MODELLING OF MARSHALLING YARD TECHNOLOGY WITH HELP OF PETRI NET

Michal ŽARNAY[1], Karol SLÁDEČEK[2], Petr CENEK[1]

[1]University of Žilina, Faculty of Management Science and Informatics, Slovak Republic

[2]Cleverlance, Prague, Czech Republic

e-mail: michal.zarnay@fri.utc.sk, karol.sladecek@cleverlance.com, petr.cenek@fri.utc.sk

Abstract

Paper presents a model of simple marshalling yard and its technology for handling of trains. The model has been created using the Design/CPN tool package as a timed coloured Petri net with hierarchy.

Keywords**:** *modelling, simulation, coloured Petri net, marshalling yard, railway, queuing system*

1 INTRODUCTION

Aim of this paper is to present a Petri net model of a simple railway marshalling yard and typical technology of train handling used in it. Motivation for building of the model consists in verifying facilities provided by formalism of Petri nets for modelling and analysis of the outlined system. This model should further serve as an environment for experimenting with algorithms for resource management.

The demonstration marshalling yard has been created based on experience with real marshalling yards. Its parameters are fictitious. A definition of technological processes has been made according to technological rules used in existing marshalling yards of Slovak railways. Values of parameters of the modelled yard are of minor importance and can be easily tuned according to user needs.

From a wide scope of existing Petri net formalisms, we have chosen timed hierarchical coloured Petri nets. Model has been created with a help of the Design/CPN tool.

After a short introduction to Petri nets we'll introduce the simple marshalling yard and its technology of trains handling. Finally, design process of the coloured Petri net is briefly described.

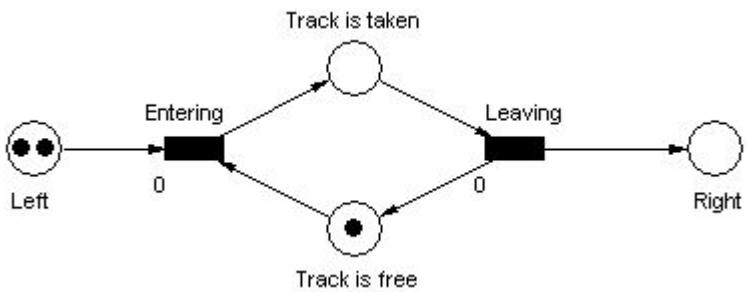
2 SHORT INTRODUCTION TO PETRI NETS

Petri net is a formalism used for modelling and analysis of systems with concurrent processes and their behaviour. As the authors in 5 mention, Petri net can be considered as a model of a system with concurrency that has graphical notation, precise mathematical language and analysis methods for specifying the system behaviour.

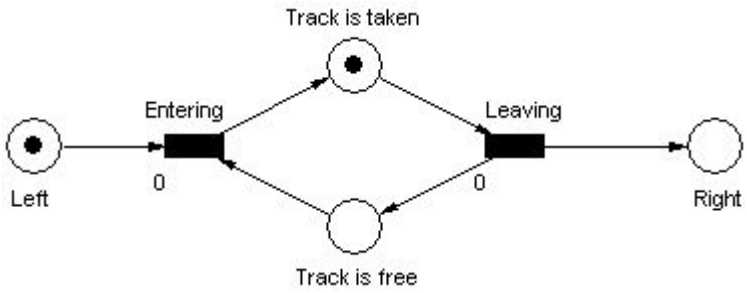
Basic construction elements of Petri net are places, transitions, arcs and tokens. Places and transitions are two types of nodes in the net. Arcs are directed edges that link places with transitions and vice versa, while no pair of nodes of the same type can be connected. Tokens are elements that move in the created network between places through arcs and transitions.

Principal difference between places and transitions lies in their relation to tokens. Places can hold tokens for longer time. Number of tokens and their distribution in places represent state of the net. Transitions take tokens from input places (i.e. place from which there is a directed arc to the transition) and provide tokens to output places (i.e. place to which there is a directed arc from the transition). This process is called firing – it performs an action in the net, changing its state. In this way, it is also possible to change overall number of tokens in the net.

