ATLANTIS PRESS

Advances in Intelligent Systems Research, volume 166

7th Scientific Conference on Information Technologies for Intelligent Decision Making Support (ITIDS 2019)

The Agent-Based Modeling Method For The Study Of Unique Mechanical Systems

Olga Nikolaychuk
Matrosov Institute for Systems Dynamics and Control Theory of Siberian Branch of the Russian Academy of Sciences
Irkutsk, Russia
nikoly@icc.ru

Pavlov Alexander
Matrosov Institute for Systems Dynamics and Control Theory of Siberian Branch of the Russian Academy of Sciences
Irkutsk, Russia
asd@icc.ru

Stolbov Alexander
Matrosov Institute for Systems Dynamics and Control Theory of Siberian Branch of the Russian Academy of Sciences
Irkutsk, Russia
stolboff@icc.ru

***Abstract*—The article discusses the MDD related agent-based modeling method that provides a four-level representation of the model elements throughout the design process. In the proposed method it is suggested to separate the description of the agent-based simulation model (ABSM) structure and behavior into domain independent and domain-specific parts. It means that there are no predefined structures of ABSM and the experts with appropriate knowledge and skills for some specific problem areas can create them. The application of the proposed method for contingency management in critical infrastructures is shown on the illustrative example that describes ABSM development process for solving problems of the unique mechanical systems states identification and forecasting.**

Keywords—Agent-Based Modeling, Model-Driven Development, Knowledge-Based System

I. INTRODUCTION

The multi-agent approach utilizes the idea of agents – active problem-solving entities coordinating with each other to provide solutions to the challenges that can't be handled by their own individual capabilities. Currently, the application of the multi-agent approach as a methodology for solving various complex problems is very popular and covers a lot of different subject domains (logistics, financial, social, et.al.). The rapid development of information and communication technologies over the past 20 years and the gradual experience accumulation in the application of the multi-agent approach are the strong contributing factors. Nowadays, the number of software tools designed to develop Multi-Agent Systems (MAS) and Agent-Based Simulation Models (ABSM), as a special case, is measured in dozens. Among all this diversity it can be highlighted the most popular means (Anylogic, JADE, Repast, Netlogo, Swarm, etc.), but still their number is large enough. In this situation, the relevant problem is the extension of scope for existing software tools application via developing corresponding design methodologies. In the current research to address that problem, it is suggested to create more abstract mechanisms for constructing ABSM that make it possible to reuse and combine a set of Agent-Based Modeling and Simulation (ABMS) techniques by a formal description of the basic elements and operations of methodologies and tools utilized in the design and implementation phases of ABMS process.

In studies related to MAS, such formal descriptions of methodologies have already been introduced. Their elements can be found in the descriptions of methodologies of agent-oriented software engineering. More explicitly, these formal descriptions can be observed in studies aimed at combining and generalizing the design methodologies [1, 2, 3, 4 ,5]. So the standard structures of MAS and ABSM have been surveyed in sufficient detail, but the additional refinement of information models covering all phases of ABSM design process is required, since the main results described in the publications are obtained for MAS, and the specificity of simulation modeling is not properly considered. There is some promising research [6, 7] in this direction and the current paper also makes its own contribution.

As the basic paradigm for describing ABSM process Model Driven Development (MDD) approach was chosen. MDD is an approach that involves the creation of meta-models and tools for their processing, on the basis of which particular models describing the problem situation are constructed. In MDD case, the creation of software is carried out automatically without the participation of the developer. As software engineering methodology MDD is the effective way deal with the complexity of automation of ABMS process because particular ABSM can be considered like a computer program with special functionality.

In the context of ABMS process, the application of the MDD approach involves the development of a multi-level system of meta-models containing information about the structure and behavior of ABSM elements, including both definitions related to ABSM methodology and the specification of the particular ABSM including subject domain characteristics.

Under these considerations, the system of meta-models, their transformations, and related tools are suggested in the paper. The overview of ABMS process basic steps in terms of proposed meta-models and tools is also presented. In the final section an illustrative example the development process of agent-based simulation model for solving identification and prediction of the unique mechanical system technical state problems is considered.

