Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 190 (2021) 789–796

Procedia Computer Science

www.elsevier.com/locate/procedia

2020 Annual International Conference on Brain-Inspired Cognitive Architectures for Artificial Intelligence: Eleventh Annual Meeting of the BICA Society

Implementation of support for a multi-type flow of requests in the payment card processing server model in the form of a petri net and its validation

B.A.Tshukin[a], V.V.Klimov[a,*], I.D. Sokolov[a]

[a]*National Research Nuclear University MEPhI, Kashirskoe hwy. 31, Moscow 115409, Russian Federation*

Abstract

In this paper we are considering the implementation of support for modeling of the input flow of differently typed requests in the model of the payment card processing system server in form of a hierarchical coloured Petri net. Also, the validation of the extended model was performed using data from a real processing system. Based on its results, we made a conclusion on the acceptability of using the model in further research and in practical application.

Keywords: Processing Server, Processing System, Coloured Petri Net, Multi-type requests, CPN Tools

1. Introduction

In our previous work [1], we have proposed a possible way to validate the model of a highly loaded processing system, which built in terms of a Petri net, and have also conducted its initial testing for the case of simulating single type transactions. As a result, we have found that our approach is quite applicable, and the constructed model simulates the original system with a sufficient degree of accuracy in a given configuration. In practice, however, processing systems rarely interact with only one type of endpoints (terminal devices) and, accordingly, process not only single type transactions. Usually, a processing server deployed in a banking organization works with several types of terminal devices (for example, ATMs from various vendors, POS terminals, Internet payment interfaces). In

* Corresponding author. Tel.: +7-495-788-5699, ext. 8509; fax: +7-499-324-2111.
E-mail address: vvklimov@mephi.ru

10.1016/j.procs.2021.06.091

this regard, it is practical to construct and validate the appropriate model of a processing system which would consider the case of processing the transactions of various types, and that we describe in this paper. First, we outline the features of the construction and structure of the model we use, and then we present the results of its validation process.

Nomenclature

ATM	automated teller machine
POS	point of sale
CPN	coloured Petri net
ML	meta language
Δt	request processing time
n	number of requests

2. Implementation for modeling of multi-type requests

In this paper, the processing server is considered as a set of software processes that process requests to the server in a certain sequence, depending on the type of request. The model of the processing server in this work is implemented in the form of a hierarchical coloured Petri net [2,3] with three levels of hierarchy, based on examples of hierarchical models presented in the documentation for the modeling tool that is used. [4-6]

The uppermost level of the model has no real representation in the modeled system itself and defines the groups of software processes, in which they are classified according to their functional purposes (for ex., "networking processes", "general function processes", "authorization processes"), as well as the interaction (transfer of the processed request) sequence between these groups. The second level of the model, which describes the structure of each processes group, contains models of software processes, and also, by-turn, describes the sequence of interaction between them. The structure of each individual software process is described and its operation is modeled at the third level.