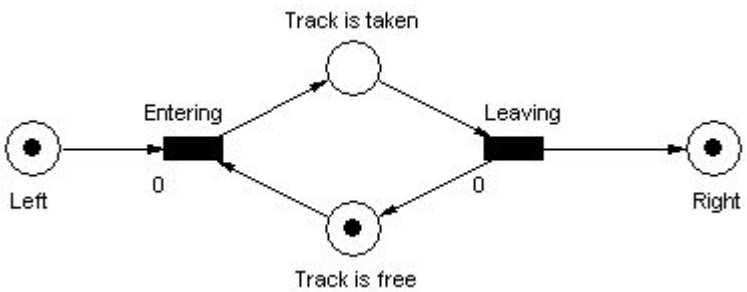
For instance, the sequence on the Figure 1 a-c could depict a model of a railway line section being used by trains moving from left (the *Left* place) through the section border (the *Entering* transition) to the line section, and further through the other section border (the *Leaving* transition) to right side (the *Right* place). The line section is modelled by the two places showing, whether it is free (the *Track is free* place) or taken (the *Track is taken* place). On the first picture, there are 2 trains waiting on the left side before entering. On the second picture, one of the trains entered the line section. On the last picture, the section has been made free from the train that moved to the next section modelled by the *Right* place.



a)



b)



c)

Figure 1 Petri net model of a line section with a train moving from left to right

The Petri net from the example belongs to the basic type called place/transition nets. This concept has been further modified to many different, usually more complicated types. Modifications are determined by attributes of the basic four elements of Petri net. Modification used for this model is timed hierarchical coloured Petri net 55.

In the used modification, time is represented by time stamps assigned to tokens. The time stamps are updated when tokens move along arcs during transitions firing.

Hierarchy is realised by connecting substitution transitions to separated subnets that further specify behaviour of transitions' actions in the system. This allows user to define actions more precisely while keeping the simple overview of the whole Petri net.

Colour is added to the Petri net, when tokens receive further attributes, i.e. they are distinguished by their values (they are not uniform black dots any more). The values can be of different types, simple or complex, like we may know from conventional programming languages: integer number, string, enumerated type, record, list, etc. To be able to manipulate with coloured tokens, also other net elements receive further attributes:

- Places have place types,

- Transitions are controlled by guards and
- Linking of nodes with arcs is specified using arc expressions.

Further details on Petri nets and their classification can be found in 5.

Modelling tool that was used for building of the model is called Design/CPN 5. It is a tool package supporting the use of coloured Petri nets (CPN). It has four integrated parts:

- The CPN Editor is used for construction, modification and syntax check of CPN models.
- The CPN Simulator provides interactive and automatic simulation of CPN models.
- The Occurrence Graph Tool supports construction and analysis of occurrence graphs for CPN models (also known as state spaces or reachability graphs/trees).
- The Performance Tool allows simulation-based performance analysis of CPN models.

We used the CPN Editor for building of our CPN model and CPN Simulator for watching and tuning of the model's behaviour. The other two tools were used for analysis of the created model.

3 SIMPLE MARSHALLING YARD AND ITS TECHNOLOGY

In this section, we'll describe marshalling yard and its technology that was defined for the model.

Marshalling yard contains one hump and three yards: arrival, sorting and departure yards. An example model consists of:

- 8 tracks in arrival yard
- 1 hump track
- 24 tracks in sorting yard
- 6 tracks in departure yard

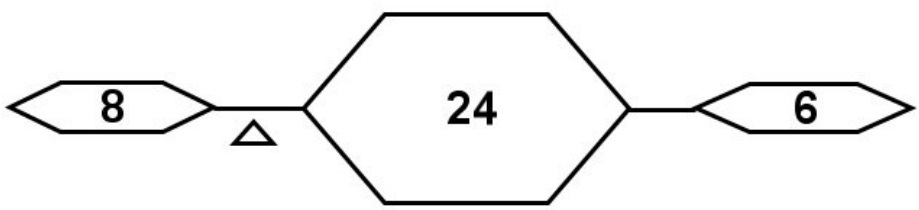


Figure 2 Scheme of modelled yard

There are two engines operating in the yard:

- 1 for humping of train that have arrived
- 1 for shunting of train between sorting and departure yards

Train locomotives arriving with incoming trains and leaving with departing trains stay in a fictitious locomotive depot.

Personnel in the yard are divided into 2 groups. First group looks after technological processes of incoming trains on arrival yard tracks and consists of:

- 6 examiners,
- 6 transiteurs,
- 2 couplers,
- 4 shunters.

