Application of Java-Technologies for Simulation in the Web

Volodymyr Kazymyr, Nataliya Demshevska

Department of Computer and Information Systems
Chernihiv State Technological University
Shevchenko st., 95
Chernihiv, 14000, Ukraine

phone: (+380)(462) 95-56-87
email: vlad@cg.ukrtel.net
ndemsh@cs.stu.cn.ua

Abstract: In this article, the visual Internet-oriented Java-based E-net Simulation System (JESS) is considered. JESS supports all the simulation stages from constructing the models to experimenting with them. The basis of JESS is an object-oriented formal model and a specification language that supports several levels of model description with an E-net specification. The Internet-orientation of the system allows to create models executed on a remote server and makes these models available from any point of the network. This article contains descriptions of the JESS simulation technique and the implementation of its Internet properties.

1 Introduction

Simulation systems take a special place among all the Internet applications. Now the task of complex systems simulation is becoming more and more urgent because simulation is the main tool in such kind of system research. To satisfy the users growing need for model construction and all the dynamic changes, it is necessary to make the sharing of one task among several research groups possible. Thus, our problem is to support the information exchange among these groups. Intranet/Internet is one of the possible tools for solving this problem.

Until recently, simulation models were mainly monolithic by design. With appearance of object-oriented programming, these models became more structured, but still, they remained monolithic. Internet for such models has changed a little. To use all the advantages of Intranet/Internet for solving simulation problems, model construction should be based on different principles.

The appearance of Internet has created an environment that caused many disciplines to revise their approaches. Disciplines connected with simulation are not among the exceptions: in particular, the concept of "simulation in Web" has appeared.

Web-based simulation represents the connection between the World Wide Web (WWW) and the field of simulation. Web-based simulation is not an existing field but rather an

idea which represents an interest on the part of simulationists to exploit web technology [1].
The web represents a new way of both publishing and delivering information. Model is a form of information. In a web-based simulation, a user sets models in any Internet-browser. The simulation is carried out on a remote server. The user receives simulation results and then is able to continue model experiments with new parameters. Thus, we can note the main advantages of web-based simulation. They are remote distributed model development and execution and distributing of model specifications over the net.
One of the basic programming languages of WWW applications is the object-oriented programming language JAVA. JAVA language has a number of advantages that allow to consider it as the main tool of Web-simulation. Here are some of these advantages:

multi-threading;
platform-independence;
network support;
built in support for sophisticated animations;
it is safer and easier to learn than C++.

The WWW and JAVA integration allows:

to develop the Web-oriented environments of model designing and setting-up;
to carry out the runs and the analysis of models, executed on the server, from any point of the network;
to carry out the development of the distributed models based on the uniform conceptual approach;
to combine the efforts of several research groups and separate developers for model creation and analysis;
to implement the concept of renting the applications placed on the server;
to create libraries of reusable model components.

JAVA-based simulation gives a unique opportunity to revolutionary change the development of tools for the simulation process support. JAVA gives a new vision of the simulation industry where simulation experts in specific areas generate compatible reusable simulation components. The simulation with the components placed on the server has a number of fundamental improvements compared to traditional simulation [2]. These components can be developed using of inexpensive professional quality JAVA development environments. They then can be located in the repository and executed through the Internet browser.

In the article the following questions are examined:

section 2 contains the description of the existing environments and languages for simulation in Web. Theirs advantages and disadvantages are analyzed;
section 3 includes the main tasks that modern simulation systems developers are concerned with;
sections 4 - 7 contain the description of JESS;
section 8 concludes the article with the definition of system improvements paths.

2 Existing environments and languages for web based simulation

Presently, two main approaches are used for Web simulation, both of which are based on JAVA. These appro aches are [3]:

1. Development of new simulation languages based on JAVA.
2. Port an existing simulation language, such as GPSS, and create it in JAVA.

Several environments for discrete-event simulations based on JAVA are described below.

Silk
A commercial processes -oriented discrete-event simulation package. The features of the process-oriented simulation are supplemented with an object-oriented general-purpose language JAVA . It provides a visual modeling environment where Silk-based modeling components can be graphically assembled using JavaBeans to create simulation applications in of software environments such as Symantec's Visual Cafe, IBM's Visual Age, and Microsoft's J ++.

JavaSim
A set of Java packages for process-oriented discrete-event simulation similar to simulation in Simula and C++SIM (from which JavaSim system has taken place). JavaSim doesn't have graphical interface.

