Global Journal of Computer Science and Technology: A
Hardware & Computation
Volume 17 Issue 1 Version 1.0 Year 2017
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Tool Support of Developing Systems Simulation with Active Elements

By Igor N. Skopin

Abstract- In this paper we propose and discuss a new approach to the simulation of developing systems is proposed and discussed. The base of the approach is idea of independent development of the so-called aspect models, whose interconnections are provided by computational environment toolkit. The main feature of the approach is the abandonment of postulate of the deterministic behavior of systems if they develop due the activity of event-driven elements behavior. We discuss their initial requirements for the tool support of the proposed approach.

Keywords: developing systems, active element, attributive representation, the aspect model, simulation, multi-aspect, multidimensional nature, multiple structures, tool support.

GJCST-A Classification: I.6.1, I.6.6, I.6.7

Tool Support of Developing Systems Simulation with Active Elements

Igor N. Skopin

Abstract- In this paper we propose and discuss a new approach to the simulation of developing systems is proposed and discussed. The base of the approach is idea of independent development of the so-called aspect models, whose interconnections are provided by computational environment toolkit. The main feature of the approach is the abandonment of postulate of the deterministic behavior of systems if they develop due the activity of event-driven elements behavior. We discuss their initial requirements for the tool support of the proposed approach.

Keywords: developing systems, active element, attributive representation, the aspect model, simulation, multi-aspect, multidimensional nature, multiple structures, tool support.

I. Introduction

The study of developing systems, whose behavior depends on the individual activity of their elements, is urgent for a wide range of research. From a practical point of view, it is important to have such control over the behavior of the system, which allows you to direct development to the right conditions for development and to limit undesirable ones. However, in most approaches to the study and design of systems, it is difficult to take into account the individual activity of the elements. This is due to the fact that the main focus of the developers is to search for a common deterministic development principle, whereas such principles are not reflected in the individual activities of the elements.

A problem in research into developing systems with active elements is the accounting for the behavior of elements not only in this system but also in other important aspects. This multi-aspect activity may contradict the object model of a system, and developers usually regard it as external non-controllable impact. Objectives of the behavior of an element rarely correspond to the system development objectives, and so it is difficult to reflect the involvement of the element in other activities in the model of the system. But this changes the activity of the element. Trying to avoid the complexity of multi-aspect properties one can build different models for different aspects. However, this way brings about no less complicated problem of adjoining models.

Author: Institute of Computational Mathematics and Mathematical Geophysics of SB RAS. e-mail: iskopin@gmail.com

The proposed approach is intended to overcome both a postulate of determinism and difficulties of multidimensional nature. One can consider it as a version of the discrete-event simulation proposed in the 1960s by J. Gordon [1]. In the most developed form it is presented in the programming methodology in Simula [2] and Simula 67 [3] languages. Unlike a traditional to use the global (linear) ordering of events in controlling a computation, we refuse from this determination that mostly admitted to optimize computer cost. We consider the support of autonomous modeling of aspects as another feature of our approach thus making it associated with the so-called aspect-oriented programming [4]. In this case, the difference is that following our proposals, one should provide the support of aspect features by a specialized design and computation environment as well as the pre-fixed representation of model elements.

Conceptually close to our approach is the development of the so-called multi-agent systems (MAS) [5]. One of the Russian researchers and developers in this area P.O. Sobolev says: "The key element of these systems is a program agent that can perceive the situation, to make decisions, and to communicate with other agents. These capabilities dramatically distinguish MAS from existing rigidly organized systems providing them with such an important new feature as self-organization. In this case, separate parts of the program can agree on how a problem should be solved. The parts acquire their activity and can initiate a dialogue with the user in advance at not prescribed times. They can work in conditions of uncertainty and offer clarification and reformulation of tasks, etc." [6]. The abandonment to find a deterministic principle, which allows identification of the best solution, the agent's activity with their purposeful behavior, the focus on operations with developing systems are a common concept for both approaches.

However, outside the multi-agent approach, there remains a problem of supporting the mutual influence aspects. The information affecting the behavior of an agent came from its environment and transferred for the use by other agents. But this is not an independent construction of mutually depended aspect models supported by tools. We declare this position in our approach as one of the main requirements. At the same time, methods of constructing multi-agent systems are applicable in our case, especially in defining the behavior of agents. For our approach, there is a very valuable information about organizing the functioning of the agent computing environment and support toolkit (an example of this kind of a system is REPAST [7]).