II. APPLICATION OF MDD APPROACH TO ABMS PROCESS

A. Metamodels Architecture for ABSM Development

Following the principles of MDD approach, where the bottom-level model (M0) corresponds to some runtime model or program while top-level one defines the most abstract elements for lower level models specification, it is proposed four-level metamodels architecture M (table 1).

Each level of the hierarchy is defined regarding to ABMS process whereas models related to these levels can be further distinguished by the type of containing information: structural, behavioral or combined.

Here M0 level relates to concreate ABSM in runtime; M1 level is specification of particular ABSM; M2 specification of ABMS methodology; M3 – most abstract specification for lower levels.

TABLE I. METAMODELS ARCHITECTURE FOR AGENT-BASED MODELS DEVELOPMENT

Level	Metamodel Type		
	Structure	***Behavior***	***Combined***
M3	M^{Ont}	M^{KB}, M^{Op}	–
M2	M^{St-A}	M^{Op-I}, M^{Op-A}	M^{A}
M1	M^{Ont}, M^{St-A-D}	M^{KB-D}	M^{A-D}, M^{A-D-I}
M0	–	–	codI

The following is a brief description of suggested set of meta-models:

- M^{Ont} is metametamodel ontology;
- M^{KB} is a meta-model describing rule-based knowledge base in terms of M^{Ont};
- M^{Op} is meta-model for describing algorithms specifications;
- M^{Op-I} is meta-model for describing methods of the ABSM implementation tools like simulators, reasoning and visualization engines on high abstraction level in terms of M^{Op};
- M^{St-A} is a meta-model of the ABSM structural elements (e.g., agents, environment, events, et al.) independent from subject domain and expressed in terms of M^{Ont};
- M^{Op-A} is a meta-model of the ABSM behavior independent from subject domain;
- M^{A} is a subject-independent agent-oriented meta-model defining some ABMS methodology, including its structural and behavioral aspects;
- M^{Ont-D} is a subject domain ontology in terms of M^{Ont};
- M^{KB-D} is a particular knowledge base that describes the behavior of subject domain objects;
- M^{St-A-D} is a structure of the ABSM for a particular subject domain area;
- M^{A-D} is an ABSM specification related to particular subject domain;
- M^{A-D-I} is M^{A-D} containing particular initial condition;
- *codI* is an execution of the M^{A-D-I} by the ABSM specification interpreter.

M^{Ont} and M^{KB} have the standard structure related to the type of information they describe. M^{Ont} contains a set of widely known primitives: concept, attribute, relation and concept instance. M^{KB} defines such elements as template, rules, different types of rule condition and rule action including performing calls to externally designed procedures or services.

The proposed metamodels architecture is developed with the purpose to divide information related to the subject domain (M^{Ont-D} and M^{KB-D}) and ABMS methodology (M^{St-A} and M^{Op-A}). So M^{Ont-D} and M^{St-A} models can be designed independently from each other and then composed together to obtain the particular structure of ABSM (M^{St-A-D} model).

It is supposed that creating the behavior of subject domain objects in the form of M^{KB-D} is similar to the ways of creating knowledge bases in many other applied systems with one worth to mention feature: elements of M^{Ont-D} can be explicitly utilized to create elements of M^{KB-D} in terms of M^{KB}. To represent behavioral part that is independent from particular domain and related to some ABMS methodology we need a structures to formalize a set of agent-oriented operations (algorithms of agent and environment lifecycles, descriptions of different types of actions/effectors, et al.) in unified and abstract way. To solve this problem M^{Op}, M^{Op-I} and M^{Op-A} metamodels are introduced and would be discussed in more details in the next section.

