TRANSPORT SYSTEM REALIZATION IN SIMEVENTS TOOL

K. Valigura, M. Foltin, M. Blaho

Slovak University of Technology in Bratislava,
Faculty of electrical engineering and information technology

Abstract

SimEvents tool allows modeling and simulating discrete-event systems. Such discrete-event systems are flexible manufacturing systems or transport systems. In this article we will focus on transport system that represent rail yard in which we can simulate movement of three trains.

1 SimEvents tool

SimEvents [1] extends Simulink with tools for modeling and simulating discrete-event systems. By SimEvents it is possible to create a discrete-event simulation model to simulate the passing of entities through a network of queues, servers, gates and switches based on events. SimEvents and Simulink provide an integrated environment for modeling hybrid dynamic systems containing continuous-time, discrete-time and discrete-event components.

Discrete-event simulation contains discrete items that are called entities. Each entity can carry data known as attributes. Every passing of entity through block is understood as event that represents immediate discrete incident. This incident changes state of variables, outputs or occurrence of next events.

2 Transport system and SimEvents

Created transport system model is rail yard that allows simulate movement of three trains. Trains are represented by entities and their passings between single parts of rail yard are events. As was said in previous chapter, every entity can carry data. In this case those data contain information about individual trains. All of this information contain number of train, train speed, total moved distance, routing number for train routing on the next rail switch and total time of train movement. Some of mentioned data is assigned to individual entity as an input parameter and others are preconfigured in m-file that is used for initialization.

3 Created function blocks

Let's take a fact that each rail yard consist of three main parts. Those parts are stations, rail switches and single sections of rails. Each of those main parts is represented by separate module. This way we can assemble arbitrary rail yard using created modules. Single modules are connected with each other by entity connections.

Beyond modules that represent main parts of rail yard we created some special modules as well. Those special modules are integrated into main modules. They are additions to SimEvents blocks to achieve required functionality. All created modules main and special are in fact subsystems that consist of SimEvents and Simulink blocks.

Main modules

The first from main modules is **Station** (figure 1) that represents section of rails ended with station. Its function is to act as start and destination point of individual trains. This module beyond entity ports has one signal port as well. Signal in this port carry information about train number which ended up in this station, its total moved distance and total time.

Figure 1: Station module

Figure 2 depicts internal structure of this module. As is shown, structure is separated into two ways. The upper one represents case that given module is destination station for entity. At the beginning is ensured using Enabled Gate block that entity is only entity in given module. This way only one entity can ends its way in this module or in other words only one train is allowed to end its way in given station. Enabled Gate block is controlled by value of input signal. The signal has value 0 in case no entity is in module, otherwise its value is 1. The presence of entity is verified by two values. They are input parameters for function generating gate controlling signal. The first value is number of entities that departed from Enabled Gate block. The number bigger than 0 indicates that some entity is already in module. The second value is number of entities in Single Server block. This block delays entity that departs given module. Alternatively, if one train is departing station other train cannot arrive into this station. It is possible only after the first train departs from station. From Enabled Gate block entity continues to Get Attribute block where speed, distance of section and total distance are read. All of this attributes are input parameters for Computing module. In Computing module total moved distance is increased by distance of given section and total time needed to pass through this section is computed. In Set Attribute block updated total moved distance is set to entity. After this step entity spends computed time in Single Server block. After expiration of this time entity comes to another Get Attribute block where number of train and total moved distance are read. The next block is Read Timer block where total time is read. All this information is send through output signal from Station module. From Read Timer block entity comes to Entity Sink block which serves as entity path terminator.