JavaGPSS
JavaGPSS compiler is a simulation tool designed for the Internet. The objective was to crate a GPSS implementation that could truly be run as an applet in any Internet browser. The JavaGPSS compiler is a Java program that translates GPSS source files into Java source code.

WSE
WSE (Web-enabled Simulation Environment) combines the use of a new web technology with the use of JAVA and Corba. WSE environment provides transparent access to simulation models and tools; dynamic acquisition, instantiation and/or modification of simulation models, global availability; and plug-and-use architecture to easily embed simulation and tools.

Simjava [4]
A process-oriented discrete-event simulation package for building models of complex systems with animation facilities. This model represents a collection of entities each running in its own thread. A user describes entities behavior in Java classes. The extension of this package allows create animated applets. There is a version of the distributed Simjava system.

Each environment of this kind includes the simulation language. Despite all the efforts of the developers, syntax of simulation languages overload the program model with details that don't t actually describe any of the system's additional features. Consequently, there appears to be a strong need in a high-level form of model representation that is free from the implementing all the described details. Besides, errors may appear because of bad links between the conceptual model (formal model) and the descriptive abilities of a concrete programming language.

Thus, it should be noted the following characteristics of the described here environments and simulation languages which can be considered as their disadvantages:

- lack of formal model (Silk, JavaSim, WSE, Simjava);
- lack of model verification (Silk, JavaSim, JavaGPSS, WSE, Simjava);
- model complexity and unclearness (Silk, JavaSim, WSE, Simjava);
- simulation language is the general-purpose programming language unknown to the majority of experts in subject fields (Silk, JavaSim, Simjava);
- lack of visual means (JavaSim);
- lack of program components use (JavaGPSS, JavaSim, Simjava).

3 The solving tasks

The task of simulation environment development that includes a simulation language and allows build models not only professional programmers but to expert users from any specific area is urgent today. Such simulation environment should fully support the life cycle of simulation models and include tools for model verification. It should also include automatic support of the development processes, analysis, compilation to program model, saving/loading and control of the simulation process.
Solution of these tasks assumes the following development stages:

1. Formalization method (it allows multi-level modeling, including a model definition in subject fields terms and model verification).
2. Specification language (its use will make the model clear for specialists from different subject fields).
3. Universal simulation environment, which supports a developed formalization method (it provides the continuity of a simulation cycle and models visualization with use of the uniform environment).
4. Tools, which ensure models distribution in the Internet network and application of web tools (it supports component using (such as CORBA and EJB) in different distributed environments).

True applications distribution can only be attained based on component technologies. In JAVA-applications, Enterprise JavaBeans (EJB) is a technology of such kind. EJB consist of reusable compiled code that is designed to be installed inside a special application server that is compliant with the Java 2 Platform Enterprise Edition.
JESS, an Internet-oriented visual system of a simulation, was developed to combine the power of simulation based on the use of a server that contains components and object-oriented design with the simplicity of models programming within a set of tasks. This system allows: create models; modify already existing models and make experiments with them; create EJB simulation components and execute them on the server. JESS

system contains a specification language that allows a user with no special preparation in programming to construct models and to research them.

4 The Formal method

The use of the formal model allows to verify the obtained models on a top abstraction level and thus to prevent potential errors in the design stage. It also encourages the creation of simulation languages and methods that enable developers to execute model decomposition on any specified level of abstraction and contain possibility of model verification.
The use of the network schemes (Petri-nets, for example) as the simulation language has an unconditional advantage that allows to represent a model of the system as a set of simple graphical constructions where such concepts as parallelism and synchronization can easily be shown. The most powerful Petri-nets extension is the E-nets [5] that provide not only qualitative, but also quantitative analysis of simulated systems.
The formal theory of E-nets allows to easily carry out verification of the obtained models. According to this, the authors developed an object-oriented formal model, which represents unification of the E-nets formal theory and the aggregates formal theory. The aggregates approach [6] is used as the structure concept.
In the given approach, the simulation scheme can be presented as the three-level architecture [7]:

1. *Models level M*, where a model is constructed from a set of aggregates:

$$M = (A, R, V)\ , \tag{1}$$

where M – model;
A – finite nonempty subset of aggregates;
R – scheme of aggregates interfaces;
V – set of a model's variables.

