UDC 004.94: 519.876.5

STETSENKO I.,
DYFUCHYN A.

PETRI-OBJECT SIMULATION: TECHNIQUE AND SOFTWARE

Nowadays information systems tend to be including components for transforming data into information applying modelling and simulation. Combined with real-time data, discrete event simulation could create powerful making decision and control systems. For these purposes, simulation software should be concentrated on creation the model as a code which can be easy integrated with other components of software.

In this regard, Petri-object simulation technique, the main concept of which is to compose the code of model of complicated discrete event system in a fast and flexible way, simultaneously providing fast running the simulation, is requisite. The behaviour description of the model based on stochastic multichannel Petri net while the model composition is grounded on object-oriented technology.

The Petri-object simulation software provides scalable simulation algorithm, graphical editor, correct transformation graphical images into model, correct simulation results. Graphical editor helps to cope with error-prone process of linking elements with each other.

For better understanding the technique, the Petri-object model of web information system has been developed. Investigation of the response time has been conducted. The experiment has revealed system parameters impact on the value of response time. Thus, the model can be useful to avoid long-running request.

Keywords: stochastic Petri net, discrete event simulation, web information system, long-running request, response time.

1. Introduction

Despite a long history of information systems, the vast majority of systems developed stay ‘data rich, information poor’. Huge storage of data can help to find a solution to problems in organizations and business. However, it is often high-effort and time-consuming process for each specific problem. To be useful data should be transformed into information, and the automatization of this process should be advanced considerably. IBM Center for The Business of Government in [1] highlights trends in the information system development, among which providing near-real-time data available to wide range of users and simplifying an interpretation of data to make information more useful.

Discrete event simulation is a powerful technique of the investigation of system behaviour conducting experiments with the model of the system (instead of the real system) that intended to improve the functioning of complicated systems (increasing performance or decreasing risks). Combined with real-time data, simulation components can create a powerful making decision and control systems. However, this requires specific demands to the simulation model: the code of the model should be implemented as a part of software, the model should be able to integrate with data storage and with end-user interface, the model should be able to be run by other components of software (not scientist). As modern simulation tools do not support these requirements, the development of relevant simulation tool is essential.

Particularly, discrete event simulation is useful for information system designing. Performance time, for example, can be estimated. In addition, the impact of parallel and concurrent processes on performance can be researched.

Systems and software engineering standard ISO/IEC 15909-1:2004 highlights following advantages of using Petri in programming systems and software designing: mathematical formalism of Petri net provides unambiguous specifications and descriptions of applications, graphical representation allows visualizing resource flow and control flow to better understanding system behavior, executable technique allows to verify ideas at the earliest and cheapest opportunity [2]. For today, the software implementing the use of Petri net to create relevant models of programming systems does not developed.

This research considers the advantages of Petri-object simulation technique to create models of complicated system. The usage of Petri-object simulation software to simulate information system behaviour is represented.

2. Related works

Discrete event simulation fundamentals are represented in book [3], where the best description of simulation algorithm and the fulfill information about random values generators can be found. However, there is no word about applying Petri nets for the purposes of simulation.

Stochastic Petri net is a widely known mathematical formalism of discrete event processes that provides system behaviour simulation. This type of Petri net is associated with transitions, the time delay of which are determined only by exponentially distributed random variables [4]. Later interpretation of this type includes time delay defined as random value with given probability distribution or zero value [5].

Coloured Petri net is an extension of stochastic Petri net in which simultaneous processing of different types of markers is possible. The mathematical basics of coloured Petri net are presented in book [6]. The number of elements needed for system simulation is significantly less in colored Petri net. However, the firing rule becomes much more complicated. In addition, the model representation has lost its simplicity compared with stochastic Petri net.

One of the latest extensions of Petri net is considered in work [7]. In work [8] the importance of usage of information system simulation to investigate the failure scenarios have been highlighted.

