific ports for the accurate modeling of the system's dynamic performance (material or information flow). In a RTS, most of the ports in each entity are used for the movement of *Trains* (e.g., entry, exit) or the commuting of passengers (e.g., exit a *Station* or embark on a *Train*).
- The information flows. They are useful for the establishment of a connection between specific ports of particular entities. For example, a *Station*'s ports are connected to its respective *Stops'* ports to implement the flow of passengers from one entity to another.

The basic entities of the defined profile are the *Line*, the *Station* and the *Stop*. *Lines* function as sequences of *Stops* and *Train* generators. When a *Train* is generated, it is positioned on the first *Stop* of the *Line*. Afterwards, it follows the sequence of *Stops* (using the rails) in its assigned direction until it reaches the end of the *Line*.

The *Station* is the main entity in our model and the RTS in general. It is closely linked to the *Stops*. It also represents the area where passengers are validating tickets and are led either to the *Stops*' platforms or to the exit and the out-of-RTS world. *Stations* are categorized into initial/terminal, ordinary and transit. The majority of *Stations* are ordinary and contain a single *Stop* with two opposite-direction platforms. The same observation applies to initial/terminal *Stations* where all the passengers disembark. Finally, transit *Stations* contain *Stops* that belong to multiple *Lines*. Using these *Stations* passengers can change direction and move to other *Stops* and, subsequently, to other *Lines*.

Last but not least, *Stops* correspond to the platforms of the RTS. Namely, a *Stop* represents the area where passengers are waiting to board an incoming *Train* or disembark from it. If they board the *Train* they commute to the next *Stop*, except if the corresponding *Station* is terminal. Otherwise they leave the platforms, enter the *Station* and either exit the system or change *Line*. Each *Stop* is owned exclusively by a specific *Line*.

The entities described above define the structural part of the system. They define the infrastructure, where *Trains* - the only moving parts of the system- can travel to provide the transportation service. In general, *Trains* can pick up passengers and transport them to another *Stop* following a specific route (*Line*). It is important to note that passengers are not treated as autonomous entities. When a *Train* reaches a *Stop*, the attributes storing numbers of passengers are updated accordingly, i.e., the capacity of the *Train* and the number of people on the platform are updated. When they reach the terminal *Station*, *Trains* can be withdrawn for maintenance.

Fig. 1 provides an overview of the aforementioned profile and its stereotypes. The basic entities, their relationships, their corresponding attributes and ports and the extended meta-classes are illustrated.

C. Simulation-Related Issues

According to the adopted methodology, library components, implementing the simulation behavior, should be utilized for the generation and the execution of the simulation. Apart from the identified main entities of the transportation profile, several auxiliary components are also required.

In general, DEVS simulation models may contain DEVS Atomic and DEVS Coupled elements [17]. DEVS Atomic elements define the behavior of the system. Moreover, they contain input and output ports, phases and specific functions that define the advancement of the simulation time and the transitioning of these phases. DEVS Coupled models are composite models that contain other DEVS Atomic or Coupled models and the couplings between them (e.g., internal/external couplings through SysML ports). In the Coupled models there are messages that simultaneously cross these coupling paths between the Atomic models and contain information or objects that travel along the couplings.

In the following, the developed library components are briefly described:

- *Simulation Controller*. It controls the time and repetitions of the simulation. In addition, the *Controller* can initiate or terminate the operation of all the other entities via signals. A typical example is the signal that the *Controller* sends to *Train* and *Passenger Generators* so as to inform them to start generating *Trains* or passengers.
- *Statistical Data Collector*. It collects all the statistics (number of passengers, etc.), produced during or after the simulation execution. When the *Collector* finishes gathering the appropriate data from the library components, it converts the results to XML and CSV files for further analysis. It is also responsible for the computations of the total number of passengers that changed *Line* and that were generated as well as the total number of *Trains* that were produced.
- *Passenger Generator*. It generates passengers that commute in the railway system, implementing the entrance of passengers at *Stations*. Due to the undetermined time and number of passengers that enter the system, the generator