2. *Aggregates level A*, which is used for an internal structure description of model unit with the help of E-net:

$$A = (T, U)\ , \tag{2}$$

where T – finite nonempty subset of transitions;
U – set of an aggregate variables.

3. *Transitions level T*, on which execution of E-network transitions is carried out:

$$T = (D, P, Z, F, M)\ , \tag{3}$$

where D – transition type;
P – finite nonempty subset of positions consisting of not intersected subsets of simply and control positions;
Z – time of transition execution;
F – transition procedure;
M – mark function that defines initial marks of transition positions.

On the first level the model of the system is represented as a collection of aggregates-subsystems that have ins and outs. On the second level the structure of aggregates are detailed. Aggregates-subsystems can be defined precisely through the E-nets. On the third level, the process of execution of separate E-net transitions is defined. It has been

proved that E-nets are equivalent to aggregates and are a formal-complete system for algorithm description. For this reason, any operation process can be described using offered scheme.
One of the possible implementations of formal model is use it in the object-oriented approach (OOA). The concepts of object-oriented approach ideally suit the implementation models of real systems. For example, parallelism is easy represented in the object-oriented model because the objects exist and behave independently.
The model of complex system which is based on the object-oriented formal model, represents a collection of objects-aggregates that cooperate with other aggregates according to the rules of aggregate systems interactions. If the E-net transition is considered as the object, the aggregate will represent a collection of interacted objects-transitions that contain time delay procedures, transformation procedures, and controlling procedures. Objects-transitions are fixed in adjacent positions. The information exchanges between these positions is carried out through the transmission of labels-objects that have a set of attributes.

5 Specification language

One of the basic elements of JESS is the graphical specification language (SL) that supports all the simulation stages and serves for model and experiment description. According to the principles of the system approach, SL for the simulation of complex systems should supports four kinds of descriptions:

1. Graphical notation of conceptual model description. The system is represented as a structure of interconnected aggregates (units of the system).
2. Graphical notation of the formal description of complex system units (aggregates) that are represented as E-nets.
3. Description of separate aggregate transitions such as definition of time delay procedures, transformation procedures, and controlling procedures of E-net transitions.
4. Experiments with a model definition are based on filling in the experiment template using one of the selected schemes.

The first language level of conceptual model description allows to carry out the structural analysis of complex systems based on the decomposition method. A system can be represented as a multilevel hierarchical structure with separate aggregates as its units. Each aggregate can be a subsystem. In this case, it is constructed according to the same principles. The task of the specification language on this level is to define the properties of the aggregates (global variables) and construct a scheme of aggregates interface with link objects. On the second level, the graphical language of E-nets is used. With the use of graphics tools, the internal aggregate structure is described as E-net. The next SL level is used to define the E-net transition procedures. In this case, the context-free language is developed. This language includes a set of key words, an assignment operator, logical operators, and mathematical and statistical functions with a developed set of functions for the implementation of various casual quantity distribution laws. The last language level provides the definition of experiment conditions. It is based on filling

in the offered templates that require to indicate the necessity for primary and secondary statistics collection and provides the definition of the experiment parameters.
On each SL level, model representation is transformed to a correspondent JAVA representation with the help of JESS because JAVA is the main language of JESS implementation. It is also possible to write a model in the JAVA language. Such model will use the main classes and objects of the specification language. It can be converted to a graphics form of system representation and viewed using JESS graphics editor.
JESS uses a convenient visual form of graphical model constructions which simplifies the verification process and allows to cover all the simulation process, including conceptual, formal, and program model development stages.

6 JESS as a visual simulation system

In the selection of the tools for the implementation of the formal model, the emphasis was placed on the possibility of distributed systems implementation in Internet-network. JAVA was selected as a programming language not only because it is an object-oriented language relevant for the selected formal approach, but also because it provides platform independence of program execution.
JESS consists of a graphics editor, a procedure text editor, a compiler for procedure description language, and a model interpreter with an experiment plan. It is has animation facilities. JESS structure is represented in Fig 1.

