Towards the Flexibility of Software for Computer Network Simulation

Alexander I. Mikov, Elena B. Zamyatina, and Roman A. Mikheev

***Abstract*—** This paper discusses network simulator TRIADNS. It is well known that the role of computer networks becomes more important due to progress in new computer technologies (distributed information systems, GRID-computing, Cloud computing and so on). So it is necessary to have effective and flexible program tools for computer network design and simulation. Indeed this program tools have to design computer networks with a lot of computer nodes, so simulation run must be time-consuming. Besides, it is necessary to study topological characteristics of computer network, to investigate traffic, to study computing node's behavior, to test the designed protocols, to study the behavior of routing algorithms and how these algorithms will behave when computer network topology is changed (new nodes can be added to network, some nodes can fail and so on). Network simulator must design and investigate not only hardware, but software too, explore computer networks, considering in particular the specific characteristics of a variety of computer networks.This paper considers approaches allowing to decide these problems: hierarchical model, using ontologies and Data Mining methods for the analyses of simulation results.

Keywords - simulation, computer networks, ontologies, routing algorithms, Data Mining, distributed and parallel simulation.

I. INTRODUCTION

Computer networks are very wide spread now. Indeed computer networks are used in distributed information systems, GRID computing, cloud computing and so on.

Widespread computer networks impose requirements to the speed and reliability of information transfer, to its effective treatment. For this reason, it becomes necessary to study traffic, to investigate new protocols, to design and develop new devices and new algorithms.

It is not always possible to apply analytical methods to investigate computer network because of the complexity of modeling object and, moreover, natural experiments can't investigate all aspects of this object too. So the designers prefer to use simulation methods and appropriate program tools (network simulators). A lot of network simulators were developed recently [1]. We consider some of them below.

Because of complexity of modeling object (computer networks) simulators should have the following properties:

- *Simulation experiment should be optimized in respect to time.* Indeed very often it is necessary to investigate large-scale networks with a tremendous amount of computing nodes. It is clear that the simulation of large-scale networks must be terminated within a reasonable time [2, 3]. But it is possible if one can perform simulation experiment on a supercomputer (cluster and so on). Besides, the investigators need the special software tools implementing special synchronization algorithm (conservative or optimistic), managing time advancement [4, 5, 6]. Moreover it is necessary to solve a problem of the equal workload on the computing nodes [7, 8, 9]. And nowadays new class of computer network simulators appears – there are simulators using graphical processors (GPU) [10].
- *A joint study of hardware and software of computer networks.* The computer network designers usually consider separately the hardware and software. However, the most appropriate solution would be to have software tools for design and analysis hardware, design and analysis of algorithms that control hardware, and for the co-design of hardware and software [11]. For example, it is very important to analyze the behavior of routing algorithm after the moment when the topology of computer network is changed (new computing node appears or some nodes become not accessible). In this case, the designer is interested in the topological characteristics of the network. These characteristics may effect the communication complexity of the algorithm. The structure of network may be represented as a graph. So it is important to investigate the structure of network using known graph algorithms (the shortest distance, for example). Nowadays the adaptable routing algorithms are applied in networks. These algorithms change their behavior depending on the values of certain characteristics of the network (overload of communication lines, for example). So it is advisable to simulate routing algorithm. Moreover it is important to simulate the behavior of various devices of computer networks and algorithms which control the behavior of these devices.

[1]This work was supported in part by Russian Foundation for Basic Research (grants 12-07-00302, 13-07-96506).

A. I. Mikov is with the Cuban State University, Krasnodar, Russia (e-mail: Alexander_Mikov@mail.ru)

E. B. Zamyatina is with the National Research University Higher School of Economics, Perm National State Research University, Perm, Russia (e-mail: E_Zamyatina@mail.ru).