The authors of work [9] considered the development of the software for simulating information systems. The modeling language for information system was proposed as an extension of PNML (Petri Net Markup Language). The description of the model is divided on information model and process model, which manipulates with information model.

3. The purpose and objectives of the study

The aim of this research is to discuss the advantages of Petri-object simulation technique, in particular, in context of information system development. The end objective of the research is to apply the Petri-object software to build an information system model and to obtain a simulation result that is valuable for information system design.

4. Materials and methods

4.1. Petri-object simulation basics

Petri-object simulation basics were stated for the first time in work [10]. During the following years, the software was developed and the experimental research was conducted. In addition, several models were built and investigated using the new technique. The new technique implementation in different fields has revealed its advantages and limitations. Thus, the technique has been improved.

The concept of Petri-object model is grounded in object-oriented technology and at the same time is based on stochastic Petri net. Object-oriented technology states the rule in which elements of the model are created and combined. Stochastic Petri net establishes activities which are being performed by the elements of the model. Provided a specified way of connections between elements, the operation (or behaviour?) of the entire model will be determined by the stochastic Petri net obtained by the union of Petri nets of its elements.

Modeller determines the classes of elements, creates objects of these classes, and the connections between objects. After that, the model is able to simulate the behaviour of a system under investigation. Instead of routine creating a huge number of events the classes of typical model elements should be defined with once determined Petri net in each of them. This makes the construction of the model more convenient, understandable and faster.

Definition 1. Class *PetriSim* is a type of object that specifies the field describing Petri net construction and methods implementing transformations of the net according to the rules of stochastic multichannel Petri net, detailed mathematical description of which has been represented in work [11].

The objects, created by the constructors of the class *PetriSim*, generate a set of objects Θ using data set, corresponding to constructor arguments. The simplest constructor contains the value in which Petri net is determined.

Definition 2. Petri-object is an object of the class *PetriSim* or its descendants (or an object of the type *PetriSim*):

$$O = PetriSim(a) \tag{1}$$

where O is Petri-object, PetriSim(a) is a constructor of the class, a is a set of values of constructor arguments among which there is the Petri net description.

To link Petri-objects, two types of connection are provided. The first one is to use shared places. It is defined to link pairs of objects by determining one or more places shared (sometimes called places fusion). The second one is to use event initialization. It is defined to link an object with a group of objects by determining additional output arcs for the transition of the object which are directed to the places belonging to the group of objects. Thus, two types of connections could be regarded as 'one-to-one' and 'one-to-many' connections.

Definition 3. Petri-object model is the model composed of the connected Petri-objects:

$$\text{Model} = \bigcup_j O_j, \qquad O_j = \text{PetriSim}(a_j), \tag{2}$$

where $O_j \in \Theta$ is an object of the type *PetriSim*, a_j is a set of arguments including Petri net.

In terms of UML, the structure of Petri-object model is the aggregation of classes that inherits *PetriSim* class or the class *PetriSim*:

$$Model \diamond\!\!- (\{C_k, k = 1, \dots L\} \lhd\!\!- PetriSim) \vee PetriSim), \tag{3}$$

where $\diamond\!\!-$ denotes aggregation, $\lhd\!\!-$ denotes inheritance, C_k is a class of Petri-objects, L is the number of classes.

In work [Stetsenko, 2011], the following fundamental statements were proved.

Statement 1. The Petri-object model functioning is described by the stochastic Petri net obtained by the union of all nets of Petri-objects the model has been composed:

$$N = \bigcup_j \widetilde{N}_j \tag{4}$$

where the union of nets means the union of places, transitions and arcs sets of these nets, $\widetilde{N}_j$ is the stochastic multichannel Petri net of Petri-object O_j united with additional output arcs in case if 'one-to-many' connection has been used to connect the object. In case of using only 'one-to-one' connection $\widetilde{N}_j = N_j$.