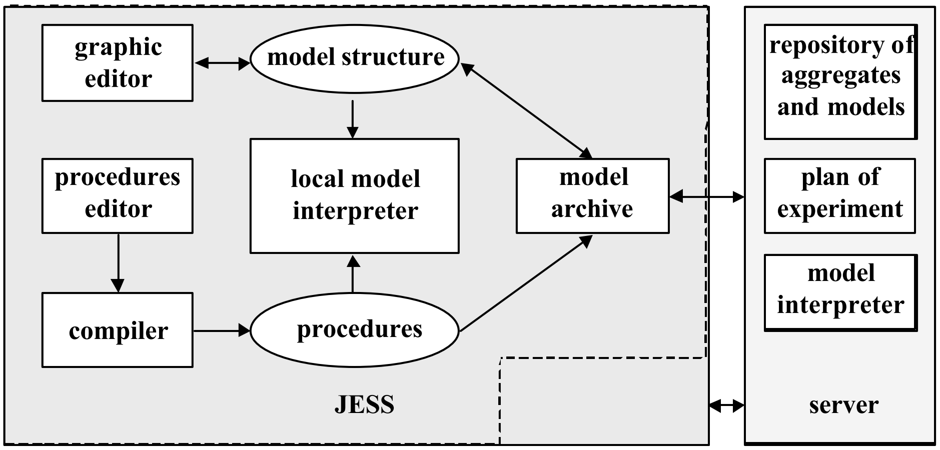


Fig. 1: JESS structure

The graphics editor serves for the visual construction of the E-net and the aggregate model. On the transition level of model construction, a user defines the control procedures, transformation procedures, and time delay procedures for each transition and defines the initial network marking. The model, constructed with the help of the graphics editor, is then transformed into an internal data structure. The compiler translates the transition procedures written in the specification language into a byte-code. Then model

interpreter executes the model according to the given experiment plan. It is also acceptable to use procedures, aggregates, and models classes, brought into the network from Web-servers.
Thus, instead of tiresome programming, there is a possibility to rapidly construct a program model from separate aggregates that are included in the repository.

JESS simulation technique

The use of the simulation technique can be demonstrated on an elementary service system. The model of such system can be presented as a combination of three aggregates: a generator, a queue, and a service device.
On the formal level, each aggregate is represented as the E-net scheme and is described in the SL.
Originally, the structure of each aggregate in SL is described in the graphics editor. There are classes for all types of transitions in JESS. The structure of an aggregate class is formed when a user chooses transitions of appropriate types and places them in the graphics editor window. If the aggregate has ins and outs, they are represented by graphics constructions as well. The links between transitions and positions are carried out by special links-objects.
Fig. 2 represents an aggregate model of the described service system. Aggregates are constructed in JESS graphics editor. For example, the generator aggregate contains 3 positions and 2 transitions. The P2 position is the out position of this aggregate.

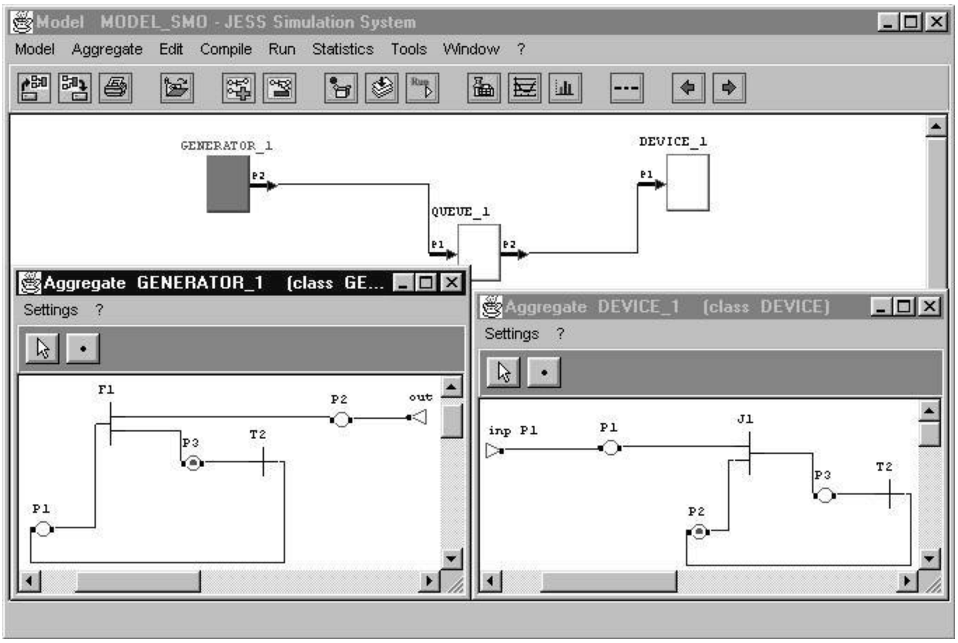

Model MODEL_SMO - JESS Simulation System
Model Aggregate Edit Compile Run Statistics Tools Window ?
GENERATOR_1
P2
QUEUE_1
P1
P2
DEVICE_1
P1
Aggregate GENERATOR_1 (class GE...
Settings ?
F1
P2
out
P3
T2
P1
Aggregate DEVICE_1 (class DEVICE)
Settings ?
inp P1
P1
J1
P3
T2
P2