B. ABSM Implementation Tools Metamodeling

To express the behavior of the ABSM on the top of M^{St-A}, M^{Op-I} and M^{Op-A} metamodels in the subject domain independent way we need to explicitly describe tools, components and methods that are traditionally used for creating ABSM. To create the top level specification (M^{Op}) the wieldy known workflow approach can be utilized [8, 9], where behavioral aspects of ABMS methodology is expressed as a sequence of operations glued together by the predefined set of operators. In the course of the current research M^{Op} has rather simple structure (see below) but it can be improved and extended in the future.

$$M^{Op}=<Op^{B}, Op^{C}>$$

$$<Op^{B}>=<Location> <Name> \{<Parameter>\}$$

$$<Parameter>=<Name><In |Out><Type of parameter>$$

$$<Type\ of\ parameter>= <Literal>|<Concept>$$

$$<Op^{C}>=<Name> \{<Parameter>\}$$

$$\{<Op^{B}>|<Op^{C}>|<Operator>\{<Op^{B}>|<Op^{C}>\}\}$$

$$<Operator>= O_{if} \mid O_{L}| O_{B}$$

Here $<Op^{B}>$ is a description of basic operation that can be directly executed by ABSM specification interpreter; <Location> is an address of its implementation; <Literal> is a standard primitive data type (string, number, boolean, etc.); <Concept> is a combination of named literals and other concepts; $<Op^{C}>$ is a description of composite operation; O_{if} is IF operator; O_{L} is loop operator; O_{B} is the grouping operator.

To address the specific of ABMS process a set of runtime abstractions have to be represented in terms of M^{Op}. It is supposed that these abstractions have to be defined as

interfaces and must be implemented in ABSM specification interpreter.

$$M^{Op\text{-}I} = <I^{sim}, I^{inf}, I^{imp}, I^{run}>$$

Simulation Engine interface (I^{sim}) contains standard ABSM operations for sending broadcast and address notification with content in the form of concept or concept instance, and for supporting lifecycle of ABSM elements (create, launch, et al.).

Inference Engine interface (I^{inf}) provides operations for reasoning based on knowledge base described in the form of $M^{KB\text{-}D}$, as well as loading initial conditions and retrieving results in the form concept instance.

Imperative Engine interface (I^{imp}) describes standard tool for performing different kinds of the external calls, for example, provides broadcasting simulation data to remote clients. It is supposed that input and output procedure parameters are literals, concept or concept instance.

Runtime Controller interface (I^{run}) is designed to facilitate processing of ABSM during simulation process. For ABSM element Runtime Controller implementer can retrieve and modify data, retrieve and execute operation in the form of Op^B or Op^C.

C. A Set of Software Tools for ABSM Development

The design and implementation phases of ABMS process are maintained by a set of software tools for creating, editing and storing proposed system of metamodels, executing transformations of these metamodels, composing final ABSM specification and performing simulations.

Ontology editor (S^{Ont}) allows one designing a hierarchical structure of concepts from abstract to more concrete ones; the definition of the properties and relations; creation of concept instances. The description of relations is carried out by constructing a semantic network, where the vertices are concepts and the relationships are edges.

Operations editor (S^{Op}) is designed for creating agent-oriented operations ($M^{Op\text{-}A}$) in accordance with $M^{Op\text{-}I}$ specification. Visual representation of $M^{Op\text{-}A}$ is done via a semantic network, where the vertices are methods and their parameters, the edges show data and control flows.

TABLE II. THE RELATIONS BETWEEN THE METAMODELS AND THE SOFTWARE TOOLS

Software tool	Specification	Output
Ontology Editor	M^{Ont}	$M^{Ont\text{-}D}$, $M^{St\text{-}A}$
Operations Editor	M^{Op}, $M^{Op\text{-}I}$	$M^{Op\text{-}A}$
Knowledge Base Editor	M^{KB}	$M^{KB\text{-}D}$
ABSM editor	M^{Ont}, M^{KB}, M^{Op}	$M^{St\text{-}A\text{-}D}$, M^{A}, $M^{A\text{-}D}$
ABSM manager	M^{A}	$M^{A\text{-}D\text{-}I}$
ABSM interpreter	M^{A}, $M^{A\text{-}D\text{-}I}$	*codI*

Knowledge base editor (S^{KB}) has the following functionality: creating knowledge bases on top of the concepts and instances from $M^{Ont\text{-}D}$ and $M^{St\text{-}A\text{-}D}$; visual construction of the rules based on M^{KB} specification; creating initial conditions; generation of knowledge base code in Jess, Clips and Drools formats; registration externally defined methods that can be accessed via supported ways of communications.