Statement 2. The transformation of the net of Petri-object model is divided on transformations of the nets of Petri-objects the model has been composed:

$$D^-(\boldsymbol{S}) = \{D^-(\tilde{\boldsymbol{S}}_j), j = 1, \dots q\}, \qquad D^+(\boldsymbol{S}) = \{D^+(\tilde{\boldsymbol{S}}_j), j = 1, .. q\}, \tag{5}$$

where $\boldsymbol{S}$ is the vector of model state, containing the state of places $\boldsymbol{M}$ and the state of transitions $\boldsymbol{E}$, D^- and D^+ are the transformations corresponding to tokens input and output in Petri net, that are described by logic-algebraic equations.

Statement 3. The Petri-object model simulation is determined by the state equations, one of which determines the moving of the current time to the nearest event and the other determines the transformation of the state of the Petri net elements:

$$t_n = \min \boldsymbol{E}(t_{n-1}), \quad \boldsymbol{S}(t_n) = (D^-)^m\left(D^+(\boldsymbol{S}(t_{n-1}))\right) \tag{6}$$

where t_n is the current time, $t_n \geq t_{n-1}$, $\boldsymbol{E}(t_{n-1})$ is the state of transitions in the previous moment of time, m is the number of repeated tokens input needed to achieve the state of the net in which all transitions are not firing.

The process of construction of Petri-object model, divided on four steps, is represented in Fig.1 on the widely known model of 'dining philosophers', that demonstrates the using of shared resources by synchronized processes. The model is composed of objects, each of which simulates the behaviour of philosopher trying to catch two sticks to start to eat. Because the total number of sticks is equal to the number of philosophers, one part of philosophers will wait for 'resources' while another part of philosophers will 'process'. In Fig.1, at start moment each philosopher has one stick. Shared places marked with gray color implement the links between objects.

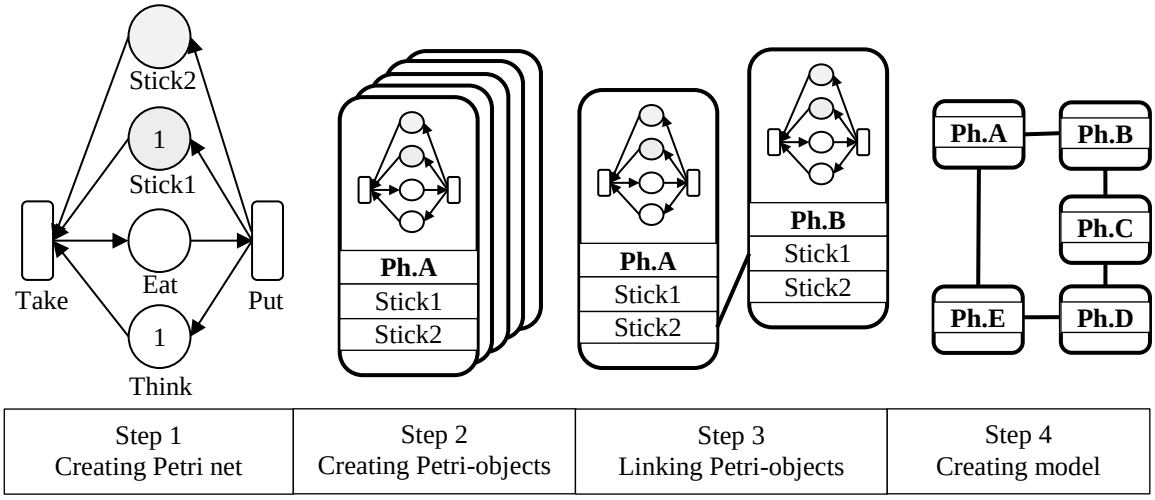

Fig. 1. The dining philosophers model.

4.2. Comparison with known Petri nets

Stochastic Petri net is a very convenient tool to investigate discrete event dynamic systems. However, its modelling power decreases if the number of events significantly increases. Multiple arcs create a confused and incomprehensible net which is difficult to edit and modify. That is why many extensions were developed to overcome this drawback: object-oriented Petri net (place, transition and net is able to be an object) [12], nets within nets (a place is able to contain a net) [13], hierarchical Petri net (a transition is able to contain a net) [14], colored Petri net (use types for markers) [15].