Second group is assigned to technology of outgoing trains on sorting and departure yards tracks and consists of:

- 6 examiners,
- 4 transiteurs,
- 3 couplers.

Professions and the number of persons are given for one shift.

Other yard workers that operate in real marshalling yards are not mentioned here, since they are not relevant for the model. The reason is that the model focuses on management of resources that are usually assigned to and released from technological processes on individual trains during their work time. Thus there is no reason to include resources which are not strictly bounded with operation of trains.

Technological processes are defined by flowcharts. There is one technology flowchart for incoming trains (Figure 3) and one for outgoing trains (Figure 4). More details to their construction can be found in 5.

The modelled marshalling yard handles some 920 wagons coming in 36 trains per day.

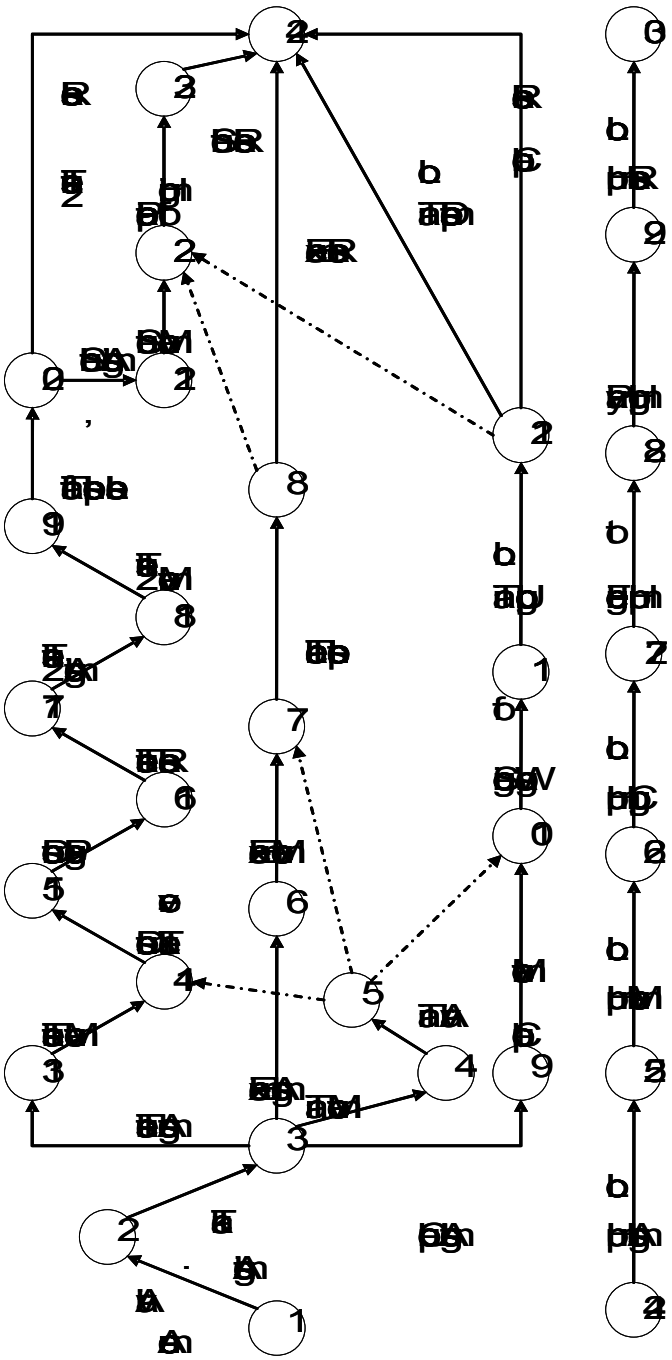