R. A. Mikheev is with the Perm National Research State University, Perm, Russia (e-mail: Mikheev@prognoz.ru).

ISBN: 978-1-61804-237-8

- *Adaptability of software simulators to incorporate into a simulation model new devices and new algorithms that govern their work.* There are various software tools to design the computer networks nowadays. The most popular are: [12] (the design of the local and global networks, multiprocessor and distributed computing systems, the ability to assess the performance of the designed system , etc.); OMNeT + + [13] (a discrete event simulator that allows investigators to explore all levels of computer networks and to include customer modules into simulation model), NS- 2 [14], etc. Each of these simulators has specific characteristics. Some tools are designed to manage local networks, while others permit the design and analyses of global networks. Some of these software tools allow network designing, but have limited modeling capabilities, others are able to perform complex analysis of specific networks (may be only global networks or local or sensor ones). Network simulators have to be able to design, simulate and analyze new types of computer networks, new devices, new algorithms and technologies because of rapid development of network technologies.

The designers and developers of computer networks simulator TriadNS tried to consider the experience of various software tools of this kind. This simulator is based on CAD Triad [15]. The ideas embodied in CAD system Triad allow it to adapt to rapid change of computer networks, new algorithms and technologies due to special linguistic and program tools:

- Linguistic and program tools for the description of the structure of computer networks and the behavior of the devices and computing nodes;
- Advanced analysis subsystem, which includes a library of standard information procedures (information procedures are obtained to collect the information about simulation model during simulation experiment and to process it) and linguistic tools to create new procedures and, therefore, new algorithms of analysis.

Furthermore, the effectiveness of the simulator is provided by distributed (parallel) simulation experiment (using the resources of several nodes of computer network, cluster or multiprocessor (the advantages of a distributed (parallel) simulation experiment are listed in [5, 16]). Optimistic synchronization algorithm (based on knowledge) and load balancing subsystem are implemented in simulator TriadNS. This software permits to reduce the time needed for simulation experiment. Moreover the effectiveness of simulation system may be achieved by the subsystem of collecting and processing of the simulation model characteristics (the processing of data may be partly carried out during simulation experiments) and intelligent analysis of simulation results (based on the methods of Data Mining). Flexibility is achieved through the use of ontologies and the mechanism of redefining models, interoperability (including in the model components developed in the other modeling systems). First of all, we should talk about how the simulation model is presented in the simulator TriadNS, the architecture of simulator and the description of each it's subsystem.

II. The Simulation Model Representation in TriadNS

Simulation model in Triad.Net is represented by several objects functioning according to some scenario and interacting with one another by sending messages. So simulation model is μ={STR, ROUT, MES} and it consists of three layers, where STR is a layer of structures, ROUT – a layer of routines and MES – a layer of messages appropriately. The layer of structure is dedicated to describe objects and their interconnections, but the layer of routines presents their behavior. Each object can send a message to another object. So, each object has the input and output poles (Pin – input poles are used to send the messages, Pout – output poles serve to receive the messages). One level of the structure is presented by graph P = {U, V, W}. P-graph is named as graph with poles. A set of nodes V presents a set of programming objects, W – a set of connections between them, U – a set of external poles. The internal poles are used for information exchange within the same structure level; in contrast, the set of external poles serves to send messages to the objects situated on higher or underlying levels of description. Special statement <message> through <name of pole> is used to send the messages.

One can describe the structure of a system to be simulated using such a linguistic construction:

structure <name of structure> ***def*** (<a list of generic parameters>) (<a list of input and output parameters>) <a list of variables description> <statements>) ***endstr***

Special algorithms (named "routine") define the behavior of an object. It is associated with particular node of graph P = {U, V, W}. Each routine is specified by a set of events (E-set), the linearly ordered set of time moments (T-set), and a set of states {Q-set}. State is specified by the local variable values. Local variables are defined in routine. The state is changed if an event occurs only. One event schedules another event. Routine (as an object) has input and output poles (Pr_{in} and Pr_{out}). An input pole serves to receive messages, output – to send them. One can pick out input event e_{in}. All the input poles are processed by an input event, an output poles – by the other (usual) event.

routine<name>(<a list of generic parameters>)(<a list of input an0d output formal parameters>) ***initial*** <a sequence of a statements> endi event <a sequence of a statements> ***ende*** ***event*** <a name of an event> <a sequence of statements> ***ende*** … ***event***<a name of an event><a sequence of a statements> ende ***endrout***

The investigator may not describe all the layers. So if it is necessary to study structural characteristics of the model, only the layer of structures can be described. The example of computer network (the layer of structure) is given below. This computer network consists of a server and several clients.