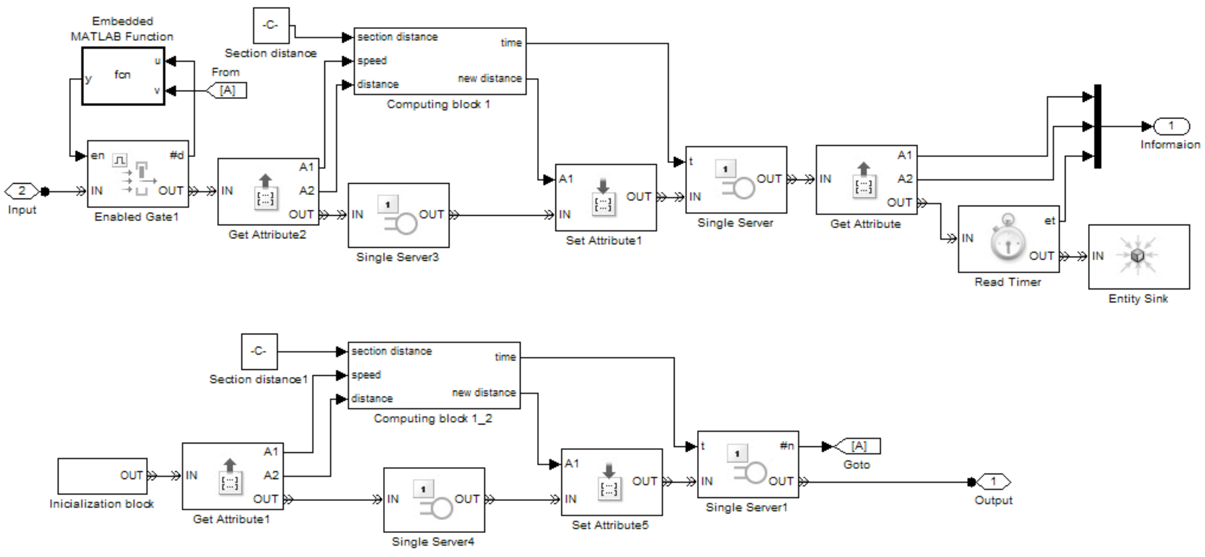

Figure 2: Internal structure of station module

The entity goes through the bottom way in case that given module is starting point for it. After departing of initialization block entity advance into Get Attribute block where speed, distance and distance of given section attributes are read. The distance of given section is obtained from Constant block. Here is its value set as input parameter. All of this attributes are input parameters for Computing module. In Computing module total moved distance is increased by distance of given section and total time needed to pass through this section is computed. In Set Attribute block updated total moved distance is set to entity. After this step entity spends computed time in Single Server block. After expiration of this time entity departs Station module.

Second from main modules is Rail switch (figure 3). Main feature of this module is to route trains to their destination station.

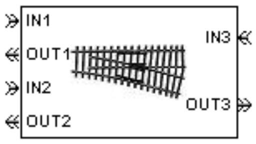

Figure 3: Rail switch module

The internal structure of this module is shown on the figure 4. After entity arrival, attributes train number, count, speed and distance are read in Get Attribute block. In Computing module total moved distance is increased by distance of given section, total time needed to pass through this section is computed and new values of count and next attributes are computed. In Set Attribute block updated total moved distance, new values of count and next attribute are set to entity. After this step entity spends computed time in Single Server block. After expiration of this time entity departs Rail switch module through output specified by value of next attribute.

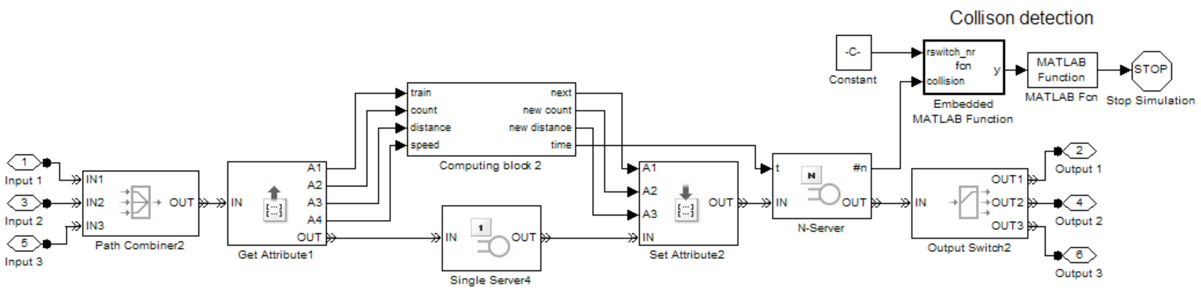

Figure 4: Internal structure of rail switch module

During passing of entity through this module it is checked weather entity is only entity in given module. Number of entities in Single Server block shows how many of them is in current module. In case that number of entities is bigger than 1 collision is detected and simulation ends immediately.

The Single section module is last module from this group (figure 5). This type of module represents section of rails with defined distance.

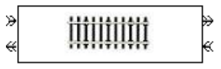

Figure 5: Single section of rails module

Internal structure of this module is shown on the figure 6.