In the first part of this paper, we present characteristic features of the approach and show how one can attain the declared capacities. The second part discusses the requirements for the toolkit support of the approach. The desired tool should be appropriate to develop and to perform simulations of models, data collection, storage, and processing of information in the course of studying a system.

II. Characteristic Features of the Approach

a) *Model System, Simulation, and Events*

We are speaking about a *model of the system* as an abstract notion, without being tied to any meaningful interpretation. The system consists of elements whose actions change some characteristics of the elements themselves and the system as a whole. This is the behavior of the real system. In the model system, we represent these characteristics as *attributes of elements*. Their change in the course of calculation one can interpret as the *model simulation of the system behavior* or *functioning of the system*.

A value of element attribute is modified only as a result of program action execution if the operational context of a program includes a mutable attribute. We postulate that all actions are methods of elements (in the sense of object-oriented programming [8]), and operational contexts form from all available elements and their attributes. The action-method which carries out an element is the manifestation of its activity in the simulation system through the model calculations. We do not impose restrictions on how described the programs of element actions, and the only thing that we require is a single *unified attributive representation of all elements* as common basis of description of action.

Elements can be *linked*, which is understood as the presence of *binary relationship* (of any nature), reflecting the relations in the real system. Due to the links a set of all elements of the model system receives the network structure. The links are a kind of attributes: they allow the element methods to access to the attributes associated with this element. This provides the opportunity to use and modify the attribute values (including links) and to influence the actions of the elements available by the links. An element can *give rise* to other ones, associated with the parent element, *add relationship ties*, or *remove* them according to requirements of modeling. It may deny all links of an element, that one can interpret as destruction of the latter. All these activities develop a network of the model system.

It is convenient to assume that there is allotted element for indicating the system as a whole. It has the especial attributive representation which allows one to distinguish between the external impacts and changes of attributes of the system. The allotted element receives ties with all other elements of the system. It is not necessary that allotted element had its prototype in a real system.

Any action of the model system runs as a response to some event that occurs during the simulation. The event-driven mechanism, used in the approach proposed, reflects Hoare's conception [9], according to which any change in the model system is an event if there are elements that *recognize* it. If this is true for such an element, one of its actions executes. Execution of this action is called the *reaction* to the event.

The types of all events are determined in advance, but a set of event types to which the element can react is formed in the dynamics of calculations (while carrying out this or other element action). A current set of the types of recognizable events is called *status* of the element. We consider that the status of the element is set as a particular attribute that coding the set of the types of events. Knowing the value of this attribute, one can always find all types of the events that the element can recognize. In the correctly constructed attributive representation opposite is truth too. If this condition fails, then it is deemed to have violated the integrity of the attributive representation of the element. Response to a joint event that is recognized by several elements is executed *jointly* and *asynchronously*. This allows demarcating parallel execution of actions. They execute as usual parallel processes in own threads. Processes management of threads uses conventional means. They coordinate several reactions of elements. Following Hoare [9], we consider that the recognition of an event is an instant action, but in the reaction execution, it is possible that other events may occur to which the element should respond.

In the event-driven mechanism we can define a *protocol of element behavior* as a sequence of triples: (<element status>, <event to which it responds>, <element reaction>). In such a notion, we define the *behavior of the model system* as a collection of all protocols of elements glued on the joint recognized events. This set determines a partial order on all the events occurred, where each included protocol is a linear chain. It is natural to consider a protocol of an element (more precisely, a sequence of its events) as element local time. In this case, we offer the mentioned partial order on all events to consider as a correctly determined *global time of the model system*. Here, by correctness we mean the most accurate information about relations for events *earlier*, *later*, *unknown when*. In the concept of model time there is no inaccuracy, which is typical for the commonly used time. Instead of the desired accuracy, the temporal relations are usually determined arbitrarily. We do not exclude the possibility of such a refinement, but we always insist that simulation of developing systems requires an accurate definition of the concept of time. Details on the local and global time one can found in [10]. Note that our understanding of time differs from that used in discrete events and multi-agent systems.

b) Multi-aspects, Structure of a System, and Attribute Representation of Elements

The behavior of elements in a real system involves the activities in various aspects of functioning. On the model level describe this, one can build models of all important aspects. They are the so-called *aspect models*. However, because of the problems of harmonizing the aspect models, developers are often limited to building a common model of the system. In our approach, we offer the support for independent aspect modeling and dynamics of mutual influence of aspect models in the simulation calculations that are common to all or several of these models.