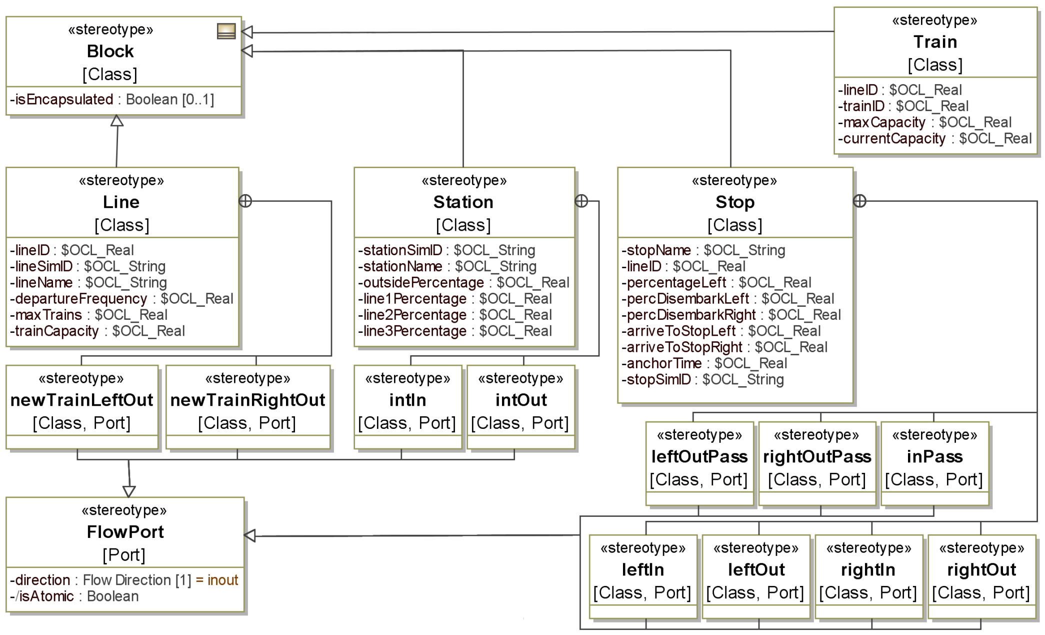

«stereotype» Block [Class]
-isEncapsulated : Boolean [0..1]
«stereotype» Train [Class]
-lineID : $OCL_Real
-trainID : $OCL_Real
-maxCapacity : $OCL_Real
-currentCapacity : $OCL_Real
«stereotype» Line [Class]
-lineID : $OCL_Real
-lineSimID : $OCL_String
-lineName : $OCL_String
-departureFrequency : $OCL_Real
-maxTrains : $OCL_Real
-trainCapacity : $OCL_Real
«stereotype» Station [Class]
-stationSimID : $OCL_String
-stationName : $OCL_String
-outsidePercentage : $OCL_Real
-line1Percentage : $OCL_Real
-line2Percentage : $OCL_Real
-line3Percentage : $OCL_Real
«stereotype» Stop [Class]
-stopName : $OCL_String
-lineID : $OCL_Real
-percentageLeft : $OCL_Real
-percDisembarkLeft : $OCL_Real
-percDisembarkRight : $OCL_Real
-arriveToStopLeft : $OCL_Real
-arriveToStopRight : $OCL_Real
-anchorTime : $OCL_Real
-stopSimID : $OCL_String
«stereotype» newTrainLeftOut [Class, Port]
«stereotype» newTrainRightOut [Class, Port]
«stereotype» intIn [Class, Port]
«stereotype» intOut [Class, Port]
«stereotype» leftOutPass [Class, Port]
«stereotype» rightOutPass [Class, Port]
«stereotype» inPass [Class, Port]
«stereotype» FlowPort [Port]
-direction : Flow Direction [1] = inout
-/isAtomic : Boolean
«stereotype» leftIn [Class, Port]
«stereotype» leftOut [Class, Port]
«stereotype» rightIn [Class, Port]
«stereotype» rightOut [Class, Port]

Fig. 1. Railway Transportation Profile entities

randomly generates passengers, according to the selected probability distribution.