Coloured Petri net was developed to simplify (or reduce) the description of a complicated system which has a lot of elements. Petri-object model has the same aim. CPN Tool combine coloured Petri net with ML programming language while Petri-object simulation combine Petri net with object-oriented programming.

Comparing the representation of the model in Fig.1 with the representation of the same model by colored Petri net [16], it should be noted that the links between elements are created in Petri-object model. It makes the perception of the model structure clearer. Coloured Petri net hides a lot of information about interaction between elements into expressions on arcs and on value declarations that makes model less readable.

Stochastic Petri net does not provide replication of subnets or save information about markers. It does not allow zero value for a time delay or priority value for transition. Colored stochastic Petri net supports subnet replication but does not support subnet replication with given parameters. It provides links between subnets but only by one pair of share places. Simultaneously Petri-object model allows

using as much as possible links between objects. In addition, it allows linking between one object and a group of objects.

In addition, object-oriented structure of Petri-object model has multiple advantages. Creating class inherited PetriSim class the user is able to add fields and methods which are specific for the problem under consideration. For example, the methods gathering the information about simulation results are often useful.

4.3. Petri-object model simulation algorithm

Simulation algorithm of Petri-object model should implement the state equation (6). The repeated actions of time moving and transformation the state of Petri net are performed until the simulation time is achieved. Due to the model transformation is divided on the transformations of the Petri-object nets, the complexity of the simulation algorithm of the Petri-object model is significantly less than the simulation algorithm of stochastic Petri net.

5. Results

5.1 Petri-object model simulation software

Petri-object simulation software is developed using Java language [17]. It consists of package `PetriObjLib` implementing Petri-object simulation algorithm and the packages providing graphical presentation of nets. The graphical editor helps to build Petri net, save it as a Java method and add method to the `NetLibrary` class. The opening net from file or reproducing net from Java method are supported by the software. In addition, animation of simulation is provided to check the rightness of created Petri net. It should be noted, exception will be generated if Petri net consists a transition which has not input or output places. After successful saving, the method can be used to create Petri-objects. When the list of Petri-objects is prepared and the links between objects are determined, the model can be created using `PetriObjModel` class. The method `go(double time)` of this class run the simulation. Exception will be generated if a time delay generator will produce a negative value.

Software main responsibilities are to provide correct simulation algorithm and correct simulation results including mean values of markers in Petri net places, mean value of buffers in transitions and the state of Petri net in the last moment of simulation.

5.2 Web information system Petri-object model creation and simulation.

To consider the example of simulation, the model of web information system based on REST architectural constraints, has been chosen. An ordinary client-server architecture based system includes a client-side application for users and a server-side application for handling user's requests. There are two types of users: authors and guests. The basic flow for authors consists of following actions: 'create', 'update', 'index', 'show', 'delete'. Guests have only two actions: 'index' and 'show'. Each action is implemented by corresponding request which needs to be handled by the server-side application. Server handles request in concurrent way by processing many requests simultaneously. Request processing consist of following operations: processing data (retrieving data from database, mapping, sorting, calculating, etc.) and rendering response. The common problem of information systems is long-running request, which is influenced by many factors such as number of users, number of simultaneously processed request, data processing algorithms, etc. The server load directly affects the response time which is very critical value for user satisfaction and need to be optimized. It can be a quite complicated task to realize what exactly affects the response time, especially in case of real-world large information systems. With the help of simulation technique, the model can be construct and investigated by making experiments.

Utilizing Petri-object model approach the following classes of objects should be defined: Author, Guest, Server. Corresponding Petri nets are represented in Fig. 2. Share places are used to link objects. The model composing of linked objects is represented in Fig.3.