Fig. 2: The model of the service system

The generator aggregate in JAVA programming language can be described as:

```
Position p1 = new Position("P1", 0, Position.S_POSS, 87, 122);
Position p2 = new Position("P2", 0, Position.S_POSS, 300, 44);
Position p3 = new Position("P3", 1, Position.S_POSS, 194, 75);
...
 GenTimeDelay tp_f1 = new GenTimeDelay(this);
PASSF f1 = new PASSF("F1", 2, tp_f1, null, 133, 37);
f1.add(p1, 0);
f1.add(p2, 1);
f1.add(p3, 2);
...
setOutPosition(p2, 355, 42);
```

The first three lines of the listing represent the creation of the generator aggregate positions. For each position, a name, the presence/absence of marking, a position type (simple or controlling) and the coordinates in the JESS graphics editor window are indicated. Then the following settings are made: the controlling procedure settings and time delay and transformation procedure settings that describe the operation process of separate transitions. The aggregate's global variables are set on this level as well.
The following program level is used to describe separate aggregate transitions and set the time delay procedure as well as the transformation and the controlling procedures.
When aggregates are constructed, the process of model description begins. A model is described on the top level of the SL. Model structure includes aggregates and links between them. In addition, it is possible to set marking and to change the initial values of global variables for each aggregate-object. The model description in JAVA looks like this:

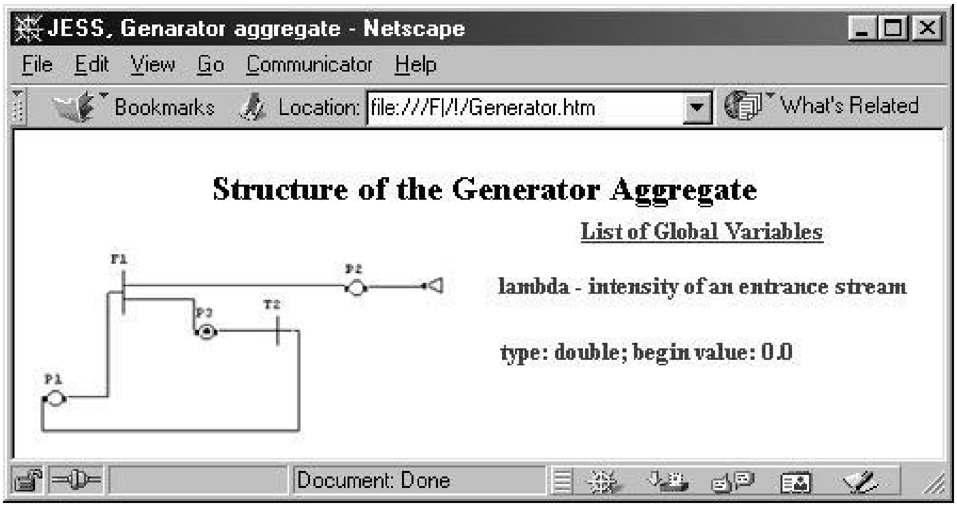
```
GENERATOR generator_1 = new GENERATOR("GENERATOR_1", 120,...)
QUEUE queue_1 = new QUEUE("QUEUE_1", 299, 78);
DEVICE device_1 = new DEVICE("DEVICE_1", 448, 28, 0.0);
...
Link l_generator_1p2_queue_1p1 = new Link(generator_1,
      "P2", queue_1, "P1", "l_generator_1p2_queue_1p1");
Link l_queue_1p2_device_1p1 = new Link(queue_1,
      "P2", device_1, "P1", "l_queue_1p2_device_1p1");
...
```

The first three lines of the listing represent the creation of the model aggregates. Next, their names and coordinates in the JESS graphics editor are set. The rest of the lines contain the description of links between aggregates.
In the last level of the SL we have set the way of model experiment, collected statistics and the representation of simulation outcomes. The number of model runs depending on simulation conditions can be either set during model creation or defined during the actual experiment with the help of the automatic stopping procedure. The research process is also possible with the definition of regeneration conditions. It is possible to visually follow the model run in the actual process as well as outside the JESS graphics environment. Also in this level the transformation of experiment plan templates to the code in JAVA is carried out.