Structure Client_Server[integer Number_of_Clients] ***def***
Client_Server := ***node*** Server<Receive, Send> +

```
node Клиент[ 0 : Number_of_Clients - 1 ] < Receive,
Send >;
integer i;
for i := 0 by 1 to Number_of_Clients - 1 do
   Number_of_Clients := Number_of_Clients +
    arc ( Client[ i ].Send -- Сервер.Receive ) +
    arc ( Сервер.Send -- Клиент[ i ].Receive );
endf;
endstr
```

Fig.1. The Structure of Computer Network Client_Server

Note, please, that the layer of structure is a procedure with parameters. Triad-model is considered as a variable. Initially it may be void and further may be constructed with the special statements of Triad-language (operations within the layer of structures.

Fig.1. gives the structure of network "Client_Server". It consists of the node "Server" and the attached array of nodes "Client". The links between nodes are set within the cycle for with the help of arcs. Input and output poles have to be specified: (arc (Server.Send -- Client[i].Receive)). The number of nodes Client may be changed by formal parameter Number_of_Client.

Client behavior scenario is described with special linguistic unit routine. The syntax of routine is given above. The description of the "Client" behavior is given below:

```
routine Client ( input Receive; output Send )[ real deltaT ]
  initial boolean Quiery_is_Send;
     Quiery_is_Send := false; schedule Quiery in 0;
     Print "Client Initialization";
  endi
  event Quiry; (* it is an event *)
     out "I send a quiry" through Send;
     Print "A Client sends a quiry to Server";
     schedule ЗАПРОС in deltaT;
  ende
  endrout
```

Fig.2. The Routine "Client"

The routine is a procedure with parameters too, it includes not only the interface parameters (input and output interface parameters "Receive" and "Send", but the parameter deltaT- the time interval between the queries of Clients to Server.

The instances of routine are formed by the statement ***let Client*** (clientDeltaT) ***be*** Client. An instance of routine may be "put" on an appropriate node with the help of statement: ***put*** Client ***on*** Model.Client[i]<Receive=Receive,Send=Send>. The input and output poles of routine are matched to the poles of node here.

There are two ways to describe model in Triad: via text editor or via graphical editor. The description of a layer of structure being built with the help of graphical editor is given below. This description is a fragment of computer network. It consists of several workstations sending messages between them. Besides, the computer network includes the routers responsible for the searching of the route.

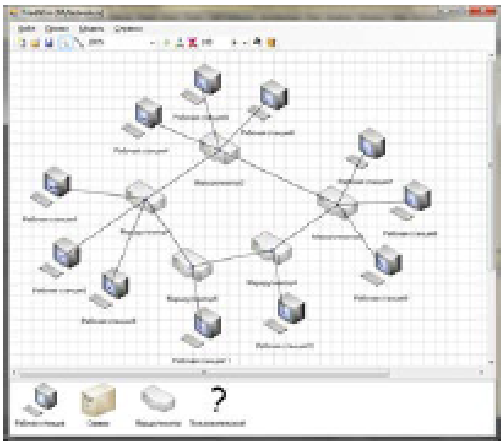

Fig.3. The fragment of computer network. Graphical editor.

The description of this fragment of computer network being built with the help of text editor is given on fig.4.

```
Type Router,Host; integer i;
M:=dcycle (Rout[5]<Pol>[5]);
M:=M+node (Hst[11]<Pol>);
for i:=1 by 1 to 5 do
      M.Rout[i]=>Router;
   M:=M+edge(Rout[i].Pol[1] — Hst[i]);
endf
for i:=1 by 1 to 3 do
   M:=M+edge(Rout[i].Pol[2] — Hst[2*i-1]);
endf;
for i:=0 by 1 to 11 do M.Hst[i]=>Host; endf;
```

Fig.4.The fragment of computer network in Triad language

Simulation model (see fig.4.) is built using graph constants. A set of special linguistic units - graph constants - presents the basic types of topologies of computer network. In the text given above the graph constant "directed cycle" (Dcycle) was used. Besides, in above example the semantic types (Type Router,Host) were used. Namely they are "router" and "host". The semantic types are used for simulation model redefining. More details will be given later.