ABSM editor (S^{A}) provides all necessary functionality to support $M^{St\text{-}A\text{-}D}$, M^{A}, $M^{A\text{-}D}$ meta-model creation, including invoking other editors and performing related trans-formations. Virtually ABSM editor realize main control flow of ABMS process and provide the entry point for interaction with users.

ABSM manager (S^{Mng}) has following main functions: setting the initial conditions for ABSM elements and simulation parameters; managing the simulation process (start, stop, pause); registration concrete components that must implement a proposed set of interface; registration components that can broadcast simulation data to external clients (for example, visualization of ABSM state).

ABSM specification interpreter (S^{codI}) provides specification execution of particular ABSM. Its modules must implement the set of interfaces defined in $M^{Op\text{-}I}$. Currently, Madkit [10] is utilized as simulation engine while Drools and Jess as the inference engine. The interpreter also contains visualization subsystem, that grants the remote access to the simulation data via standard web browsers.

The editors are implemented as a web application with GUI built on the top of HTML, CSS and JavaScript languages and popular JavaScript libraries such as jQuery and jQueryUI. The data storage functionality of the editors is implemented by well-known PostgreSQL, while communications is based on HTTP/HTTPS, SOAP and Websocket protocols.

D. Transformations

Suggested metamodels architecture and a set of software tools allows one to describe the transformation of these models and define information process of ABMS including the following steps.

Ontological modeling of the ABMS structure. At this stage, using the ontology editor, you need to define $M^{St\text{-}A}$ – the concepts describing the architecture of the agent, the environment, and other elements of the ABMS.

Ontological modeling of the ABMS behavior. At this stage, using the operations editor, it is necessary to describe the behavior of the ABSM elements (agent, environment) using the methods defined in $M^{Op\text{-}I}$.

$$M^{St\text{-}A} \times M^{Op\text{-}I} \rightarrow M^{Op\text{-}A} \quad (1)$$

Formation of the ABMS methodology description. At this stage, using the ABSM editor, necessary references between obtained elements of MOp-A and MSt-A are established (2). Thus, the description of the ABSM structure elements and its behavior is combined into a single M^{A} metamodel.

$$M^{Op\text{-}A} \times M^{St\text{-}A} \rightarrow M^{A} \quad (2)$$

Ontological modeling of the subject domain. At this stage, using the ontology editor, the concepts of the given subject domain, their properties and relations are described and hence MOnt-D model is defined.

Formation of the agent model structure for a given subject domain. At this stage, using the ABSM editor, a correspondence between the concepts of the subject domain and the concepts describing the ABSM structure is established. For example, "Specific product" – "Agent", "Pressure sensor" – "Sensor", "Increase pressure" – "Action". This can be represented as transformation (3).

$$M^{Ont\text{-}D} \times M^{St\text{-}A} \rightarrow M^{St\text{-}A\text{-}D} \quad (3)$$

Formation of the agent model behavior for a given subject domain. At this stage, using the knowledge base editor, knowledge bases describing the behavior of domain objects is designed (4) with elements of $M^{Ont\text{-}D}$ and $M^{St\text{-}A}$ as only input.

$$M^{KB} \times M^{Ont\text{-}D} \times M^{St\text{-}A} \rightarrow M^{KB\text{-}D} \quad (4)$$

Formation ABSM specification for a given subject domain. At this stage, using the ABSM editor, the MA metamodel is instantiated into specification driven by information by subject domain (5).

$$M^{A} \times M^{St\text{-}A\text{-}D} \times M^{KB\text{-}D} \rightarrow M^{A\text{-}D} \quad (5)$$

Setting initial conditions for ABSM. At this stage, ABSM specification is extended by the information about initial state of ABSM elements as well as some simulation parameters, so the following transformation is defined.

$$M^{A\text{-}D} \rightarrow M^{A\text{-}D\text{-}I} \quad (6)$$

Performing simulations. At this stage, with the help of the ABSM specification interpreter, ABSM simulation is carried out via interpretation (7) of $M^{A\text{-}D\text{-}I}$.