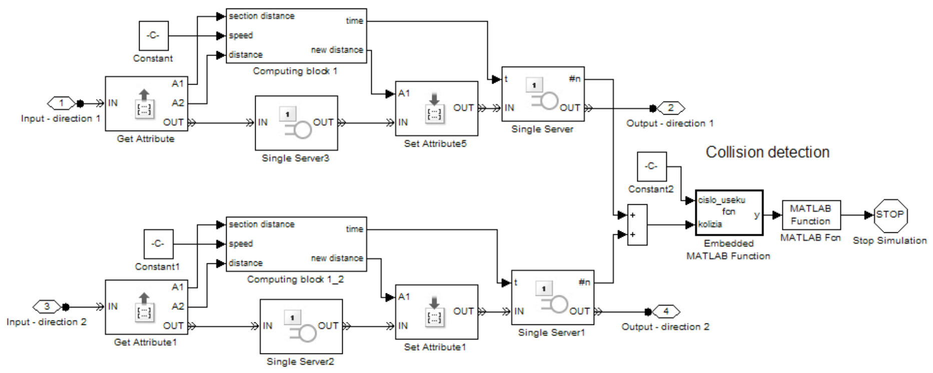

Figure 6: Internal structure of single section module

After entity arrival, attributes speed, section distance and distance are read in Get Attribute block. All of this attributes are input parameters for Computing module. In Computing module total moved distance is increased by distance of given section and total time needed to pass through this section is computed. In Set Attribute block updated total moved distance is set to entity. After this step entity spends computed time in Single Server block. After expiration of this time entity departs Single section module.Collision in this module is controlled the same way as in Rail switch module.

Each main module has two configurable parameters. First of them is distance of current section and the second one is number of this section. The second parameter uniquely identifies current module from others modules of the same type. Unique identification is important to determine in which module collision occurred. Given module is colored red in case of collision and simulation ends immediately.

Special modules

Group of special modules consist of couple of modules. The first of them is Initialization block (figure 7) which is integrated into Station module.

Main features of this module are:

- to generate entity,
- assign initial values to entity,
- start entity timer,
- delay entity start if required.

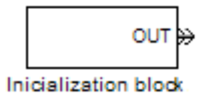

Figure 7: Initialization block

Internal structure of this module (figure 8) consist of three ways. Each of them represents concrete number of train that starts in station where given initialization block is integrated.

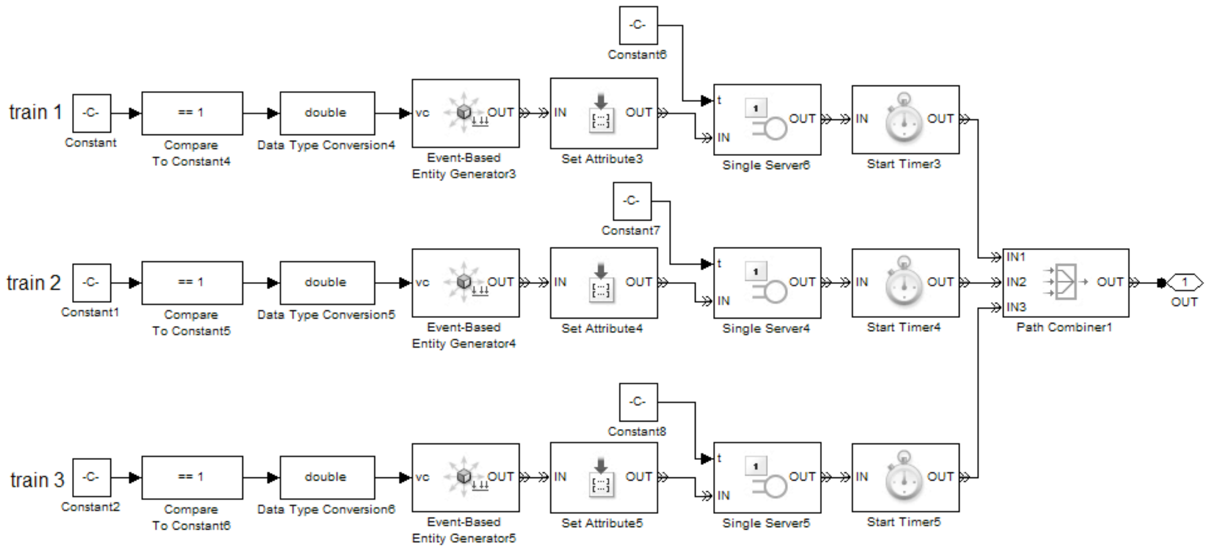

Figure 8: Internal structure of initialization block

Change in input signal of Event-based Entity Generator block cause entity generating. At the beginning of the simulation is value of this signal 0. In case that constant equals to value true, Compare To Content block changes signal value to 1. The constant represents variable which indicates weather trains with certain number starts in given Station. After generating entity goes on to Set Attribute block where initial values of attributes are set. From this point continues to Single Server block where is delayed if required. Default delay value is 0. After this step entity comes to Start Timer block where its timer is started. At the end all the ways are connected into one using Path Combiner block and entity departs initialization module through it.

Following attributes are assigned to entity:
- train number
- count
- speed
- distance

The second module from this group is Counting module. There are two types of computing module. First one (figure 9) is integrated into Station and Single section module.

Main features of this module are:

- update of total moved distance,

- computing of time to pass through given section.