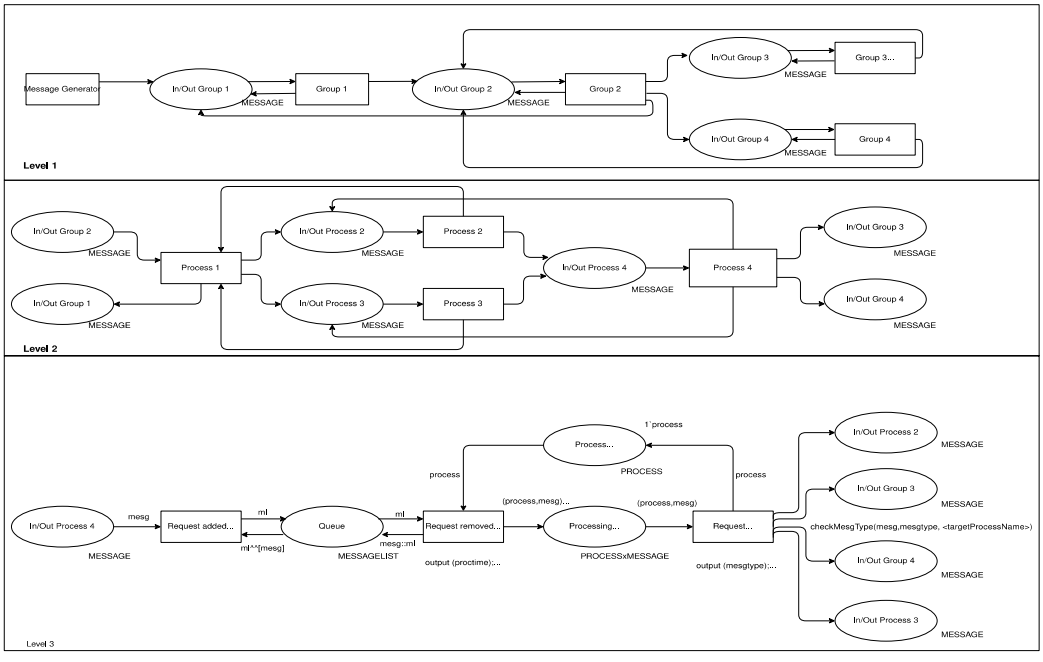

Fig. 1. General view of the processing server model structure.

Visually, for the case of a processing server containing 4 groups of processes and one of these groups containing 4 processes, this three-level structure of the model is shown above, in the Fig. 1.

To simulate the processing of different request types, forking was introduced in this Petri net. As a result, the implementation of token routing was required. The following decisions, based on the capabilities of the CPN Tools - modeling tool used in the study [4], were made to implement the routing:

1. Use of Cartesian product composite type “MESSAGE” tokens.(“type” is defined as “color” in the terminology of coloured Petri nets, however, to simplify perception, here and further in this paper, the definition “type” will be used instead). Each token contains a numerical value, defining the index number of the token, as one of the components of the Cartesian product, and a list of string values, containing the names of the processes of the simulated server in the following form [<process 1>, <process 2>, ..., <process n>], as another component. This list describes the sequence in which the token passes subnets that simulate individual server processes, i.e. it sets token path in the Petri net, which simulates the entire processing server (the first and second levels of the model).
2. To simulate the processing of request flows which differ by request type distribution, the ability to load the process list values for the generated tokens from a text file was added to the subnet, which simulates the source of requests ("Message Generator").

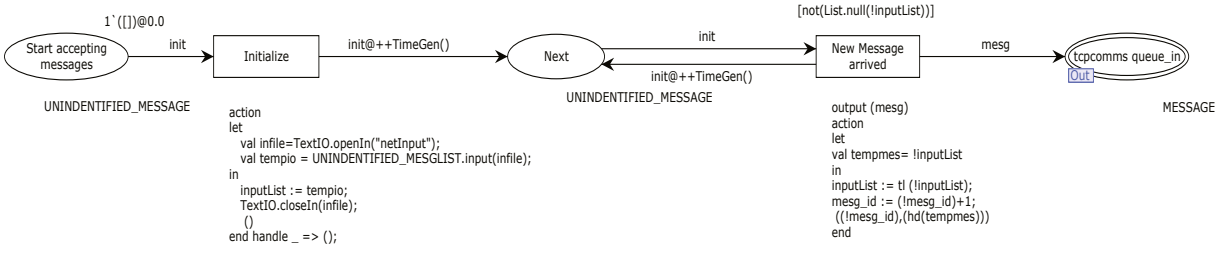

Fig. 2. Petri net for modeling the source of requests to the processing server.

When the “Initialize” transition is executed (Fig. 2), this file is read into the variable named “inputList” of a list type (ultimately it contains a list, each element of which is a list of string values itself). By-turn, when the “New Message arrived” transition, which simulates the event of arrival of a new request for processing by the server, is executed, the next element is extracted from the beginning of the “inputList”. Based on the extracted element, the new token of the “MESSAGE” type is generated. Additionally, tokens of the “UNINDENTIFIED_MESSAGE” type are used in this Petri net. The tokens serve as initializing ones when simulation is started and used to set generation interval for other tokens that model the requests. The generation time interval itself is generated by the “TimeGen()” function.