There are the several standard procedures in the structure layer. The investigator is able to take out from the structure of model a lot of characteristics: a set of nodes, a set of arcs, a set of edges and etc. Moreover one can find the shortest distance between two nodes, connected components (procedure GetStronglyConnectedComponents(G)), selection of the structure layer (procedure GetGraphWithoutRoutines(M)) and so on.

Besides, the investigator obtains the linguistic and programming tools enabling him to write the absent procedure by himself.

The investigation of the structure layer only is static process. The simulation process may take place only after the definition of the behavior of all nodes. As it was noted above the behavior is determined by the statement ***Put***.

It is well known that a simulation is a set of object functioning according to some definite scenarios controlled by synchronizing algorithm. The simulation run is initialized by the statement ***simulate***:

Simulate <список моделей> ***on condition of simulation*** <имя условия моделирования>(<настроечные параметры>)(<параметры интерфейса>)(<список информационных процедур>; <последовательность операторов>).

One can pay an attention to the fact that the several models may be simulated under the same conditions of simulation simultaneously.

III. THE ALGORITHM OF INVESTIGATION

The objects of simulation model are managed by the special algorithm during the simulation run. Let us name it as "simulation algorithm" (CAD system Triad has distributed version and corresponding algorithm for distributed objects of simulation model too) [15]. CAD system Triad includes analyses subsystem implementing the algorithm of investigation - special algorithm for data (the results of simulation run) collection and processing.

The analysis subsystem includes special objects of two types: *information procedures* and *conditions of simulation*. Information procedures are "connected" to nodes or, more precisely, to routines, which describe the behavior of particular nodes during simulation experiment. Information procedures inspect the execution process and play a role of monitors of test desk.

Conditions of simulation are special linguistic constructions defining the algorithm of investigation because the corresponding linguistic construction includes a list of information procedures which are necessary for investigator.

The algorithm of investigation is detached from the simulation model. Hence it is possible to change the algorithm of investigation if investigator would be interested in the other specifications of simulation model. For this one need to change the conditions of simulation. But the simulation model remains invariant. We may remind that it is not possible in some simulation systems.

One can describe the information procedure as so:

information procedure*<name>(<a list of generic parameters>)(<input and output formal parameters>)*

initial *<a sequence of statements>* ***endi***

*<a sequence of statements>****processing*** *<a sequence of statements>…****endinf***

It is possible to examine the value of local variables, the event occurrence and the value of messages which were sent or received. A part of linguistic construction 'processing' defines the final processing of data being collected during simulation run (mean, variance and so on). Let us present the linguistic construction ***conditions of simulation***:

Conditions of simulation<name>(<a list of generic parameters>)(<input and output formal parameters>) initial <a sequence of statements> ***endi*** <a list of information procedures> <a sequence of statements> ***processing*** <a sequence of statements>…***endcond***

The linguistic construction ***conditions of simulation*** describes the algorithm of investigation which defines not only the list of information procedures but the final processing of some information procedure and checks if conditions of simulation correspond to the end of simulation.

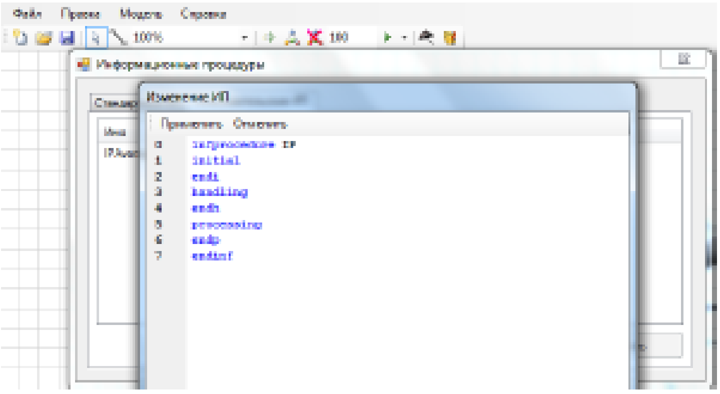

Fig.5. The form for information procedure

The subsystem of visualization represents the results of simulation. One can see the representation of the results of simulation run at fig.6.