- *Station*. It is the main entity in our model and is closely linked to the *Passenger Generator* and the *Stops*. The main computations, implemented in this component, involve the new and commuting passengers, i.e., the number of new passengers (produced by the *Generator*), entering the *Station*, and the number of passengers that change direction and move to other *Stops* (and subsequently to other *Lines*).
- *Stop*. *Stops* correspond to the platforms of the RTS. This component provides a large part of the simulation computational part. In a *Stop*, passengers wait for an incoming *Train* to embark (except if the corresponding *Station* is terminal), while the passengers that are already on the *Train* disembark. If the waiting passengers board the *Train*, they commute to the next *Stop*. The passengers that disembark from it, exit the platforms, enter the area of the *Station* and either leave the system or change *Line*.
- *Line*. When a *Train* is generated, it is driven via a specific *Line* to its first *Stop*. In the meanwhile the *Line* informs the *Collector* that a new *Train* has been produced. Afterwards, the *Train* follows the sequence of *Stops* (using the rails) in its assigned direction until it reaches the end of the *Line*. The computational part of this component produces statistics for the number of the generated *Trains* and their average capacity.
- *Line-Train-Sink*. After a *Train* has traversed the entire length of a *Line*, it exits the terminal *Station* and it is driven to the *Line-Train-Sink*. At this point, it stops its function. Furthermore it informs the *Statistical Data Collector* that a *Train* has reached the end of the *Line*. Thus, this component is used for the update of a *Line* and its current moving *Trains*.
- *Train*. As mentioned in subsection III.B it is the only moving part of our transportation system. After the passengers embark, the *Train* transports them to the next *Stop* running along the rail track (*Line*). This component includes the characteristics of a *Train*, i.e. a unique ID and its maximum and current capacity. The latter is updated according to the number of passengers embarking or disembarking at *Stops*.

In general, the *Passenger Generator*, the *Station*, the *Stop*, the *Line* and the *Line-Train-Sink* are connected to the *Simulation Controller* and the *Statistical Data Collector*. In addition, the *Controller* and the *Collector* are linked together. Furthermore, the *Passenger Generator* is connected to the *Station* and the *Station* to the *Stop* (and vice versa). Also, the *Line* and the *Line-Train Sink* are linked with the corresponding initial and terminal *Stops*. All the interconnections are materialized through DEVS ports. The *Trains* are traveling through these ports in the form of messages. While this movement takes place, number-formed passengers enter the system, commute via the *Trains* and exit the system. Finally, as a technical side note, all of the aforementioned components are DEVS Atomic. Provided their availability, a Configuration class is required to combine and couple them accordingly.

D. Generating Simulation Models from Transportation System Models

QVT is a language for defining relations between a source and a target model, conforming to respective meta-models [4]. In this case, the source model is a RTS SysML model and the target model is a DEVS simulation model. A set of

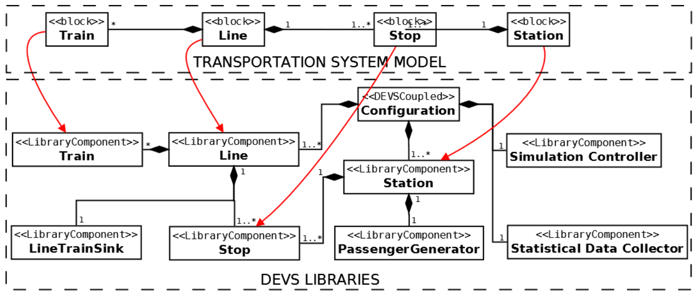

TRANSPORTATION SYSTEM MODEL
<<block>> Train
<<block>> Line
<<block>> Stop
<<block>> Station
DEVS LIBRARIES
<<DEVSCoupled>> Configuration
<<LibraryComponent>> Train
<<LibraryComponent>> Line
<<LibraryComponent>> Station
<<LibraryComponent>> Simulation Controller
<<LibraryComponent>> LineTrainSink
<<LibraryComponent>> Stop
<<LibraryComponent>> PassengerGenerator
<<LibraryComponent>> Statistical Data Collector