3. To route tokens between Petri nets that simulate the operation of individual processes, in each such a net, the function “checkMesgType” was added to the output arcs of the transition that corresponds to the event of completion of the request processing by the process of the real-world system. (CPN ML simulation programming language, which based on the Standard ML group of functional languages [7], is used to implement functions and define the logic of the model).The implementation of this function is presented below:

```
fun checkMesgType(mes_id,mes,currentMesgType, mesgType) =
if String.compare(currentMesgType, mesgType) = EQUAL then
   1`(mes_id,(tl(mes)))
else empty
   handle Empty =>  empty;
```

As you can see, this function compares the current value of the next subnet that simulates the next server process, at the beginning of the list inside the current token, with the value specified in the third parameter when the function was called (parameter “mesgType”). The value from the beginning of the token’s list is retrieved during transition execution by the use of the auxiliary “GetMesgType” function. If the compared values match, the

"checkMesgType" function returns the changed token, which has the first element from the internal list removed. If there is a mismatch, no tokens are returned. Also, these implementation features provide protection against token looping within the upper-level net that simulates the entire processing server.

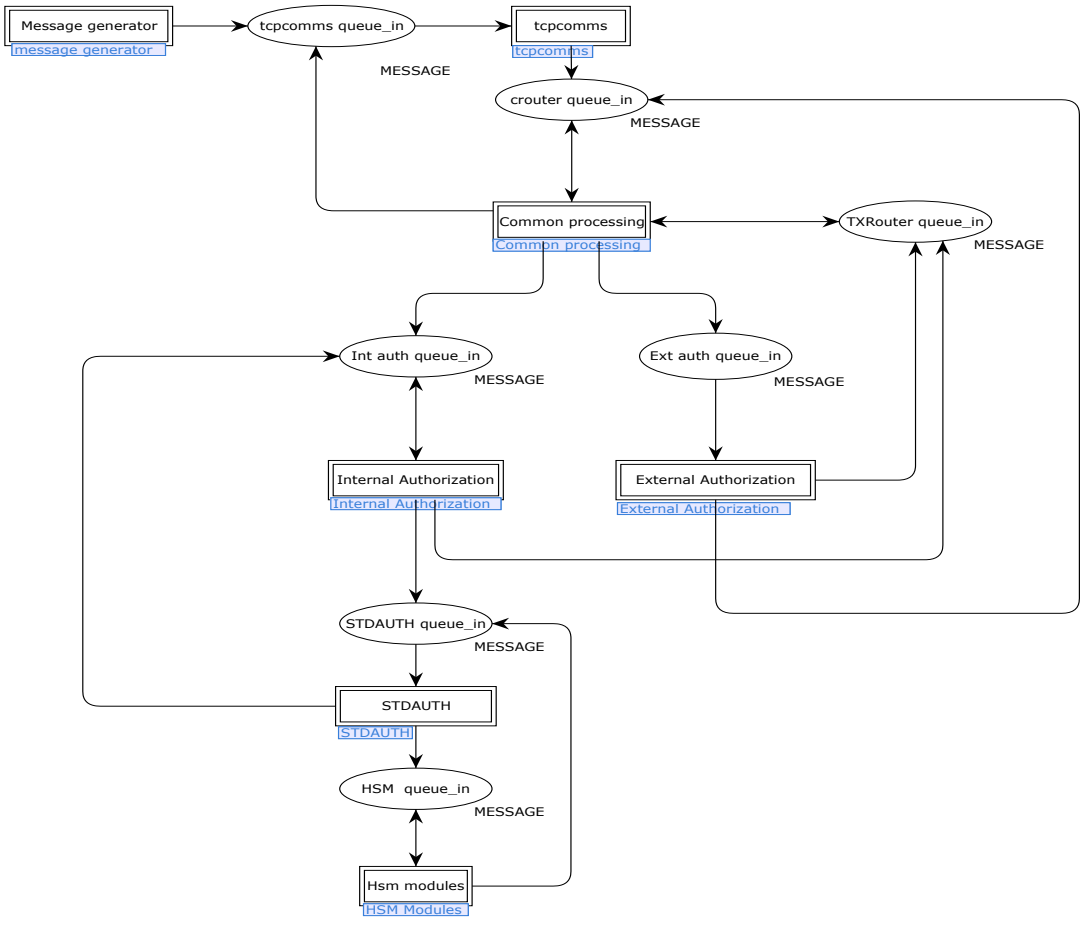

Fig. 3. The upper level of the Petri net that models the processing server [1].

Based on the decisions described above, the second and third levels of the processing server model from an earlier work [1] were implemented; a visual representation of its first level is also shown in Fig. 3.

3. Validation of extended model

In contrast to the model, which was validated earlier, in the current version the following subnets for simulating server processes will be used in the subnets (corresponding to the process groups) while modeling the functioning of the processing server: in the common processing group - ATM_Wincor, ATM_Diebold (for simulating general processing of requests from ATMs that operate on the Wincor Nixdorf, Diebold 912 protocols), 8583POSint (for simulating general processing of requests from POS terminals), EPAYint (simulating general processing of requests from payment or billing systems), Srvgate (simulating general processing of requests from the Internet); in the External Authorization group - NWInt (simulation of external authorization of requests). In this regard, it is initially necessary to validate Petri nets that simulate these processes separately.