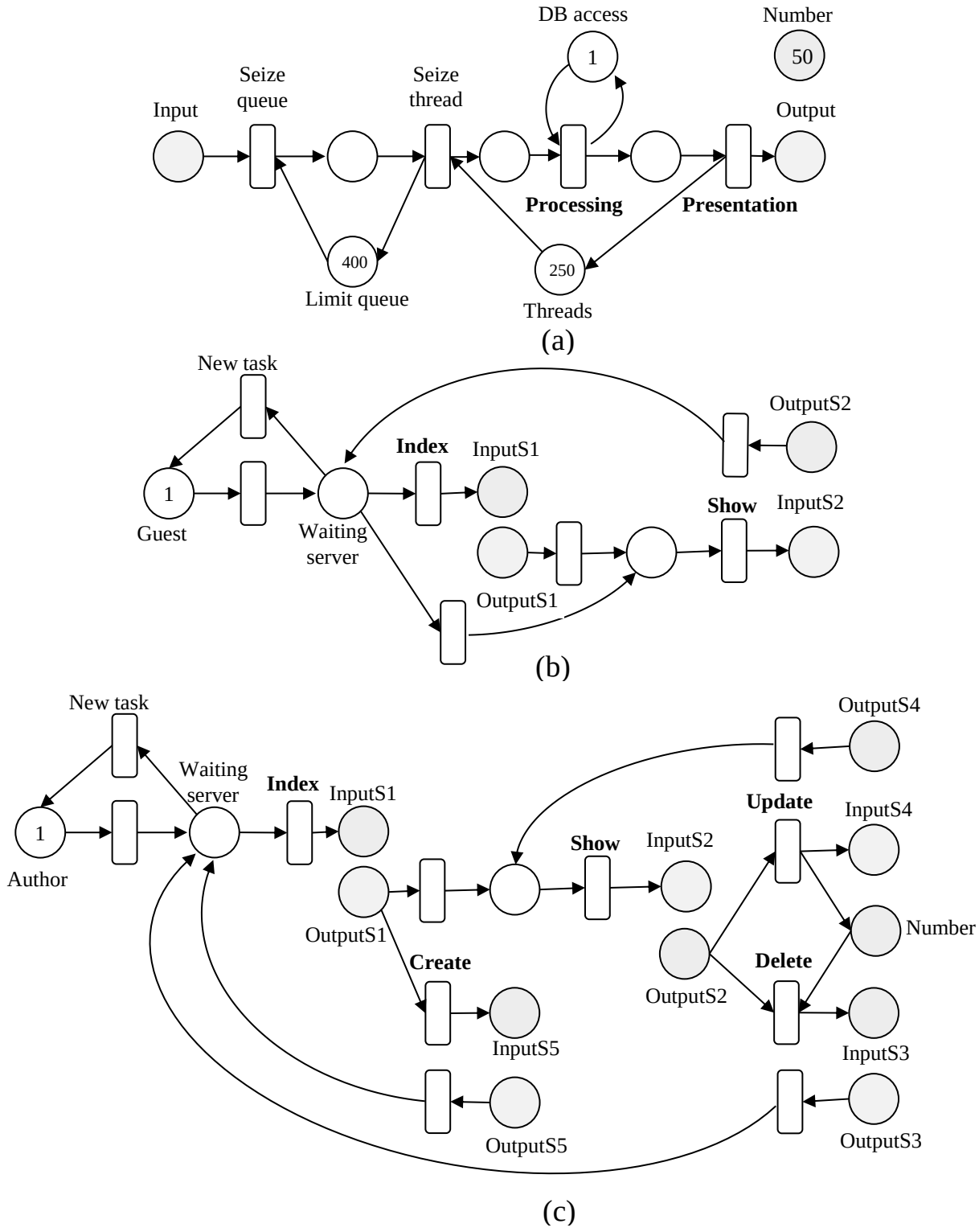

Fig. 2. The nets of Petri-objects Server (a), Guest (b), Author (c).