Fig. 2. Mapping SysML transportation entities to DEVS

Listing 1. Excerpt of the DEVS simulation model

```
<?xml version="1.0" encoding="UTF-8"?>
<Devs:MODEL xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:Devs="urn:DEVS_MM.ecore"
    xsi:schemaLocation="urn:DEVS_MM.ecore DEVS_MM.ecore">
  <DEVS_COUPLED>
    <MODEL_NAME text="Transportation_System"/>
      <COMPONENT_REFERENCE_LIST>
        <COMPONENT_REFERENCE text="Line"
                   xsi:type="Devs:T_Component_Reference">
          <LIBRARY_COMPONENT class="Line">
            <INIT_PARAMS>
              <INIT_PARAM name="lineID"
                     xsi:type="Devs:T_Value_Init_Param">
                <VALUE type="Real" value="3"/>
              </INIT_PARAM>
              <INIT_PARAM name="lineName"
                     xsi:type="Devs:T_Value_Init_Param">
                <VALUE type="String" value="Line 3       ↩
                     Doukissis Plakentias-Aghia Marina"/>
              </INIT_PARAM>
            </INIT_PARAMS>
          </LIBRARY_COMPONENT>
        </COMPONENT_REFERENCE>
        ...
      </COMPONENT_REFERENCE_LIST>
    <INTERNAL_COUPLING>
      <INTERNAL_COUPLING_SPEC>
        <INTERNAL_OUTPUT component="Line"
                         port="NewTrainLeftOut"/>
        <INTERNAL_INPUT component="Stop" port="leftIn"/>
      </INTERNAL_COUPLING_SPEC>
      ...
    <INTERNAL_COUPLING>
  </DEVS_COUPLED>
</Devs:MODEL>
```

QVT relations were defined, in order to convert a hierarchy of entities, describing the whole RTS model to the respective DEVS Coupled model. The latter consists of other DEVS models either Atomic or Coupled. Fig. 2 provides an abstract, visual representation of this mapping. In some cases there is one-to-one mapping between the system model and DEVS elements (*Stop*, *Train*), while, in others, DEVS elements are implemented as a composition of basic and auxiliary simulation components. An example of the generated DEVS model, as the outcome of the transformation, is presented in Listing 1, in XML Metadata Interchange (XMI) format.

Eclipse Indigo Edition and its extension, MediniQVT, were used for the definition and execution of the QVT transformation from RTS SysML to DEVS models.

In order to produce executable Java code from the DEVS simulation model, the EXtensible Stylesheet Language Transformations (XSLT) transformation, referenced in [6], was used. The input of the transformation is the XMI representation of the DEVS model and the output is the Java code for respective DEVS Coupled component that uses, initializes and interconnects the simulation library components.

In addition, users have the ability to interact with the simulation, controlling the execution in real-time, pausing it and changing the time advancement frequency.

E. Simulation Results Recording and Utilization

The simulation results, collected during simulation execution, are divided into two categories, i.e., single and cumulative. Single results are produced and shown during the execution of the simulation, namely, at run-time. They represent generated and commuting passengers quantitatively as numbers. Cumulative results are obtained after the completion of the simulation. They illustrate total, average and maximum numbers of passengers that commute at *Stops* and *Stations*. Cumulative simulation results are extracted and stored in an Extensible Markup Language (XML) document. The majority of the simulation results, both single and cumulative, are mainly produced by the entity *Stop*. The single results represent passengers that were already in a *Stop* or in a *Train*, that embark on and disembark from a *Train* and those who remain at a *Stop*. Moreover, the current capacity of a moving *Train* and the number of passengers that disembark at a terminal *Stop* are also extracted. Regarding the cumulative results, via the computations implemented in *Stops*, we obtain total, average and maximum numbers and percentages of passengers that remained at a *Stop*, missed a *Train* and commuted through *Stops*.