The outcomes of simulation can be represented as tables that contain the expectation of estimated characteristic, dispersion, coefficient of a variation; as graphs of dependence of the estimated characteristics in relation to time or to a given parameter; or as the histograms.

7 Implementation of JESS internet-properties

As it has been mentioned above, JESS allows to transform aggregates and models constructed in a graphics editor into the files with the code in JAVA programming language. These files can be compiled by call of the external JAVA compiler in JESS environment, packed in JAVA archives, and placed on the server for common use. Using JESS, it is also possible to generate HTML-files that contain references to archive files stored on the server along with the descriptions of aggregates and models that are included in these archives. Above that, a user can view the graphical structure of an aggregate with the help of an Internet browser because appropriate applets for each aggregate and model from archive are generated. The task of the applet is to present the structure of an aggregate or a model. Fig. 3 shows the structure of a generator aggregate displayed in a browser.


JESS, Genarator aggregate - Netscape
File Edit View Go Communicator Help
Bookmarks Location: file:///F|/!/Generator.htm
What's Related
P1
P2
P3
T2
P1
Document: Done
Structure of the Generator Aggregate

List of Global Variables

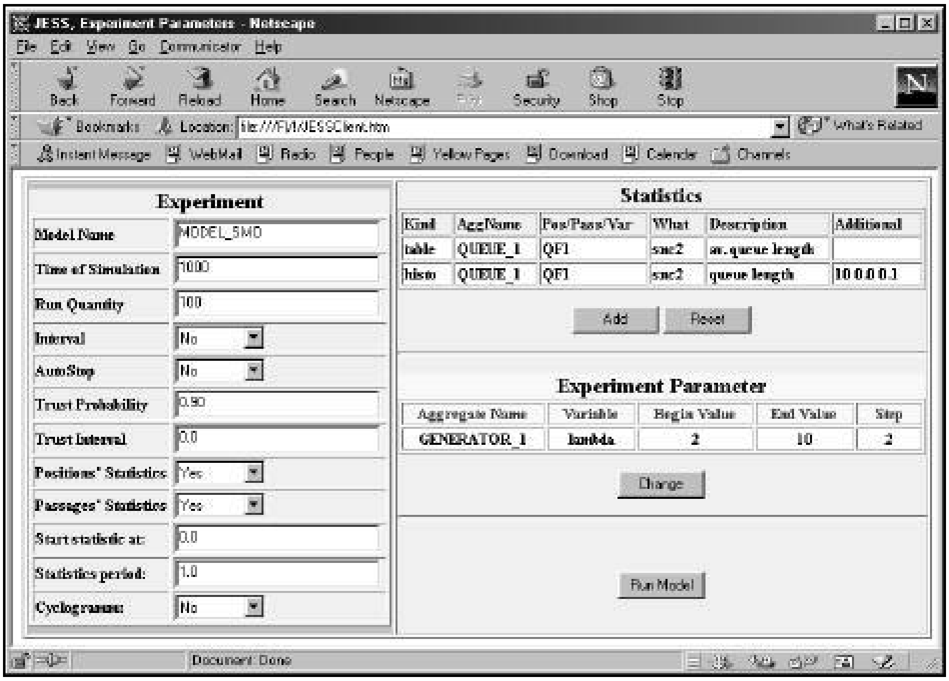
lambda - intensity of an entrance stream

type: double; begin value: 0.0

Fig. 3: Representation of a generator aggregate in Netscape Communicator

Thus, in local environment the user can get classes for aggregates and models from the network, load them in JESS, and use for further work.
In addition, JESS allows to locate aggregate and model classes on the enterprise server and to start simulation from any point of the network by setting-up aggregate and model components registered on the server along with the definition of the experiment plan.
JESS uses the EJB technology to create reusable components executed on the server and offers the possibility to generate files with aggregates and models EJB-components descriptions and files with their deployment descriptors. After these files are generated

and compilation of EJB-component files is executed with external compiler, we may deploy them on any server compatible with the J2EE specification. JESS also allows to execute a model outside its environment without animation.
Any kind of an Internet browser can be used for model setting and execution as well as for the review of simulation outcomes. Fig. 4 shows the view of HTML-page with the experiment settings.