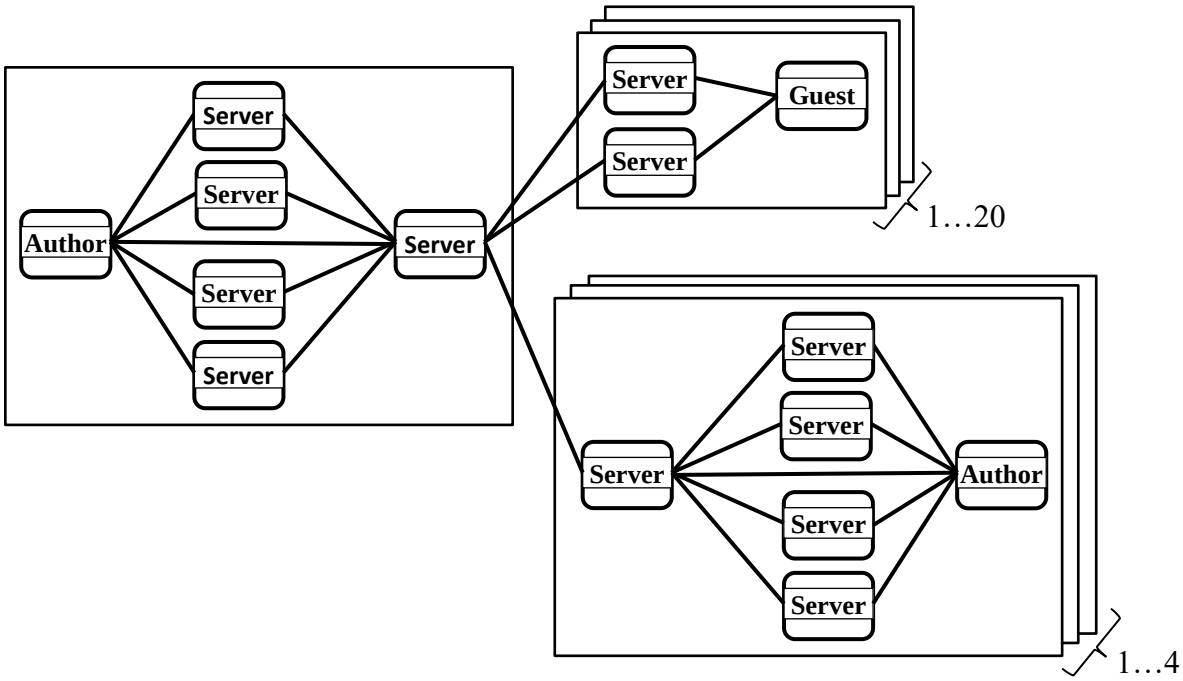

Fig. 3. Composing the information system model of Petri-objects Author, Guest, Server.

The experimental investigation of the information system parameters impact on response time is represented in Fig. 4. In the first experiment the ratio of rendering time to processing time was 10^6. In the second experiment the same ratio was decreased to 10^5. If the rendering time is much more than the processing time, the increasing of the number of threads helps to decrease significantly the response time until the quantity of threads becomes sufficient for the requests flow. Further increasing the quantity of threads leads to unproductive use of server resources. Reduction of the rendering time cause changes in system behavior. In this case the processing time is the main reason for occurring a long-running request because the presentation is fast enough. The increasing of the number of threads leads to increasing the response time because more requests are able to start processing and, as a sequence, the waiting time in the data base queue is grow. When the number of threads is large enough for requests flow, there is not further increasing of response time and it is an unproductive use of resources.

Thus, based on the simulation results, the decision can be made for each set of system parameters. For example, in case of a large number of users with author rights a shorter response time can be achieved if the ratio of rendering time to processing time is large enough and the number of available threads is sufficient. Conversely, when information system is used by a small number of users, a shorter response time can be achieved is the ratio of rendering time to processing time is not so big.

5.3 Algorithm complexity investigation

In case of models with hundreds of elements and connections, traditional simulation techniques cannot be applied because of fast growing computational complexity and, as a sequence, the performance time. However, Petri-object simulation algorithm complexity is polynomial. The experiment conducted on the model of information system confirms this fact (Fig.5).