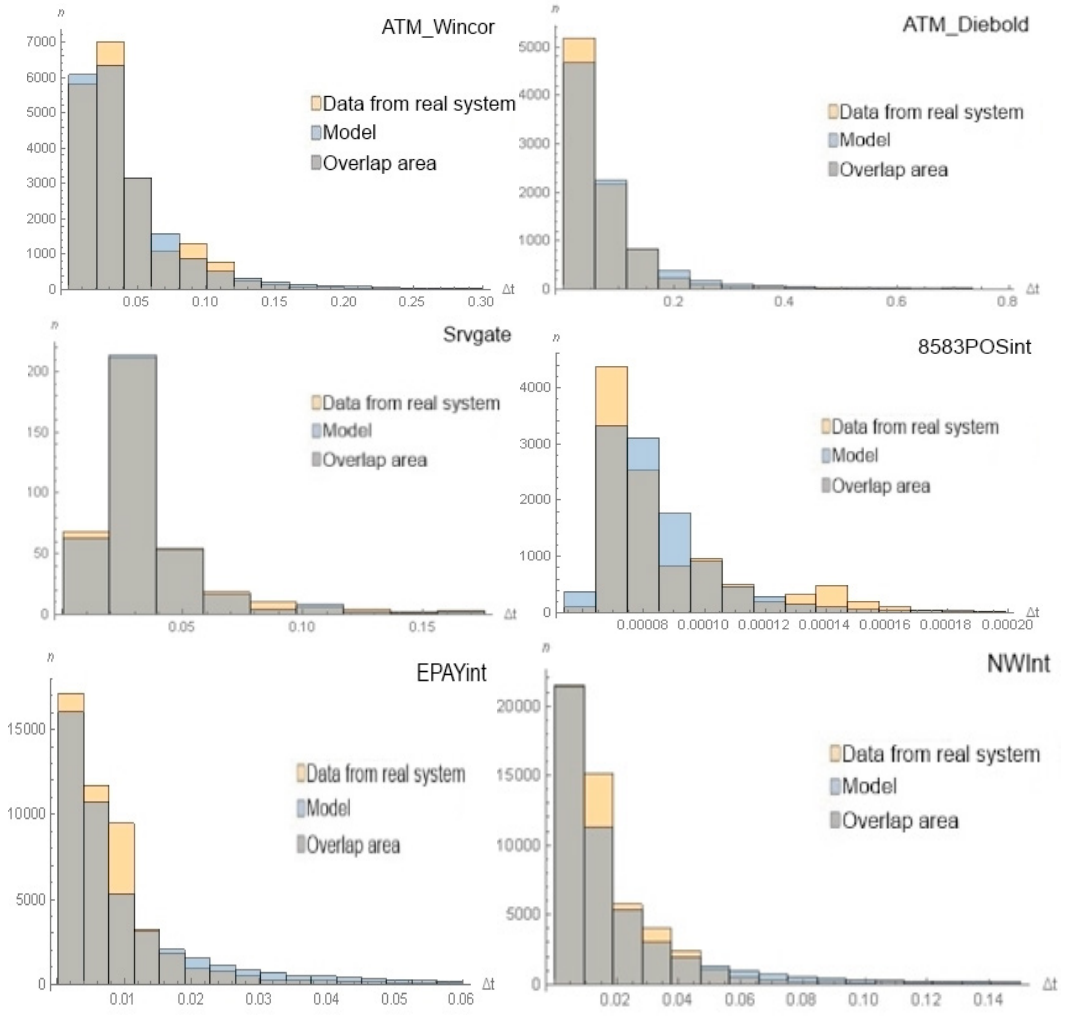

Fig. 4. Comparative diagrams for time delays when requests are processed by actual server processes and their models. (Processing time (Δt) is specified in seconds; n is the number of requests).

As a validation result, a comparison of the output data from subnets with the real data for server processes request processing time is shown in Fig. 4. The Frechet distribution was used to model the delays, that happen during the processing of requests by processes, for all validated subnets; it was determined to be one of the most suitable after analyzing a test sample of data from a real system. The implementation of delays in the Petri net itself, which built in CPN Tools, is performed in the same way as in the previous study [1], with the use of the mathematical expression for obtaining the Frechet distribution from the uniform one, which described in [8]. Based on the validation results, the obtained server's processes models have an acceptable degree of correspondence to reality, which can be also visually seen in the diagrams above.

After the validation of the subnets that simulate individual processes, it is necessary, in accordance with the validation process described in [1], to validate the entire Petri net that simulates the processing server, considering the single-type transaction flows, that pass along the same path between internal subnets. For validation, single-type flows, passing along the following paths, will be used (paths are described as a sequence of level 3 subnets traversed by markers):