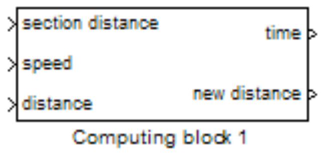

Figure 8: Computing block 1

This type of computing module has three input and two output signal ports. Input signals carry values that represent input parameters for Embedded Matlab Function block integrated in given Computing module. This function block performs mentioned features of Computing module. By analogy, output signals represent output variables of this function.

The second type of Computing module (figure 10) is integrated into Rail switch module. Its main features are same as the first type appended by following:

- determine direction where train will go on from this module – routing,

- increment value of Count attribute.

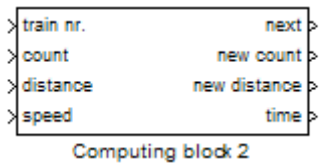

Figure 9: Computing block 2

This type of computing module has four input and four output signal ports. As the first type of this module those signals represents input and output parameters of Embedded Matlab Function block. Function inside this block performs mentioned features.

4 Initialization

Before start of simulation initial values of variables have to be set. This step is important for successful simulation. Initialization is involved by m-file. Consequently input parameters have to be assigned into Matlab command line. Those parameters are number of trains, speed of individual trains, trains start delay, number of start and destination station. According to number of start and destination station track is selected. All tracks are preconfigured in initialization file.

After assembly of rail yard it is necessary to configure initialization file. It has to correspond with newly created model of rail yard. Number of stations has to be set and all tracks the trains can go along.

5 Example of transport system model

The example of rail yard model with six stations as well as way how the collision of trains is signalized is depicted on figure 11.

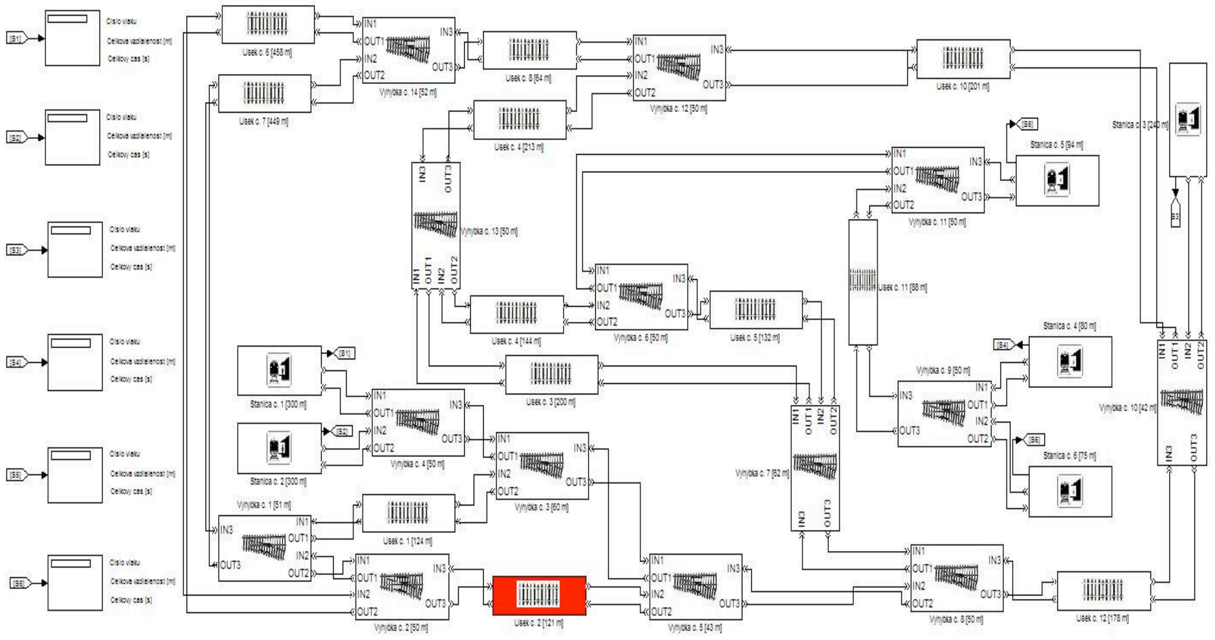

Figure 11: Transport system model

6 Conclusion

Described transport system model demonstrates possibilities to use SimEvents tool for realization transport systems as well as other systems which behavior depends on incurred events.

7 Acknowledgement

This work has been supported by Scientific Grant Agency VEGA 1/0544/09.

References

[1] The MathWorks, Inc., SimEvents – Getting started
[2] The MathWorks. *Getting Started Guide*. 2008
[3] Posterus.sk. Matlab Tutorials. [online] http://www.posterus.sk/?cat=7, 2009

Ing. Kamil Valigura
kamil.valigura@stuba.sk

Ing. Martin Foltin, PhD.
martin.foltin@stuba.sk

Ing. Michal Blaho
michal.blaho@stuba.sk