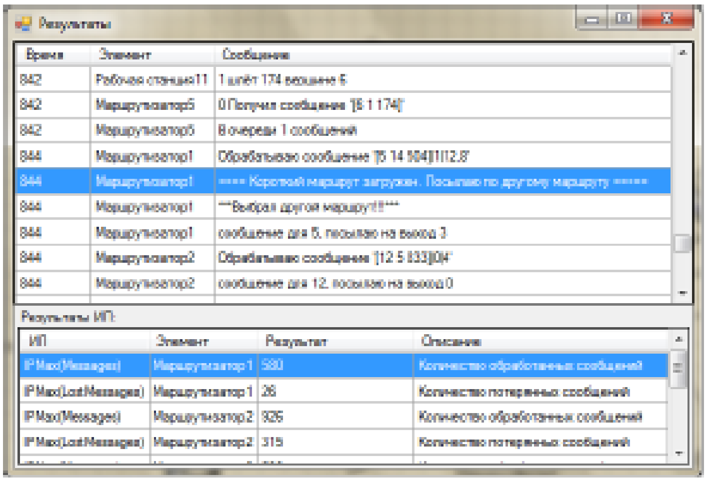

Fig.6. The results of simulation

IV. THE COMPONENTS OF SIMULATION SYSTEM TRIADNS

Let us consider simulation modeling system TriadNS, its appointment, its components and functions of each component. So simulation system Triad.Net is a modern version of previous simulation modeling system Triad [6] dedicated to computer aided design and simulation of computer systems. Triad.Net is designed as distributed simulation system, so various objects of simulation model may be distributed on the different compute nodes of a computer system. One more specific characteristic of Triad.Net – remote access, so several investigators may fulfill a certain project from different computers situating in different geographical points.

Distributed simulation system Triad.Net consists of some subsystems: compiler (TriadCompile), core of simulation system (TriadCore), graphical and text editors, subsystem of testing and debugging (TriadDebugger), subsystem of distributed simulation (synchronization of simulation model objects which are situated on different compute nodes of computer system, conservative and optimistic algorithms realization)(TriadRule), subsystem for equal workload of compute nodes (TriadBalance), subsystem of remote and local access (TriadEditor), subsystem of automatic and semiautomatic simulation model completeness (TriadBuilder), the subsystem for remote access and a security subsystem from external and internal threats TriadSecurity), the subsystem of automatically extending the definition of the model (TriadBuilder), the subsystem of intellectual processing of the results of simulation experiment (TriadMining). Initially we address to the specific characteristics of simulation model in TriadNS.

V. Knowledge Representation

It is important to involve into the simulation process not only the specialists in simulation but the specialist in specific domains and specialists in the other spheres of knowledge. That is why it is necessary to adjust a simulation system to specific domain. Indeed the investigator of computer network may use a graph theory while studying the structure of network, or a queue network theory, or the theory of Petri Nets. Ontologies are used in TriadNS to adjust the simulation system to specific domain.

Ontologies can be applied on the different stages of simulation [17, 18]. Very often ontologies are applied for the simulation model assembly. So the simulation model may consist of separately designed and reusable components. These components may be kept in repositories or may be found via Internet. The ontologies keep the information about interconnections of simulation model components and other characteristics of these components.

Ontologies enable investigators to use one and the same terminology.

Ontologies allow to make the repositories of components to store not only an information about their characteristics, interfaces, but the information about their interconnections.

The base ontology is designed in TriadNS.

Its basic classes are: TriadEntity (any named logic entity), Model (simulation model), ModelElement (a part of simulation model and all the specific characteristics of a node of structure layer), Routine (node behavior), Message (note, please, that structure layer nodes of simulation model can interchange with messages) and so on.