Type1: Message generator – tcpcomms – crouter – ATM_Wincor – TxRouter – HostInt – STDAUTH – Racal_Int – STDAUTH – HostInt – TxRouter – ATM_Wincor – crouter – tcpcomms

Type2: Message generator – tcpcomms – crouter – ATM_Diebold – TxRouter – HostInt – STDAUTH – Racal_In – STDAUTH – HostInt – TxRouter – ATM_Diebold – crouter – tcpcomms

Type3: Message generator – tcpcomms – crouter – 8583POSint – TxRouter – HostInt – STDAUTH – Racal_Int – STDAUTH – HostInt – TxRouter – 8583POSint – crouter – tcpcomms

Type4: Message generator – tcpcomms – crouter – EPAYint – TxRouter – HostInt – STDAUTH – Racal_Int – STDAUTH – HostInt – TxRouter – EPAYint – crouter – tcpcomms

Type5: Message generator – tcpcomms – crouter – Srvgate – TxRouter – HostInt – STDAUTH – Racal_Int – STDAUTH – HostInt – TxRouter – Srvgate – crouter – tcpcomms

Type6: Message generator – tcpcomms – crouter – ATMInt – TxRouter –NwInt – crouter – tcpcomms

The following type of requests will not be validated, because it was carried out within the scope of the previous work: Message generator – tcpcomms – crouter – ATMInt – TxRouter – HostInt – STDAUTH – Racal_Int – STDAUTH – HostInt – TxRouter – ATMInt – crouter – tcpcomms [1].