The aspect model defines the structure of a set of elements, and their relations called the *aspect structure*. It is a part of the model system network. The model system We consider the modeling system as the union of all its aspect structures. When constructing aspect models is provided a joint and equitable co-existence of aspect structures (we call this feature *multiplicity of system structures* [11]), it is possible to co-operate all or selected aspect models (including one model). To ensure such an operation with aspect structures, we consider it necessary to use a standardized format for attributive representation of elements. This format provides for splitting the representation into blocks related to each of the aspect structures. These blocks are called the *aggregate of aspect attributes*. Each aggregate contains the data assigned to this structure, current status, as well as methods of aspect actions. The combination of all aggregates contains all information about any of the aspect models.

The necessity to distinguish between the aggregate of attributes related to different aspects follows from requirements to the standardized attribute representation of elements. These requirements also mean that the activity of an element is an aspect property: ability to affect the system as a whole through the actions related to an aspect. In our approach, the construction of aspect models takes into account the fact that a user operating on some element has access to others only through references to them, and access to attributes of other aspects is allowed for the type of this element. This limitation leads to independence of constructing each aspect model from its environment.

The event generation is global in the sense that a created event is accessible for the response of elements in any aspect models. The recognition of an event and the response of an element to it are local, i.e. they are represented in a certain aspect model. It is possible that the element response to one event in several models, but this means only the occurrence of several independent reactions.

c) Horizontal and Vertical Modeling of a System

As a model of a system, we consider a set of all aspect models whose joint simulation forms the characteristics of a system represented among the attributes of an indicating element. To test various hypotheses concerning the control of the simulated system, one can use the truncated joint simulation. In this case, he chooses for calculations not all possible models and establishes the lack of influence on their behavior through external actions. This use of aspect models is called *horizontal modeling of the system*.

From the standpoint of horizontal modeling, implementing the behavior of all aspect models is not necessary. For all unrealized aspect models, you need to present elements that replace missing implementations. Each such element must ensure the generation of events adequate to the required behavior of the aspect model. Thus, the replacement element will create conditions for the functioning of other aspect models. It would be naive to expect that the model of the system, constructed using replacement elements for aspects, will be adequate to the real simulated processes. But due to the addition of the replacement elements with programs reflecting real aspect behavior, the quality of modeling will grow. Note that in this way it is possible to build various versions of aspect programs, and, as a consequence, to test different versions of modeling the real system for selecting the best of them.
In particular, when modeling aspects, the developer can use concepts that are different from the agreements of our approach. In this case, she/he must ensure that the proposed development of the model system is coordinated only at the interface level. Otherwise, it will be necessary to ascertain the conceptual incompatibility of approaches. In fulfilling this condition, one can pose the problem of replacement of aspectual models with real data from external sources. This brings our concept to the level of real-time systems for making management decisions. Leaving out details, let us note that the widespread use of a similar approach in practical applications of multi-agent systems (see, for instance, [6]) indicates to its effectiveness in our case as well.

The element representing the model of the system as a whole differs from the other elements only in that it alone interacts with the external environment of the model. The environment affects the system, generating corresponding events, to which only it responds, and receives information about the model behavior only from it. We can consider the environment of our model as a some supersystem in which our system is an integral part. And for this additional modeling, the element representing our system, is a usual an active element in its horizontal model, i.e., in the new *model of a higher level*. To meet different needs, one can build different supersystems, which in turn, one can use as active elements of models of even higher levels. Thus, the multilevel modeling can be a source of aggregated data for the hierarchical control in various areas. Such a kind of modeling is called *vertical*. As in the case of the horizontal modeling, this approach allows for the replacement of models with real data.

d) Groups of Elements, Subsystems, and Operation with Groups

One can distinguish *meaningfully related groups of elements* in the model system, which sometimes it is convenient to consider as independent ones. Sometimes one can consider a set of attribute descriptions of the elements of the group as the model of a subsystem, but it is not quite accurate: a system (and a subsystem) is not reduced to a sum of its parts. So, it is needed to speak about own attributes of a subsystem, i.e., the subsystem receives the status of the element affecting the behavior of the system as a whole. This formal status we may regrade as a definition: an element indicating a group (i.e., it is linked with all the group elements by a specifically prescribed relation, called *grouping*) and all elements of the group are called *subsystem*. The grouping relation is dynamic because the element belonging to a group, as well as its connection with the element indicating the subsystem can vary.