Using the plugin and the entities' simulation ID, the cumulative results are integrated back to the SysML transportation model. The main concept is that the obtained results are applied to the corresponding entities as new attributes. For example, after the execution of the simulation, new specific attributes were obtained for the *Statistical Data Collector* entity. These attributes represent particular results for that entity (e.g., StatsTotalNumPassCommuting, StatsTotalNumPassChangedLine, etc).

In parallel, the results are also transformed into CSV format in order to simplify the creation of the related diagrams. In general, four CSV documents are constructed to store the cumulative results of the *Lines*, the *Stations*, the *Stops* and the *Statistical Data Collector*.

IV. Case Study: the Athens Metro Network

The Athens RTS is composed of three *Lines*. Moreover, sixty-one *Stations* comprise the entire underground and overground transportation system of metro and urban railway. Six of them are initial/terminal, four are transit and the rest of them are ordinary *Stations*. The constructed model can be considered as a comprehensible, easy-to-use and flexible schema that can be easily extended. Without the redundant information of a bulky and obscure model, the system analyst has the ability to figure the system out, evaluate its entities

Listing 2. Excerpt of the XML results

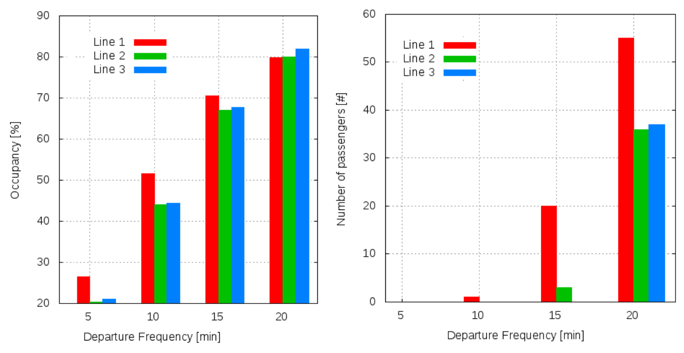
```
<results>
    <RESULT id = "_ 17_0_58_347" name = "Collector" ↩
        stereotype = "Collector">
        <VALUE>
            <Name> StatsTotalNumPassGenerated </name>
            <Value> 1330645 </value>
        </VALUE>
        <VALUE>
            <Name> StatsTotalNumPassChangedLine </name>
            <Value> 170634 </value>
        </VALUE>
        <VALUE>
            <Name> StatsTotalNumTrains </name>
            <Value> 1346 </value>
        </VALUE>
    </RESULT>
</results>
```

and insert desired requirements to detect errors and problems in it.

Using the design environment, the analyst can define the parameters of the simulation. After the execution of simulation, simulation results are incorporated into the appropriate predefined attributes of the system model. They mainly represent statistics about the basic RTS entities.

An excerpt of the simulation results is presented in Listing 2. These specific results are related to the *Statistical Data Collector* library component. The diagrams illustrated in Fig. 3 are produced using the simulation results from the library components. Fig. 3 (a) illustrates the change of the occupancy of *Trains* according to their generation frequency in each *Line*. In general, executing the simulation with different scenarios and changing the *Train* generation and departure frequency (minutes), the following may be observed: when the *Trains* arrive, one after the other, at a higher time interval, more passengers are generated, gathered on the platforms (*Stops*) and board the incoming *Train*, increasing its occupancy.

Furthermore, using the same changing frequencies and scenarios, another interesting observation can be made: infrequent *Train* arrivals cause increased passenger accumulation at *Stops*. When the incoming *Train* reaches full occupancy, due to lack of capacity, many passengers that could not get aboard will have to wait for the next *Train*. Fig. 3 (b) shows the average number of passengers per line that miss the *Train*.