As well as in the early study, an average workload of 1 request per 0.9 s was used while validating on single-type flows. A comparison for values distributions of the total time intervals that are spent on request processing by all processes in the network, between the model and the real system is presented below for the described types of requests:

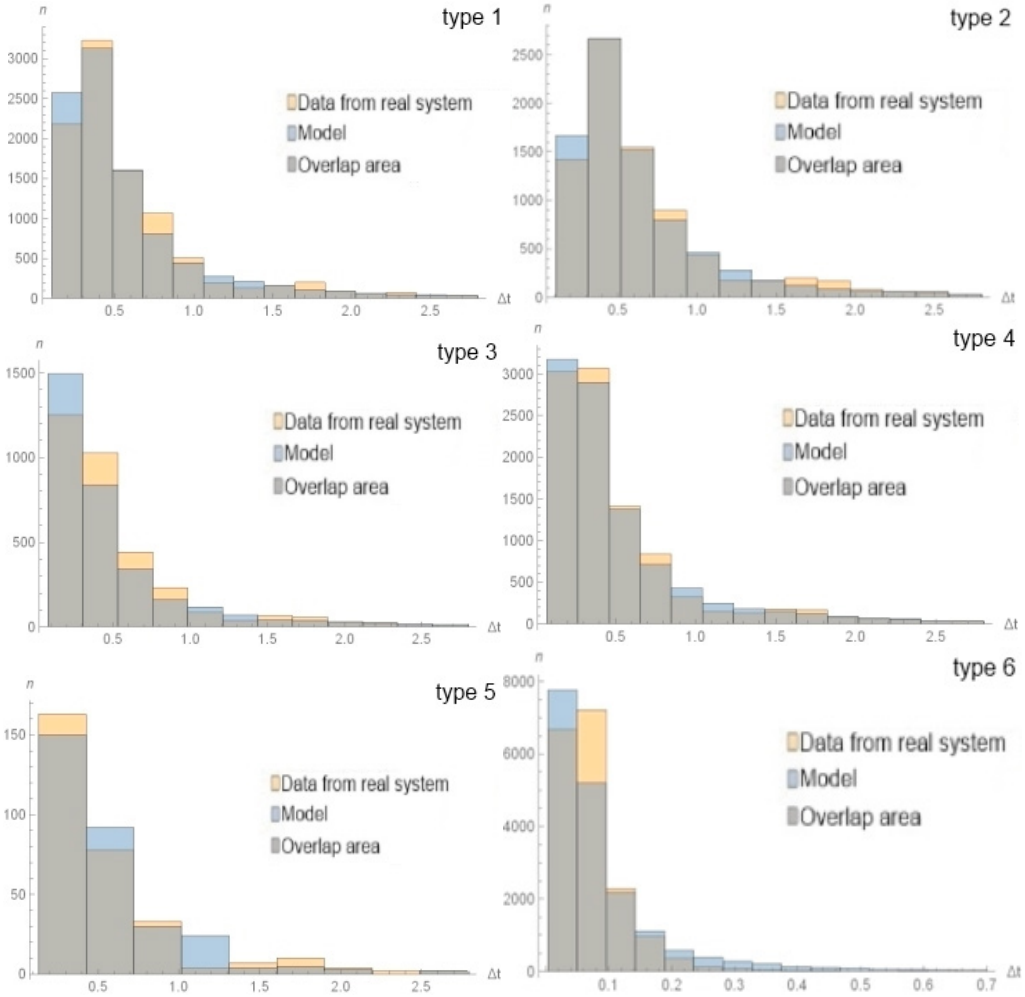

Fig. 5. Comparative diagrams of delays in processing various types of requests by the processing server and its model (processing time (Δt) is specified in seconds; n is the number of requests).