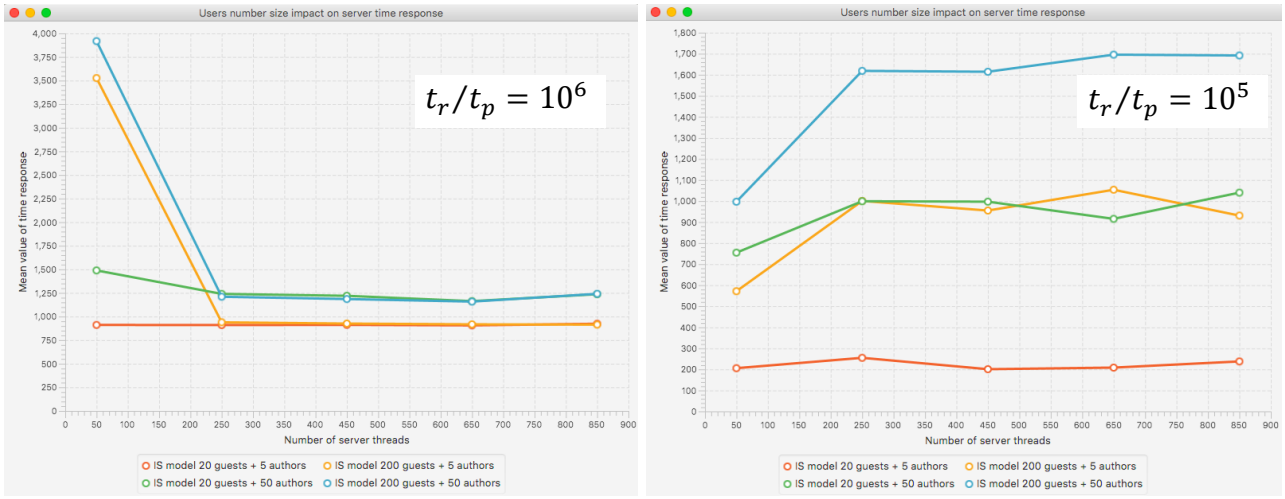

Fig. 4. The information system parameters (the number of users, the ratio of rendering time tr to processing time tp, the number of threads) impact on the response time.

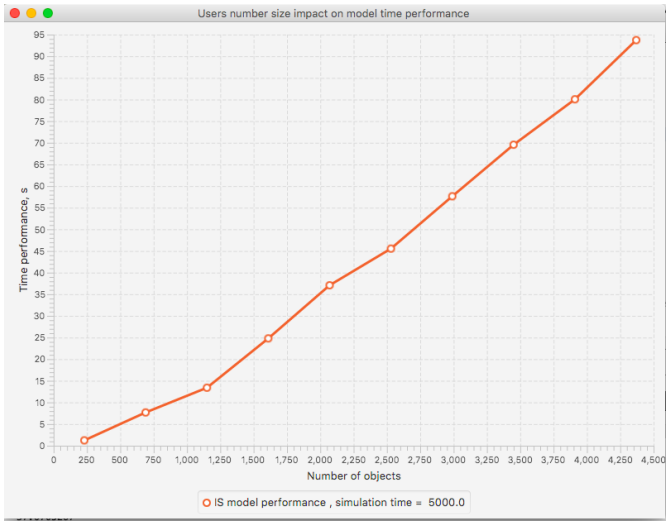

Fig. 5. The model performance time experimental investigation.

6. Conclusions

Thus, Petri-object simulation is a powerful technique to create simulation models of complicated discrete event systems. It provides fast replication the nets of typical model elements with given parameters, construction the model in a way that its behavior is described by the Petri net too, scalable simulation algorithm due to dividing transformation of the model net on transformation of the model elements net. The nets, building with graphical editor, Petri-objects, and the whole model are saved as a Java code that provides the model fast integration with other components of software and its flexible usage.