$$M^{A\text{-}D\text{-}I} \rightarrow cod^{I} \quad (7)$$

III. An Illustrative Example

A. Unique Mechanical Systems Description

As an illustrative example of applying the proposed MDD-oriented approach the development process of a unique mechanical system (UMS) agent-based simulation model is considered. The unique mechanical systems are understood to be systems manufactured in 1 ... 5 copies, operated in different conditions and implementing extreme technological operations as part of complex technical systems. The studies of such objects are conducted by with the authors' participation over the past 15 years. To date, methodology and related knowledge bases have been developed for solving various types of problems [11] and discrete event models [12] were created.

The UMSs are functioning under extreme conditions and can be characterized as dangers ones. So research and ensuring technological safety problems have to be stated and solved. In particular, the genesis, identification, and prediction of the UMS technical state problems are relevant allowing to form an effective system for managing the security of a UMS object. The genesis problem is solved by describing the dynamics of the technical state for the past period of functioning [11], which is the cause of the state under consideration, the identification problem is a description of the technical state at the current time, prediction problem is a description of the technical state dynamics at subsequent times, starting with the considered one.

The mentioned problems are poorly structured, information about the degradation processes regularities is heterogeneous and is presented both as formal models and as precedents and experts knowledge. Also, note that a common feature of the UMS studying is the lack of statistical information about the dynamics of changes in the technical state. So to provide a comprehensive understanding of UMS the application of special methods for processing qualitative information (in particular, artificial intelligence methods) is required.

The extensibility of information and algorithmic tools is an essential property due to the permanent appearance of new information obtained from the experiments and the analysis of the UMS operational data. In this regard, using agent-based modeling style does not strictly require to define in detail the behavior of the system as a whole (unlike system dynamics models or discrete-event models): model development is possible in the absence of in-depth knowledge of global dependencies, and refinements are usually made at the local level and do not require global change.

Following the proposed methodology [11], UMS can be considered as a hierarchical structure: "Detail (D) – Assembly Unit (AU) – UMS". Each element of the UMS model is considered on a set of causal information levels:

- **functional** (F) – evaluation of the object in terms of functioning: OK or not OK;
- **technical** (T)– description of parameters according to technical requirements;
- **physical** (Ph) – description of mechanical-physical-chemical parameters of degradation processes;
- **degradation processes** level (DP) – mechanism and kinetics of the processes.

In turn, each information level describes the dynamics of the UMS elements technical state and the system as a whole, which is reflected in the sequence of state classes:

- **O**riginal state (defect);
- **A**cceptable damage state (damage);
- **I**nvalid damage state (breakdown);
- **D**estructed state (failure).

For example, let us consider the basic algorithm for solving the identification problem of the UMS technical state (ITS). We represent the algorithm in the form of a generalized sequence of steps (8), where each step is the solution of the corresponding subtask:

$$ITS_{D} \rightarrow ITS_{AU} \rightarrow ITS_{UMS} \quad (8)$$

Each of these subtasks is determined by the selected classes of technical state dynamics ($k = 1..K$) for X element type and information levels ($m = 1..M$):

$$ITS_X = (ITS_{DP} \rightarrow ITS_{Ph} \rightarrow ITS_T \rightarrow ITS_F) \times$$

$$\times (ITS_O \rightarrow ITS_A \rightarrow ITS_I \rightarrow ITS_D):$$

$$(ITS_{11} \rightarrow \ldots \rightarrow ITS_{1M} \rightarrow ITS_{21} \rightarrow \ldots \rightarrow ITS_{KM}) \quad (9)$$

For each stage of identification process an appropriate knowledge base and a variety of computing modules have to be developed. Some examples can be found in [11, 12].

B. Agent-Based Modeling Process

The application of the proposed model-driven agent-based modeling approach for solving identification, and prediction of the UMS technical state problems can be considered as an iterative process of gradually refining the structure and behavior of ABSM elements with the ability to reuse the results of previous stages, as well as knowledge bases developed in other studies [11, 12]. Let us discuss this process on the example of designing the most important element of the ABMS – the agent.