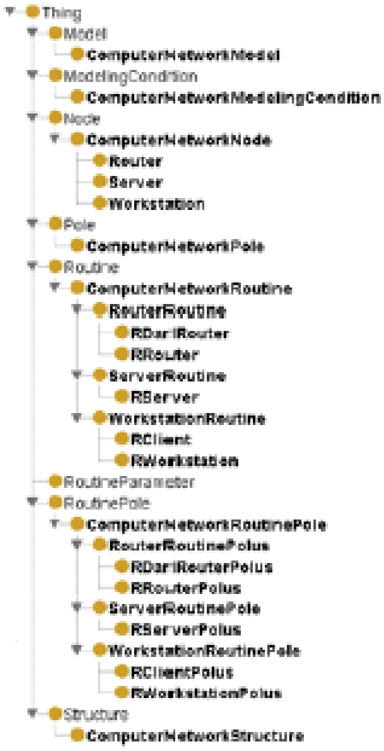

Fig. 5. The hierarchy of the classes of the ontologies in TriadNS

The basic properties of base ontology are:

- *The property of ownership:* model has a structure, a structure has a node, a node has a pole and so on.
- *The property to belong to something* — inverse properties to previous one– The structure belongs to the model, the node belong to strucrure, the pole belong to the node and so on.
- *The properties of a pole and an arc connection:* connectsWithArc (Pole, Arc), connectsWithPole (Arc, Pole).
- *The property of a node and an appropriate routine binding*-putsOn (Routine, Node).
- *The properties of a node and an appropriate structure binding:* explicatesNode (Structure, Node), explicatedByStructure (Node, Structure).
- *The property of the model and conditions of simulation binding* (Model, ModelingCondition).

The simulator TriadNS has some additional special subclasses of the base classes (specific domain – computer networks). (fig.6.):

- *ComputerNetworkModel* (a model of a computer network), ComputerNetworkStructure (a structure of a computer network model).
- *ComputerNetworkNode* (a computer network element, it contain several subclasses: Workstation, Server, Router).
- *ComputerNetworkRoutine* (a routine of a computer network) и т.д.

This ontology includes two special properties of a pole. These properties are used to *check* the conditions of matching routine to a node:

ISBN: 978-1-61804-237-8

- *isRequired(ComputerNetworkRoutinePole, Boolean)* – this property check if it is necessary to connect a pole with another pole?
- *canConnectedWith(ComputerNetworkRoutinePole, ComputerNetworkRoutine)* –this property check the semantic type of an element of a structure being connected.

VI. Redefining of Simulation Model

An ordinary simulation system is able to perform a simulation run for a completely described model only. At the initial stage of designing process an investigator may describe a model only partly omitting description of behavior of a model element μ_{r*} = {STR, ROUT*, MES}). Simulation model may be described without any indication on the information flows effecting the model (μ_{s*} = {STR*, ROUT*, MES}) or without the rules of signal transformation in the layer of messages (μ_{m*} = {STR, ROUT*, MES}). However for the simulation run and the following analysis of the model all these elements have to be described may be approximately.

For example, in a completely described model each terminal node $v_i \in V$ has an elementary routine $r_i \in$ ROUT. An elementary routine is represented by a procedure. This procedure has to be called if one of poles of node v_i receives a message. But some of the terminal nodes v_i of partly described model do not have any routines. Therefore the task of an automatic completion of a simulation model consists either in "calculation" of appropriate elementary routines for these nodes, i.e. in defining $r_i = f(v_i)$, either in "calculation" of a structure graph $s_i = h(v_i)$ to open it with (in order to receive more detailed description of object being designed). It was mentioned above that the routine specifies behavioral function assigned to the node, but the structure graph specifies additional structure level of the model description. And at the same time, all structures s_i must be completely described as the submodels.

These actions have to be fulfilled by the subsystem TriadBuilder.

Subsystem TriadBuilder [19] attempts to search the appropriate routine by the help of base ontology (it was described earlier). It may be found thanks to special semantic type (semantic type "Router" and "Host", for example).

Model completion subsystem starts when the internal form of simulation model is built according to a Triad code.