In particular, the model of web information system, based on REST architectural constraints, is developed by means of Petri-object simulation. Simulation results show the system parameters impact on response time. For example, the large number of threads could not decrease the response time (and avoid long-running request) if the ratio of the rendering time to processing time of data is not sufficiently large. Thus, simulation is able to reveal the drawbacks in information system design.

Further investigation aims to extend Petri-object simulation technology by adding nested Petri-objects which will help to simulate more complicated systems.

References

1. IBM Center for The Business of Government. Kamensky, J.: Data rich, but information poor. (2018). http://www.businessofgovernment.org/blog/data-rich-information-poor, last accessed 2020/08/28.
2. ISO/IEC 15909-1:2004 Systems and software engineering — High-level Petri nets — Part 1: Concepts, definitions and graphical notation. https://www.iso.org/standard/38225.html, last accessed 2020/08/28.
3. Law A. M.: Simulation modelling and analysis. President Averill M. Law & Associates, Inc.Tucson, Arizona, USA, www.averill-law.com. 5th Edition. — New York: McGraw Hill (2015).
4. Haas, P.J.: Stochastic Petri net: modelling, stability, simulation. Springer-Verlag New York (2002).
5. Balbo, G.: Introduction to Generalized Stochastic Petri Nets. In: Bernardo M., Hillston J. (eds) Formal Methods for Performance Evaluation. SFM 2007. Lecture Notes in Computer Science, vol 4486. Springer, Berlin, Heidelberg (2007).
6. Jensen, K.: Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. 2th edition. Springer-Verlag Berlin Heidelberg (1996).
7. Zhao, N., Yu, Y., Wang, J. et al. An extended object-oriented Petri net supporting the description and evolution of components: EOOPN. Cluster Computing 22, 2701–2708 (2019).
8. Silva, P.S. et al.: Simulation in Information Systems: Potential of the Vulnerability Theory. In: Quintela Varajão J.E., Cruz-Cunha M.M., Putnik G.D., Trigo A. (eds) ENTERprise Information Systems. CENTERIS 2010. Communications in Computer and Information Science, vol 109. Springer, Berlin, Heidelberg (2010).
9. van der Werf, J.M.E.M., Polyvyanyy, A.: The Information Systems Modeling Suite. In: Janicki R., Sidorova N., Chatain T. (eds) Application and Theory of Petri Nets and Concurrency. PETRI NETS 2020. Lecture Notes in Computer Science, vol. 12152. Springer, Cham (2020).
10. Stetsenko, I.V.: State equations of stochastic timed petri nets with informational relations. Cybernetics and Systems Analysis 48(5), 784-797 (2012).
11. Stetsenko, I.V.: Theoretical Foundations of Petri-object Modeling of Systems. Mathematical Machines and Systems 4, 136-148 (2011). (In Russian)
12. Miyamoto, T.: A Survey of Object-Oriented Petri Nets and Analysis Methods. IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences E88–A (11), 2964-2971 (2005).
13. Valk, R.: Object Petri Nets. In: Desel J., Reisig W., Rozenberg G. (eds) Lectures on Concurrency and Petri Nets. ACPN 2003. Lecture Notes in Computer Science, vol 3098. Springer, Berlin, Heidelberg (2004).
14. Fehling, R.: A concept of hierarchical Petri nets with building blocks. In: Rozenberg G. (eds) Advances in Petri Nets 1993. ICATPN 1991. Lecture Notes in Computer Science, vol 674. Springer, Berlin, Heidelberg (1993).
15. Jensen K., Kristensen L. M.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer-Verlag Berlin Heidelberg (2009).
16. CPN Tools. Dining Philosophers. http://cpntools.org/wp-content/uploads/2018/01/diningphilosophers.pdf, last accessed 2020/08/28.
17. GitHub. https://github.com/StetsenkoInna/PetriObjModelPaint, last accessed 2020/08/28.