Attributes of a subsystem are formed both by their behavior (actions of its elements) and under the influence of the behavior of other elements that provide an external information for the subsystem. In particular, the links occurring in a subsystem as relations with external elements are set for the subsystem as a whole, and it can re-direct them to their elements. At the same time, this does not exclude the possibility of independent recognition of events using its elements.
On a set, consisting of groups, including also the groups composed of all the elements of each aspect structure we define *set-theoretic operators*: intersection, union and supplement. Their results can be considered as groups. They are sometimes interpreted in terms of their behavior.

We formally introduce the operator called *convolution of subsystem*. Its performance is the replacement of the interaction of the subsystem elements with the environment by the interaction of element indicating the group (subsystem). In this case generation of events to them are transferred from the subsystem elements to the element indicating the subsystem. A convolution subsystem can be considered as a *black box*, whose *inputs* provide information produced by the subsystem reactions to events and *outputs* are presented by generation of events from the subsystem, providing information for the environment. But such a consideration is not quite correct: every subsystem, being formed in some aspect, cannot screen from its elements belonging to other aspect models.

Convolution can be used in the top down construction of models, starting with the upper level, gradually refining the models by revealing the subsystem-elements structure. It fits for localization of models that are beyond our approach developed, but adequate for certain processes and can be coordinated with events of the system. However, it is should be noted that convolution is applicable only in the cases when there is not information about the individual behavior of the subsystem elements.

III. Initial Requirements of Tools

The approach proposed to studying or developing systems is aimed at designing the supporting toolkit. We present it as a software package, whose facilities are referred to the three categories:

- Ensuring the dynamic simulation;
- Control of simulation, collection and processing of simulation results;
- Facilities for models development.

The simulation of every aspect model can be implemented with the use of a variety of algorithm, including external ones for our toolkit. Therefore, it is necessary that our toolkit be an *open software system*. Only incompatibilities of interfaces can prevent the use of external computational models. This is a common requirement for all the three categories.

a) Ensuring the Dynamic Simulation

Active elements and the autonomy of their behavior in each aspect model, as well as the mutual effect of models require that each element should be defined as a separate computational process that is running in a separate thread. Each process has a local memory, which is filled up with data of attributive representations of the element. In our approach the event-driven technique does not require the shared memory of different processes. If a shared memory is necessary, one can provide auxiliary elements, whose processes are responsible for the delivery of information to basic processes on demand. Of course, this does not exclude the possibility of optimizing the main mechanism on the system level: for statically computed cases of operations with events, it is possible to replace reactions by a direct call of the respective methods (this optimization technique is discussed in Section 3.5 of [12]).

The event-driven communication of processes (methods of elements) ensures the transfer of information between processes through the *messages* associated with events. The messages contain data formed by the elements that generate the events. Using messages, it is possible to transmit specific data as well as links to the attributive representations of various elements. Thus, there is a possibility of the direct effects of elements on each other. Such effects, generally speaking, may be incorrect if one does not take care of matching synchronization. On the attributive representations level there should be an access system enabling/disabling flags, dynamic priorities, and other well known features to support the matching of communicating processes [9].

Thus, the parallel and asynchronous execution of processes of elements of the model is realized, if necessary, they coordinate and synchronize the joint reactions controlled by a conventional technique of parallel computing. A specific feature of the simulation requires that the number of threads involved in the simulation, be sufficiently large. Therefore, our support toolkit requires of the modern high performance hardware architecture. On the other hand, at least for the first versions of the system, whose one of the main objectives should be working out the methodology of modeling, it is reasonable to choose an existing universal system with parallel computing as the technical and program environment of the simulation. So, it is sensible to develop our toolkit as a specific amplification of the chosen system. Promising in this regard is the system [13], whose users are provided with the opportunity to adapt algorithms to specific features.

b) Control of Simulation, Collection and Processing of Simulation Results

The control of simulations is used to give the user feature to influence the calculations in order understanding the significance of model factors. Such control is considered as a part of an event-driven technique. We establish that all elements recognize a special event of simulation named as *pause* and may response to it, maintaining their status for re-starting the activity in the future. A pause allows one to receive the current information about calculation, to change some parameters of models, to add or to delete elements and to specify their relationships. At the time of pause, to all elements is assigned a special state in which they respond to a single event named as *resumption of modeling*.