Figure 3 Technology flowchart for incoming trains

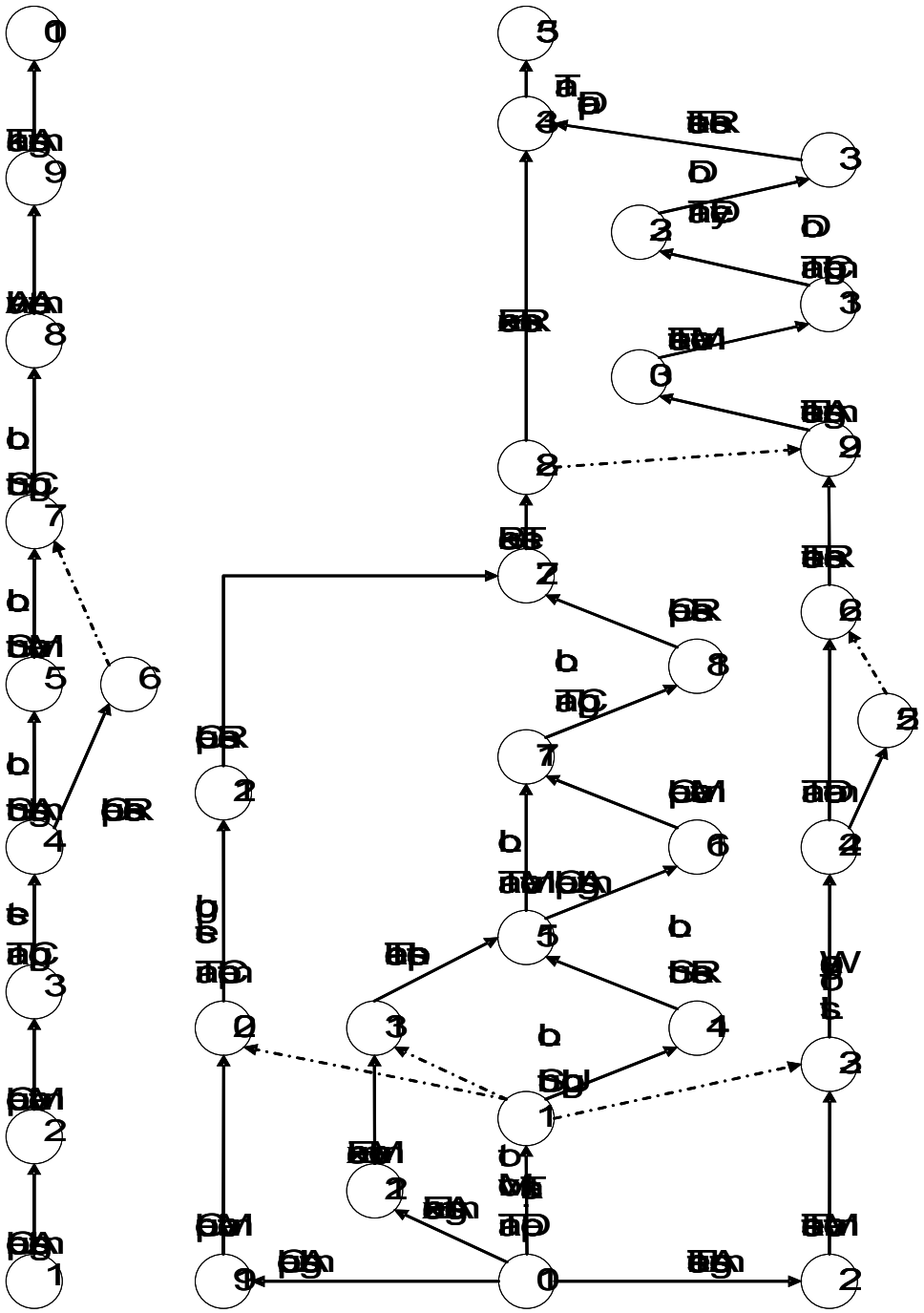


Figure 4 Technology flowchart for outgoing trains

4 DESCRIPTION OF THE MODEL

In this section, we'll describe a model design. Description of all model details would make this paper too long (because of the model's complexity) and understanding them requires additional knowledge about coloured Petri nets. That is why we will mention only most important parts and advise reader to find out other details in 5 or by contacting authors of the paper.

CPN model of the marshalling yard contains one principal net and three subnets. It is hierarchically organised, where subnets represent transitions in the principal net. (Figure 5):

- Generator of trains – built on the *Generator#3* page,
- Arrival technology process – built on the *ArrTrServ#4* page
- Departure technology process – built on the *DepTrServ#5* page

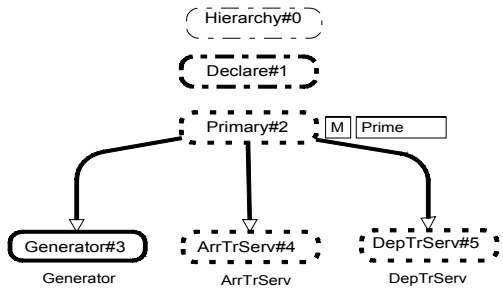


Figure 5 Hierarchy of Petri net model.