(a) (b)

Fig. 3. Train Occupancy per Line (a) and avg. Number of passengers that missed the train (b) per Departure Frequency

V. Conclusions

Transportation systems are complex dynamic systems comprising different subsystems and components strongly influenced by the human factor. Here, we propose a model-based framework for enhanced modeling and performance evaluation facilities for RTSs. A SysML profile is used for RTS modeling, while executable simulation models are generated, using QVT, MOF and DEVS. Furthermore, the system model is complemented with the simulation results.

Future work involves the evolution of the framework, to include more types of public transportation systems, i.e. buses, trams, etc, allowing the study of public transportation through a wider perspective. Moreover, the approach could be further extended to include (inter)national transportation systems that provide passenger flows to the local transportation systems.

References

[1] K. K. Azevedo, "Modeling sustainability in complex urban transportation systems," 2010.

[2] P. Grube, F. Núñez, and A. Cipriano, "An event-driven simulator for multi-line metro systems and its application to santiago de chile metropolitan rail network," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, pp. 393–405, 2011.

[3] J. Mather, "The devsjava simulation viewer: A modular gui that visualizes the structure and behavior of hierarchical devs models," Ph.D. dissertation, UNIVERSITY OF ARIZONA, 2003.

[4] OMG, "Meta object facility (MOF) 2.0 Query/View/Transformation specification," *Transformation*, no. April, pp. 1–230, 2008. [Online]. Available: http://www.omg.org/spec/QVT/1.0/PDF/

[5] G.-D. Kapos, V. Dalakas, M. Nikolaidou, and D. Anagnostopoulos, "An integrated framework for automated simulation of SysML models using DEVS," *Simulation*, vol. 90, no. 6, pp. 717–744, 2014.

[6] A. Tsadimas, G.-D. Kapos, V. Dalakas, M. Nikolaidou, and D. Anagnostopoulos, "Integrating simulation capabilities into sysml for enterprise information system design," in *System of Systems Engineering (SOSE), 2014 9th International Conference on*. IEEE, 2014, pp. 272–277.

[7] "ATTIKO METRO S.A." [Online]. Available: http://www.ametro.gr/page/default.asp?id=4&la=2

[8] "Athens-Piraeus Electric Railways." [Online]. Available: http://www.stasy.gr/index.php?id=33&no_cache=1&L=1

[9] M. H. Cha and D. Mun, "Discrete event simulation of maglev transport considering traffic waves," *Journal of Computational Design and Engineering*, vol. 1, no. 4, pp. 233–242, 2014.

[10] A. Sajedinejad, S. Mardani, E. Hasannayebi, K. Mir Mohammadi, S. A. Reza, and A. Kabirian, "Simarail: simulation based optimization software for scheduling railway network," in *Simulation Conference (WSC), Proceedings of the 2011 Winter*. IEEE, 2011, pp. 3730–3741.

[11] H. Schlenker, "Distributed constraint-based railway simulation," in *Applications of Declarative Programming and Knowledge Management*. Springer, 2005, pp. 215–226.

[12] J. L. Espinosa-Aranda and R. García-Ródenas, "A discrete event-based simulation model for real-time traffic management in railways," *Journal of Intelligent Transportation Systems*, vol. 16, no. 2, pp. 94–107, 2012.

[13] X. Xiao-Ming, L. Ke-Ping, and Y. Li-Xing, "Discrete event model-based simulation for train movement on a single-line railway," *Chinese Physics B*, vol. 23, no. 8, p. 080205, 2014.

[14] F. E. M. Martínez and B. Colucci, "Final report application of siman arena discrete event simulation tool in the operational planning of a rail system."

[15] K. Schuitemaker, J. Braakhuis, and M. Rajabalinejad, "A model based safety architecture framework for dutch high speed train lines," in *System of Systems Engineering Conference (SoSE), 2015 10th*. IEEE, 2015, pp. 24–29.

[16] O. M. G. Inc, "UML 2.0 Superstructure Specification," October 2004.

[17] T. G. Kim and B. P. Zeigler, "The DEVS formalism: hierarchical, modular systems specification in an object oriented framework," in *WSC '87: Proceedings of the 19th conference on Winter simulation*. New York, NY, USA: ACM Press, 1987, pp. 559–566.