Through a series of pauses, *monitoring of system behavior* can be arranged, i.e. user can obtain information about changes in the structure, build of element protocols, and collect integral characteristics. This information may be used for the results processing. For comparison of the simulation versions options for saving the suspended configurations of the simulated system in the repository are provided. Standard facilities of version control (see, for example, [14]) are supplemented with options selection according to integral characteristics.

There is no need to develop special means for the simulation results processing, because today the software market offers a variety of advanced products for this purpose (see, for example, [15]). A good solution is to focus on the features associated with potentiality of the above mentioned support system of parallel programming [13].

c) Facilities for Models Development

The proposed toolkit should provide *tools for building and editing models*. First, there are editors for adding and deleting elements of the model, modifying their attributes, etc. Editing should be followed by validation. In preparing the initial configuration of a model system or its modification during a pause, as well as in the course of the simulation calculation, we need *integrity control of models*, which prohibits an incorrect configuration of simulation. In particular, the initial configuration should be realistic. If an algorithmic verification of the tolerance of simulation is not possible, we provide the user control of configurations.

Modeling and support of simulation are based on appropriate *means to visualize structures*. Different structures of a set of elements should be coordinated. In the model design and monitoring of simulations, an element should be shown jointly in all the structures. The choice between the removal and preservation of elements in different structures should be known for the correct decision-making. Structures of different aspect models should be highlighted in different colors. Visualization of a modeling process must support the correctness of constructions.

A set of development tools to support different strategies for modeling should support different strategies of design. In particular, we offer the support of *bottom-up* and *top-down modeling strategies*.

In the bottom-up strategy, building of each aspect model is started from construction of a primary set of basic elements. Each type of elements is provided with a certain attribute representation. At first, methods of element actions are given as dummy routines. Later on should be refined in developing the aspect model. Specifying the links is needed to determine their properties as relations and permissions of an access of element to other ones. In this case it may be required to add new types of elements, for instance, in forming subsystems.

In the top-down strategy, development of an aspect model begins with subsystems and their elements indicating the subsystems. The subsystems are supplemented with new basic elements or a set of already-prepared elements. The new or refined existing elements are described in the same manner as just as in the bottom-up strategy.

In both cases, the development of models should be supported by tools of tracing the life cycles of elements, reflecting the degree of readiness of their attribute representation. The life cycle of an element class is considered complete when the path from the original class to its readiness to use to generate the elements is traversed, and all conditions for the correctness of the descriptions of each aspect model are provided.

After the aspect models are constructed, are coordinated the influences of the aspect actions on the behavior of other models. This process can result to the identification of classes, refinement of the behaviors of aspect models, and other modifications of the descriptions of classes, that aimed at ensuring the correctness of the horizontal model of the system.

The process of designing and using model systems should be provided by system-wide means of supporting development, as well as managing model calculations. Discussion of this aspect of the proposed toolkit is beyond the scope of this paper. Nevertheless, it should be noted the main requirement for a set of developer tools, without which the expectation of the advantages of using the concept presented above will not justify itself. This is the fullness and integrity of support for user activity. The requirement concerns both the functionality of the software system and its interface: it should be ensured adequate reflection in the interface of all the features associated with the creation and using of models. The problems associated with the requirement of interface completeness and the possibilities for their solution were discussed in [16]. In [17] we proposed a technique that can be used to create interfaces that meet the requirement of fullness and integrity.

IV. Conclusion

The proposed approach for studying developing systems is aimed at supporting the development of models and fulfillment of a series of simulations for obtaining information useful in study the real systems. The work related with this processes is very laborious. It requires analysis of input data and different variants of system behavior, the identification of significant aspects and the construction of aspect models. So we propose the approach to creation of a tool support to ensure the development of models, carrying out calculations, performance control, management of simulations, collecting and processing of results. Depending on the purpose of such tools, its architecture and modeling support capabilities can be very different. Therefore, we confine ourselves to discussing only key concepts. Nevertheless, it is appropriate to note a number of aspects of developing the toolkit to support the development of models, as a software system, the quality of implementation of which essentially affects its usefulness. These issues are discussed in detail in [18], where, in particular, we point out the need for instrumental support for the development and use at all stages of setting and solving the problem of mathematical modeling.