The *Hierarchy#0* page shows the hierarchy chart visible at the figure. The *Declare#1* page keeps declarations of colours (types), constants and functions used in the Petri net.

The *Primary#2* page contains the principal net – overview of model with all principal processes in marshalling yard: processing of trains after arrival, sorting of wagons and processing of created outgoing trains with their departure (Figure 6).

The incoming trains are created with help of the *Generator* transition in the top-left corner of the net. After creation they are stored in the *Incoming Trains* place.

If the arrival yard has free capacity, the trains enter the yard and their arrival technology process is started (*ArrTrServ* transition). At the end of this process, wagons are humped. The sorting process is modelled by the *Moving Wagon* and *Wagon Was Humped* places and *Sort* transition. Sorted wagons are stored in the *Relations* place in 24 relations (from A to X).

When number of wagons on any relation track reaches a limit (defined in the *MaxTrainLength* constant), a new train for the relation is initialized through the *Complete* transition. It receives an ID from the *New Train ID* place, moves to the pool of complete trains and its technology process before departure (*DepTrServ* transition) is started. As the latter is finished, the train moves in the *Train Departure* place as

ready for departure. The *Output* transition causes the train to leave the station. All the departed trains are then stored in the last place in the bottom-right corner.

There are a number of places that help in watching the behaviour of the model.

Locomotive depot is modelled by the *Locomotives* place. All engines in the model are present here. Shunting and humping locomotives, when free or occupied, train locomotives only when free, detached from their train. If they are free, the token's 2-tuple has 0 at the second place. If they are occupied, the second value indicates the ID of train that is using the locomotive.

The same approach is used for personnel – both available and occupied employees are stored in two places: the *In Human Pool* and *Out Human Pool* places for personnel for incoming and outgoing train technologies respectively. The tokens for personnel are 2-tuples containing personnel profession in the first place and train ID or 0 number at the second.

Similar way is used for tracks in the respective yards, however only for free tracks. The *In Free Tracks* place holds free tracks for the arrival yard and the *Out Free Tracks* for the departure yard. Occupied tracks are stored in a different place.

Two additional places in the top-right corner store train data during their stay in the station. The *Present Trains On Tracks* place keeps information about current occupied track and length of the train (number of wagons), the *Details Of Trains* place stores wagons data. In this way, flows of data in the Petri net are minimised, because tokens showing processing of technology hold only necessary information about processed train (its ID) and other data about train are accessed from mentioned places only when necessary.

All of the places for resources and data work as fusion places accessible on any page in the Petri net project. It is a similar principle as the access to global variables from various modules in a conventional software project.

A generator of incoming trains is designed in the subnet on the *Generator#3* page (Figure 7). At first, length of new train measured in number of wagons is generated. It uses discrete uniform distribution of probability in the range of 10 to 50 wagons.

It is followed by a generation of wagons, where every wagon gets one destination station address out of 24 (labelled from A, to X). Every address will occur with a uniform probability distribution law.

As the train is completed by receiving its ID (incoming trains in the model receive odd numbers starting from 1), a time period to go before arrival of a new train is generated. This has again a uniform distribution of probability in the range of 5 to 65 minutes.