First, *model analyzer* searches the model for incomplete nodes, and marks them. Thus, the model analyzer will mark all *Rout* nodes. After the inference module starts looking for an appropriate routine instance for each of marked nodes according to specification condition (the semantic type of node and routine must coincide).Then the condition of configuration must be checked (the number of input and output poles of node and the number of poles of routine must coincide).

After the appropriate instance has been found, it may be put on the node.

VII. Intellectual Analysis of the Simulation Experiment Results

It is well known that the goal of a simulation experiment is to obtain the most accurate and adequate characteristic of the studied object. This stage of simulation deals with data collection and processing. The special syntax units such as information procedures and conditions of simulation are designed in TriadNS. Information procedures and conditions of simulation are described above. Note, please, that data collection and data processing with the help of information procedures permit to obtain more adequacy results. Information procedures monitor only these characteristics of simulation model which are interested for investigator. In contrary some other simulators able to monitor and to collect a set of predefined characteristics.

But we can note another problem: the results of simulation experiment are not ordered and not structured. The processing of a simulation experiment results requires highly skilled analysts.

So we can state the appearance of several papers with the suggestion to make the additional processing of the results of simulation experiments [20] and to apply the methods of Data Mining for these purposes [21].

Usually investigators obtain standard report with the results of simulation. The additional processing allow to find dependences between characteristics of the modelling objects.

The analyses of these dependences allow to reduce the overall data capacity, dimension of problem and eventually to optimize the simulation experiment.

The additional processing may be done with the special software tools of TriadNS (component TriadMining). TriadMining use the results of the information procedures, the results are processed with the help of regression analyses, time serious, Bayesian networks and so on. We mentioned above that an information procedure monitors the implementation of the sequence of events, the variables changing and so on. It is well known that the sequence of the predefined events allow to find crashes in nodes of telecommunication systems. Here is an example of information procedure.

information procedure event_sequence (***in ref event*** E1,E2,E3;***out Boolean*** arrived)
  initial interlock (E2,E3); Arrived := false;
   case of e1:available(e2);
     e2:available(E3):
          e3:ARRIVED:=true;
   endc
  endinf

Fig. 6. The information procedure to detect the proper sequence of events

So investigator may detect the arrival of the sequence of events E1→E2→E3. The statement ***interlock*** provides input parameter blocking (event E1 in this case). It means that information procedure doesn't watch parameters being marked in interlock statement. The statement ***available*** allows beginning the marked parameter monitoring again.

Information procedure monitors the changing of variables and the moments of appropriate time. So the time series may be formed. It is necessary to analyze the similarity of two or more time series. So it is possible to find dependences between the elements of simulation model and reduce the data capacity.

VIII. Conclusion

The paper discusses the problems of flexible software for computer network simulation. Authors consider ontology approach application to automatic redefining of simulation model and to adjusting the simulation system to the specific domain.

Simulator TriadNS is provided with a convenient graphical interface. Simulator permits separate and joint hardware and software modelling. Another distinguished characteristic of the simulator is the ability to make a distributed simulation experiment.

The Data Mining methods allow to simplify the analyses of the simulation experiment results. Ontologies enable to automate the simulation model construction and to achieve the interoperability of the software tools (to use components designed in the other simulation systems).

References

[1] S. Salmon, H.ElAarag. Simulation Based Experiments Using Ednas: The Event-Driven Network Architecture Simulator. In Proceedings of the 2011 Winter Simulation Conference S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds. The 2011 Winter Simulation Conference 11-14 December 2011. Grand Arizona Resort Phoenix, AZ, pp. 3266-3277.

[2] A.I.Mikov, E.B.Zamyatina The simulation model technologies for big systems investigation // In Proceedings of the Scientific Conference "Scientific service on the Internet" – M.: MSU, 2008. C.199-204.[in Russian]

[3] Y.Liu, Y.He. A Large-Scale Real-Time Network Simulation Study Using Prime. M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds. The 2009 Winter Simulation Conference 13-16 December 2009. Hilton Austin Hotel, Austin, TX, pp. 797-806.

[4] Riley, R.M. Fujimoto, M. Ammar. A Generic Framework for Parallelization of Network Simulations", in Proc. 7th Int.Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 1999, p. 128-135.