As can be seen in the diagrams, according to the modeling results for the processing of 6 different request types by the processing server, the functioning of the system is quite well reproduced by the model in the case of using single-type request flows. In the next step, the validation of the model with multi-type stream is needed. For a simulated system workload, based on the data from log files, we will use a heterogeneous stream of requests that contains 6 types of requests described above in the following percentage: type 1 - 26.2%; type 2 - 11.1%; type 3 - 4.5%; type 4 - 28.0%; type 5 - 0.4%; type 6 - 29.7%. The number of requests per unit of time, that we use, is the same as when modeling the single-type request flows. The final comparative diagram of the simulation results with real data is presented below:

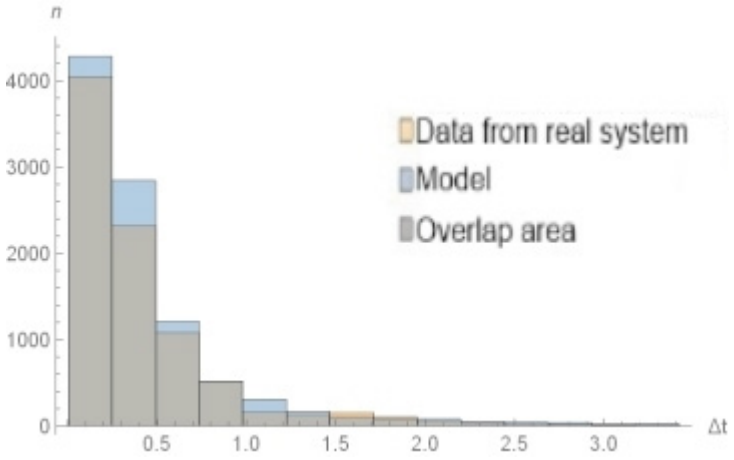

Fig. 6. Comparative diagram of delays in processing a multi-type request flow by the processing server and its model (processing time (Δt) is specified in seconds; n is the number of requests).

It can be seen that for the case of multi-type flow, the model has a good correspondence with the real system as well.

Based on the validation results, the current model can be recognized as suitable for evaluating the performance of the processing server under various workloads basing on the simulation results that are obtained with the dependence on the characteristics of the simulated request input stream.

4. Conclusions

A simulation model of the processing server was built in the form of a hierarchical coloured Petri net, that uses composite type markers, marker routing implementation and supports external file input data loading. These improvements made it possible to implement the ability for simulation of multi-type request flows processing by the system, and simplified the input of simulation parameters. The validation on the resulting model was performed. According to its results, model has showed a good degree of correspondence of the simulation results to real data from the processing server. In the future, it is planned to further improve the algorithms and interfaces for transferring model input/output data, as well as the interfaces for interacting with it.

References

[1] Tschukin Boris Aleksievich, Klimov Valentin Vyacheslavovich, and Sokolov Ilya Dmitrievich. (2017) “Aspects of development and validation of coloured Petri net for modeling of the high-loaded banking processing system server.” *Information-measuring and control systems* **7** (**15**): 37-45.

[2] Jensen, Kurt (1991) “Coloured Petri Nets: A High Level Language for System Design and Analysis”, in Jensen Kurt and Rozenberg Grzegorz (eds) *High-level Petri Nets. Theory and Application*, Heidelberg, Springer

[3] Jensen, Kurt, Huber, Peter, and Shapiro M. Robert. (1991) “Hierarchies in coloured petri nets.”, in Rozenberg Grzegorz (eds) *Advances in Petri Nets 1990. ICATPN 1989. Lecture Notes in Computer Science vol. 483*, Heidelberg, Springer

[4] Westergaard , Michael (2018) “CPN Tools Documentation” http://cpntools.org/category/documentation/, accessed 19 Dec 2020

[5] Jensen, Kurt “Hierarchical Protocol” (2018) http://cpntools.org/wp-content/uploads/2018/01/hierarchicalprotocol.pdf, accessed 19 Dec 2020

[6] Jensen, Kurt “Telephones” (2018) http://cpntools.org/wp-content/uploads/2018/01/telephones.pdf, accessed 19 Dec 2020

[7] Milner, Robin, Tofte, Mads, Harper, Robert, and MacQueen, David. (1997) “Introduction”, in *The Definition of Standard ML (Revised)*, Cambridge, Massachusetts, MIT Press

[8] de Gusmão R.S. Felipe, Ortega M.M. Edwin, and Cordeiro M. Gauss. (2011) “The generalized inverse Weibull distribution.” *Statistical Papers* **52**: 591–619