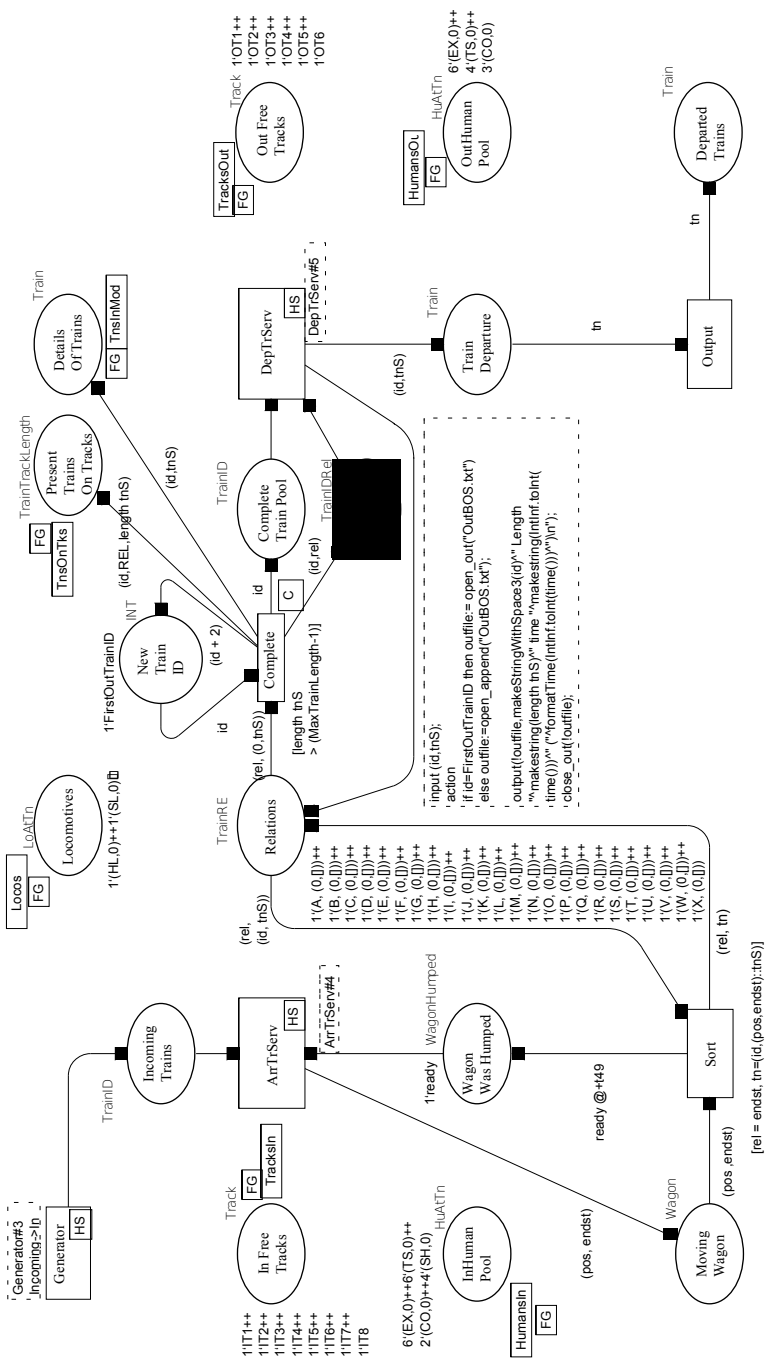


Figure 6 Principal net on primary page.