[5] R.M. Fujimoto Distributed Simulation Systems. In Proceedings of the 2003 Winter Simulation Conference S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds. The 2003 Winter Simulation Conference 7-10 December 2003. The Fairmont New Orleans, New Orleans, LA, pp. 124-134

[6] E. Zamyatina, S. Ermakov. The Synchronization Algorithm of Distributed Simulation Model in TRIAD.Net. Applicable Information Models. ITHEA, Sofia, Bulgaria, 2011, ISBN: 978-954-16-0050-4, pp.211-220.[in Russian]

[7] L. F. Wilson, W. Shen Experiments in load migration and dynamic load balancing in Speedes // Proc. of the Winter simulation conf. / Ed. by D. J. Medeiros, E. F.Watson, J. S. Carson, M. S. Manivannan. Piscataway (New Jersey): Inst. of Electric. and Electron. Engrs, 1998. P. 487–490.

[8] G. Zheng Achieving high performance on extremely large parallel machines: Performance prediction and load balancing: Ph.D. Thesis. Department Comput. Sci., Univ. of Illinois at Urbana-Champaign, 2005. 165 p. [Electron. resource]. http://charm.cs.uiuc.edu/.

[9] A.I.Mikov, E.B.Zamyatina, A.A.Kozlov The Multiagent Approach to the Equel Distribution of the Workload. Natural and Artificial Intelligence, ITHEA, Sofia, Bulgaria, 2010, pp.173-180.

[10] L. Djinevski., S. Filiposka, D.Trajanov Network Simulator Tools and GPU Parallel Systems. In Proceedings of Small Systems Simulation Symposium 2012, Niš, Serbia, 12th-14th February 2012, pp.111-114

[11] W.Hu, H.S. Sarjoughian A Co-Design Modeling Approach For Computer Network Systems. . In Proceedings of the 2007 Winter Simulation Conference S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds. The 2007 Winter Simulation Conference 9-12 December 2007 J.W. Marriott Hotel, Washington, D.C., pp. 124-134

[12] [NS-2. 2004] The Network Simulator - NS-2. Доступно на сайте: http://www.isi.edu/nsnam/ns [Проверено 21 марта 2012]

[13] [OPNET, 2004] OPNET Modeler. Доступно на сайте: <http://www.opnet.com> [Проверено: 21 марта 2012]

[14] [OMNeT++, 2005] OMNeT++ Community Site. Доступно на сайте: http://www.omnetpp.org. [Проверено: 21 марта 2012]

[15] A.I. Mikov Simulation and Design of Hardware and Software with Triad// Proc.2nd Intl.Conf. on Electronic Hardware Description Languages, Las Vegas, USA, 1995. pp. 15-20.

[16] R.E. Nance Distributed Simulation With Federated Models: Expectations, Realizations And Limitations. In Proceedings of the 1999 Winter Simulation Conference. P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds., The 1999 Winter Simulation Conference 5 – 8 December 1999 Squaw Peak, Phoenix, AZ, pp. 1026-1031.

[17] P Benjamin., K.V Akella., K Malek., R Fernandes. An Ontology-Driven Framework for Process-Oriented Applications // Proceedings of the 2005 Winter Simulation Conference / M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.,– pp 2355-2363

[18] P. Benjamin.,M. Patki, R. J Mayer. Using Ontologies For Simulation Modeling // Proceedings of the 2006 Winter Simulation Conference/ L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds. –pp.1161-1167

[19] A.Mikov A., E.Zamyatina, E. Kubrak. Implementation of simulation process under incomplete knowledge using domain ontology. In proceedings of 6-th EUROSIM Congress on modeling and Simulation. 9-14, September, 2007, Ljubljana, Slovenia, Vol.2. Full papers, 7 pp.

[20] G. Neumann, J.Tolujew , From Tracefile Analysis to Understanding the Message of Simulation Results, proceeding of the 7th EUROSIM Congress on Modeling and Simulation, Prague, Czechia, 2010, 7 pp.100-117

[21] T. Brady, E.Yellig, Simulation Data Mining: a new form of simulation output, 37th Winter Simulation Conference, Orlando, USA, 2005, pp 285-289.