In this paper, we did not dwell on issues of the input and output of information, and only in passing discussed outlined the mechanism for monitoring the simulation pauses. These are important points for the real use of model complexes and they should be solved at the next stage of design. In principle, these problems are solvable, but it seems reasonable to give concrete proposals for these issues based on a detailed analysis of subsequent further requirements.

We should mention the issues of computational complexity of simulation, which are associated with our approach as applied to practical tasks. Multithread execution of the elements actions, event-driven technique, etc. require high performance computing. In this regard, our project should incorporate the use of existing general means as well as development of special facilities of the architecture-dependent optimization. To improve the performance of the simulation, the adaptive software platform should be used. Although this task is beyond the scope of our discussion, we note that it must be solved in a more general context, implying the organization of parallel computations. In this connection, the use of the above-mentioned system [13] seems a reasonable solution.

Acknowledgment

The article presents the work that is supported by the grant RF RSF № 14 11 00485 "High-performance methods and technologies of electro physical processes and devices modeling".

References Références Referencias

1. T. J. Schriber. *Simulation using GPSS*. — New York, Wiley. 1974.
2. O.J. Dahl and K. Nygaard. *SIMULA – A language for programming and description of discrete event systems. Introduction and user's manual*. NCC, Sept. 1967.
3. O.J. Dahl, B. Myhrhaug and K. Nygaard. *SIMULA 67 Common Base Language*. NCC, May 1968.
4. G. Kiczales, O. Lamping, A. Mendhekar and other. *Aspect-oriented programming* http://citeseerx.ist.psu.edu/viewdoc/similar?doi=10.1.1.115.8660&type=ab
5. M. Wooldridge, *An Introduction to Multi-Agent Systems*. — John Wiley & Sons Ltd, 2002, paperback, 366 pages, ISBN 0-471-49691-X.

6. P.O. Skobelev. *Holistic approach to the creation of open multi-agent systems* — Proceedings of the 3rd International Conference on control and modeling of complex systems, Samara: SSN RAS, 2001, p. 147 – 160 (in Russian).
7. REPAST: *Recursive Porous Agent Simulation Toolkit*. http://repast.sourceforge.net/.
8. G. Booch. *Object-Oriented Analysis and Design with Applications*. Addison-Wesley. 2007. ISBN 0-201-89551-X
9. C. A. R. Hoare. *Communicating Sequential Processes*. — Prentice-Hall International. Englewood Cliffs, New Jersey, 1985.
10. I.N. Skopin. *Local and global time in modeling developing systems*. — In Proceedings of the 7th Ershov International Conference "Perspectives of System Informatics", Workshop on Science Intensive Applied Software. Novosibirsk, 2009, p. 254–259 (in Russian).
11. I.N. Skopin. *Hierarchy and developing simulation systems*. In Problems of System Informatics. Novosibirsk: Ltd. "Siberian Scientific Publishers", 2010. — p. 188-214 (in Russian).
12. N.N. Nepeivoda, I.N. Skopin. *The Foundation of Programming*. — Moscow, Izhevsk: Institute of Computer Sciences Reseaching, 2003 (in Russian).
13. V.Malyshkin. *Assembling of Parallel Programs for Large Scale Numerical Modeling*. – In the Handbook of Research on Scalable Computing Technologies. IGI Global, USA, 2010, 1021 pp, Chapter 13, pp. 295 – 311. ISBN 978-1-60566-661-7
14. Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato *Version Control with Subversion.Next Generation Open Source Version Control*. — O'Reilly Media, 2004, 320 p.
15. S.V. Maklakov C.B. *BPwin and ERwin CASE-tools of developing information systems*. — Moscow: Dialog-MIPR, 2001. — 340 c.
16. I.N. Skopin. *Development of software systems interfaces*. — In System Informatics, issue 6 "The problems of architecture, analysis and software development". Novosibirsk: Nauka, 1997, p. 34-96 (in Russian).
17. I.N. Skopin. *An approach to the design of application interfaces*. — In joint issue of the book "Computing technology" v. 20 and "Bulletin of Al-Farabi KazNU. Series Mathematics, Mechanics, Computer Science" № 3 (86), 2015. — Novosibirsk, Almaty. — p. 332 – 345. ISSN 1560-7534, 1563-0285 (in Russian).
18. V.P. Il'in, I.N. Skopin. *About Performance and Intellectuality of Supercomputer Modeling*. - Programming and Computer Software, 2016, Vol. 42, No. 1, pp. 5–16.