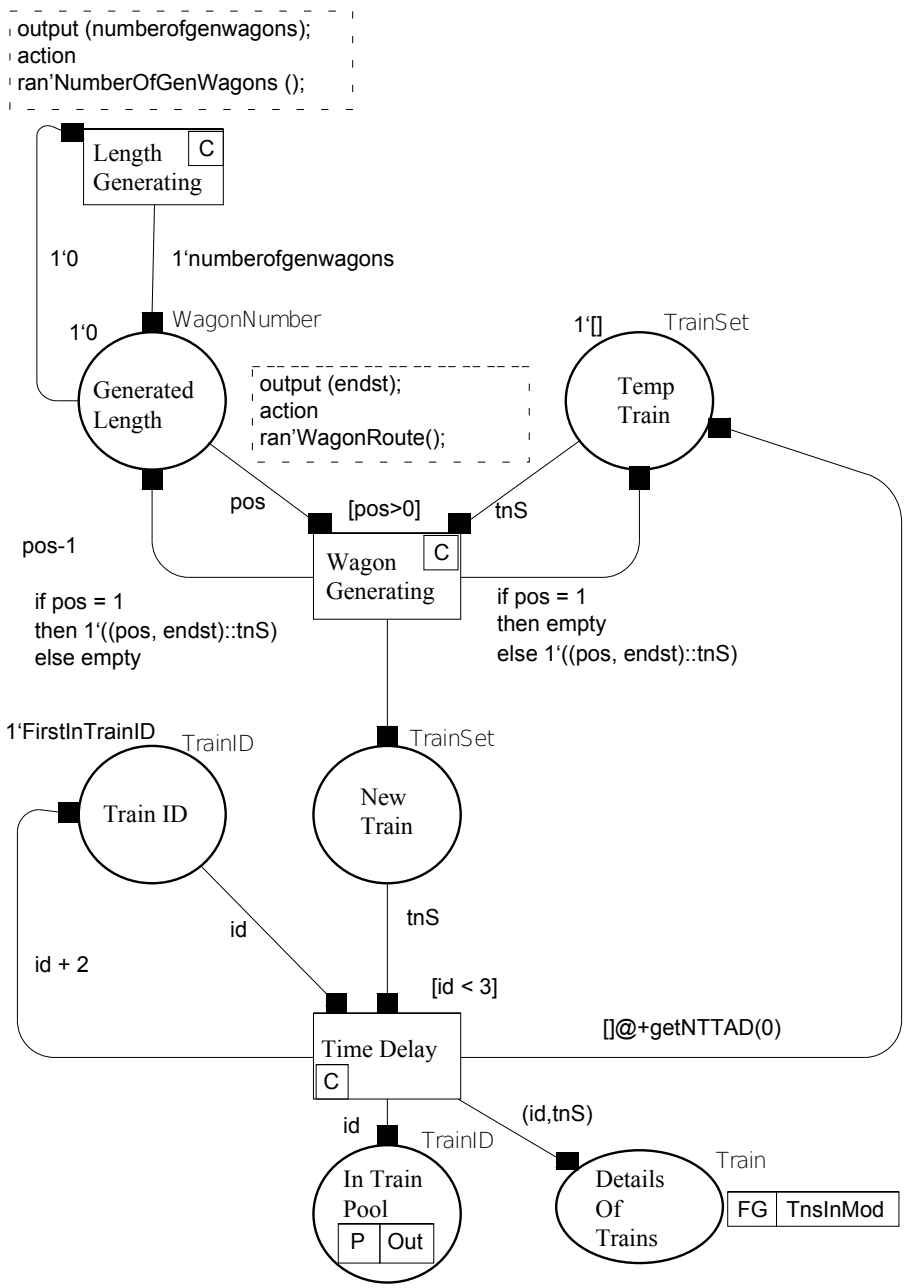


Figure 7 Subnet for generating trains (Generator#3 page).

The *ArrTrServ#4* and *DepTrServ#5* pages contain respective technological processes and they are built based on flowcharts from Figure 3 and Figure 4.

For building the subnets, the following transformation was used: edge of the flowchart is represented by place in the Petri net, and flowchart node is equivalent to Petri net transition. This is valid for the whole flowcharts with few exceptions.

An example of this transformation can be seen on a fragment of outgoing train technology on the Figure 8. The edge names and node numbers correspond to place names and transition numbers.

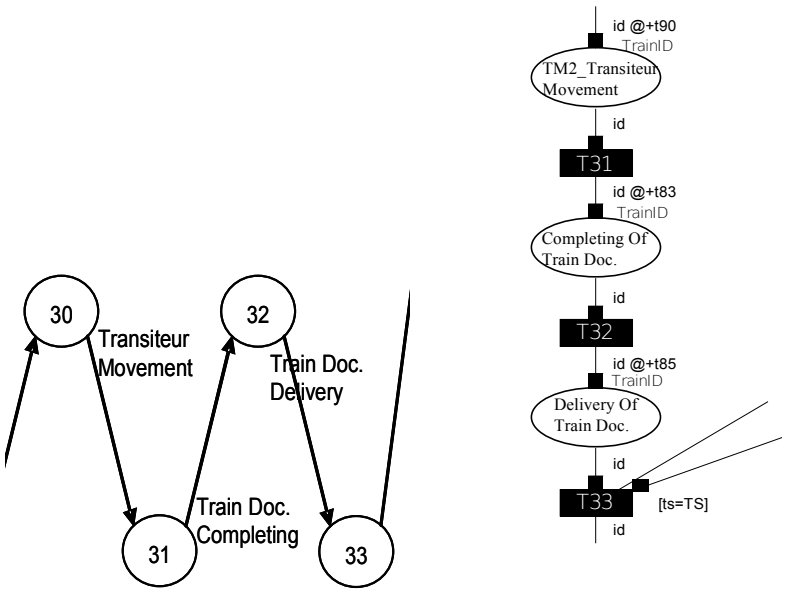


Figure 8 Fragment from outgoing train technology: form of flowchart on the left-hand side, Petri net on the right-hand side.

Tokens showing progress of each technology process contain only train ID and time stamp. All other data used by technological processes are stored in fusion places, in order to minimise data flows in the net, as it was already mentioned in description of the primary page.