JESS, Experiment Parameters - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: file:///F|/1/JESSClient.htm What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Add Reset
Change
Run Model
Document: Done

Experiment

Model Name	MODEL_SMO
Time of Simulation	1000
Run Quantity	100
Interval	No
AutoStop	No
Trust Probability	0.90
Trust Interval	0.0
Positions' Statistics	Yes
Passages' Statistics	Yes
Start statistic at:	0.0
Statistics period:	1.0
Cyclogramm:	No

Statistics

Kind	AggName	Pos/Pass/Var	What	Description	Additional
table	QUEUE_1	QF1	snc2	av. queue length	
histo	QUEUE_1	QF1	snc2	queue length	10 0.0 0.1

Experiment Parameter

Aggregate Name	Variable	Begin Value	End Value	Step
GENERATOR_1	lambda	2	10	2

Fig. 4: Remote settings of the experiment plan

It is possible to receive access to the Internet-compatible simulation components allocated on the server both from inside and outside the enterprise. The users can use these components or reset them to implement their own urgent tasks. Internal developers can update components and save them in enterprise models repository for further use. The Internet can serve for models sharing which will allow both the enterprise and its clients to work collaborate in improving the system characteristics. Users can share their models between internal and external clients who are able to view and execute the models through the network using any JAVA-compatible browser and any hardware platform.

8 Conclusions

JESS provides continuity of a simulation cycle using a uniform environment. Models of complex systems are described in a convenient visual form of the graphical

constructions. Technological and business processes modeling, enterprise modeling, control systems modeling etc. are the examples of JESS application. The use of the uniform formal method provides an invariant approach of system simulation for any system independently of its data domain.
JESS was used in a ship controlling system simulation [8]. The developed model contains 11 aggregates with about 10-40 transitions in each aggregate.
Further system perfection in the distributed execution of the simulation models is possible. Undoubtedly, it will influence model productivity. To investigate this influence, several experiments were carried out for local and distributed program models. In addition to this, the distributed program model was also studied in the local environment. In this case, the model's aggregates were executed as separate processes on the same computer [9]. As a result, we can draw the following conclusions:

1. Time of simulation almost linearly depends on the number of model transitions.
2. The execution of the distributed model in a local environment causes productivity losses compared to the local model.
3. The distributed model has an advantage in productivity over a larger dimension model that contains more than 35 transitions in one aggregate.

Subsequent system perfection also is possible in multi-agent technology use for formal definition of aggregates, transitions and formalism extension on the interaction level.

References

1. Fishwick, P.: Web-Based Simulation: Some Personal Observations. In 1996 Winter Simulation Conference, 1996, San Diego, CA, pp. 772-779.
2. Kilgore, R.; Healy, K.: Java, Enterprise Simulation and the Silk Simulation Language. In Proceedings of the 1998 International Conference on Web-Based Modeling & Simulation, ed. P.Fishwick, D. Hill, and R. Smith. SCS, San Diego CA, 1998, pp.195-201.
3. Kuljis, J.; Paul R. J.: A Review of Web based Simulation: whither we wander? In Proceedings of the 2000 Winter Simulation Conference. ed. J.A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, pp. 1872-1881.
4. Howell, F.; McNab. R.: 2000. Simjava. Available on <http://www.dcs.ed.ac.uk/home/hase/simjava/index.html>.
5. Nutt, G.: Evaluation Nets for Computer Systems Performance Analysis. FJCC, AFIPS PRESS, 1972, Vol. 41, Pt. 1, pp. 279-286.
6. Buslenko, N.: Complex systems simulation. Science, Moscow, 1978. (in Russian)
7. Kazymyr, V.; Demshevska, N.: Formal Object-oriented Approach to Complex Systems Simulation. In Proceedings 1th Int. Scientific and Practical Conference on Programming UkrProg'98, Kiyv, 1998. Cybernetic center of NSA of Ukraine, Kiyv, 1998; pp. 593-598. (in Russian)
8. Demshevska, N.: The Simulation of the Ship Control System. In Visnik of Chernihiv State Techological University, Chernihiv, Ukraine, 1999; Vol. 9, pp. 153-159. (in Ukrainian)
9. Lytvynov, V.; Kazymyr, V.; Havsiyevych, I.:CORBA based Distributed Simulation System. Mathematical Machines and Systems, 2000; Vol. 2,3, pp.76-87. (in Russian)