Time in the CPN model is represented by:

- Time stamps assigned to tokens,
- Internal counter called clock,
- Conditions of making the timed tokens available.

On the Figure 8, e.g. at the top, reader can see the string: *id @+t90*. This means that the token in *id* will come to the *TM2_Transiteur Movement* place with its time stamp increased by the *t90* constant (what in this case represents 120 seconds for transiteur movement before completing of train's documentation) and it will stay in the

place until the clock will advance to the new time stamp. Only then will the token be available for the *T31* transition to move to the *Completing of Train Documentation* place.

Time stamps are not assigned to all tokens colours used in the model. They are assigned to tokens representing wagons, trains, locomotives and personnel, i.e. entities that are used for analysis of model performance.

Size of the model counts is 72 transitions (3 of them are substituted by subnets) and 137 places including all instances of fusion places and port places (between main net and subnets).

5 CONCLUSION

In this paper, we tried to describe some details about a Petri net model for simple marshalling yard and its technology. Petri net model has been built using the timed coloured Petri net with hierarchy.

The model contains one primary net and three subnets with over 70 transitions and almost twice as many places. It represents a simple marshalling yard with 3 basic track yards and 24 relation tracks.

The created model can be further used for simulation and analysis of the system. Simulation and analysis results will be discussed in another paper, since including them here would make this paper too long.

Modelling of the yard using Petri nets brings benefits as well as shortcomings.

Among the benefits one can count:

- Simple construction elements for building the model without the necessity of writing a long and complex code
- High modelling power of this formalism thanks to colours and time attributes
- Hierarchy that allows using modular approach in model creation
- Available tools for simulation and analysis of created model without a necessity of developing them
- Graphical support that helps easier understanding of the system

As shortcomings, we must mention that for large and complex systems, like marshalling yard is, the Petri net gets quite large – our modelled marshalling yard has been rather simple and yet the Petri net model has over 70 transitions and almost twice as many places.

If the system is too large and too complex, it is possible to focus on some part of it. This can be studied separately and after development, it can be included in the whole system, thanks to hierarchy feature.

The Design/CPN tool has proven to be capable for this type of application. It has integrated useful features, although making modifications and managing sometimes unexpected behaviour of the tool, especially by complex models, is time consuming.

As a replacement, a new tool called CPN Tools is intended 5. It contains major improvements especially in model construction and simulation. However, some useful analysis features of Design/CPN are not available for the CPN Tools yet.

A future development supposes possible moving from Design/CPN tool to CPN Tools, when the new tool will offer the same functionalities as the former one on an acceptable level.

REFERENCES

[1] Desel, J., Juhás, G.: "What is a Petri Net?" Informal answers for the informed reader. In: H. Ehrig et al. (Eds.): Unifying Petri Nets, LNCS 2128, Springer-Verlag Berlin Heidelberg 2001, pp. 1-25, ISBN 3540430679

[2] Jensen, K.: An Introduction to the Practical Use of Coloured Petri Nets, http://www.daimi.au.dk/~kjensen/papers_books/use.pdf or In: Reisig, W., Rozenberg, G. (eds.): Lectures on Petri Nets II: Applications, Lecture Notes in Computer Science vol. 1492, Springer-Verlag 1998, 237-292.

[3] Jensen, K.: An Introduction to the Theoretical Aspects of Coloured Petri Nets, http://www.daimi.au.dk/~kjensen/papers_books/rex.pdf or In: de Bakker, J. W., de Roever W.-P., Rozenberg, G. (eds.): A Decade of Concurrency, Lecture Notes in Computer Science vol. 803, Springer-Verlag 1994, 230-272

[4] Sládeček, K.: Model of Railway Station by Means of Petri Net, Final Paper, University of Žilina, 2004, Žilina (in Slovak)

[5] Žarnay, M.: Use of Petri Net for Modelling of Traffic in Railway Stations, In: Proceedings of international conference Infotrans 2004, February 2004, Pardubice

[6] http://wiki.daimi.au.dk:8000/cpntools/cpntools.wiki - CPN Tools - Computer Tool for Coloured Petri Nets

[7] http://www.daimi.au.dk/designCPN/ - Design/CPN - Computer Tool for Coloured Petri Nets

[8] http://www.informatik.uni-hamburg.de/TGI/PetriNets/ - Petri Nets World: Online Services for the International Petri Nets Community

Acknowledgment

This work has been supported by the institutional grant FRI 3/2004.