Phase 1. As UMS has a hierarchical structure it is considered 3 types of appropriate agents: Detail (A^D) – Assembly Unit (A^{AU}) – UMS (A^{UMS}). Typically at the beginning of development process there is a not a lot of understanding about the specific structure of the agent behavior. Therefore, it is reasonable to reuse the common command cycle based on the standard architecture of a single-level reactive agent: sensing – communicating – decision-making – acting. According to the approach, the decision-making part is implemented by agent's knowledge base (KB^A) containing rules for describing the behavior of the agent and previous results can be utilized [11, 12] as the sequence of decision-making is determined by (8). The decision-making process is oriented vertically so at first all agents of A^D type is acting then A^{AU} and finally A^{UMS}.

Phase 2. Refining the decision-making process for each agent depending on its state during the identification problem by introducing new knowledge base related to appropriate classes: $KB^A = \{KB^O, KB^A, KB^I, KB^D\}$.

Phase 3. Adding problem-specific information about four levels of decision-making (9) for each type of agent via further specification of KB^A, for example, $KB^O=\{KB^{O\text{-}F}, KB^{O\text{-}T}, KB^{O\text{-}Ph}, KB^{O\text{-}DP}\}$

Phase 4. Based on obtained KBs and related command cycles it now becomes possible to extend identification problem up to prediction one. For this purpose a new KB^{deg} type describing the kinetics of the degradation process is proposed. KB^{deg} is responsible for changing A^D state regarding evolution in time. New A^D command cycle has KB^{deg} related reasoning executed before identification problem command cycle defined at 2nd and 3rd phases

So phases 2 and 3 are examples of decision-making knowledge bases detailing whereas phase 4 is an illustration of reusability principle of agent behaviors for solving different problems.

IV. CONCLUSIONS

The approach for agent-based simulation models development is presented in the paper. Application of MDD approach for describing development process allows suggesting the sets of metamodels, transformations and software tools that all together compose an essence of the approach.

One of the distinguished features of the proposed approach is the ability for explicit separation of the agent-based model specification into two parts: agent-oriented and domain-specific. This feature opens the opportunity to intensively reuse different methodologies aspects and apply them to particular subject domains.

As an illustrative example, the development process of the agent-based simulation model for solving the identification and prediction of the unique mechanical system technical state problems is considered. For instance, in the paper 4 phases of the agent's command cycle design process is presented. Approach-based consecutive refinement and the reuse of existing knowledge bases obtained in related studies facilitate ABSM development of complex systems.

The main challenge for future work is to utilize the proposed organization principles to create different types of supporting systems. The simplest case is the system for automated testing of hypothesizes about structure and behavior regarding the elements of the agent-based model. For example, for the subject domain concept chosen to be an agent different architectures and decision-making algorithms can be varied without changing domain-specific part of the agent-based model specification and after that compared by the human expert or some criterion according to the simulation results.

ACKNOWLEDGMENT

The reported study was partially supported by RFBR projects 18-07-01164, 18-08-00560.

REFERENCES

[1] Q. Tran, and G. Low, “MOBMAS: A methodology for ontology-based multi-agent systems development,” Information and Software Technology, vol. 50, pp. 697–722, 2008.

[2] G. Beydoun, G. Low, B. Henderson-Sellers, H. Mouratidis, J. G. Sanz, J. Pavon, and C. Gonzalez-Perez, “FAML: A generic metamodel for MAS development,” IEEE Transactions on Software Engineering, vol. 35(6), pp. 841–863, 2009.

[3] G. Kardas, “Model-driven development of multiagent systems: a survey and evaluation,” The Knowledge Engineering Review, vol. 28(4), pp. 479–503, 2013.

[4] F. Klügl and P. Davidsson, “AMASON: Abstract Meta-model for Agent-Based SimulatiON,” in LNCS, vol. 8076, M. Klusch, M. Thimm and M. Paprzycki, Eds. Heidelberg: Springer, 2013, pp. 101–114.

[5] M. Challengera, M. Mernikb, G. Kardasa, and T. Kosarb, “Declarative specifications for the development of multi-agent systems,” Computer Standards & Interfaces, vol. 43, pp. 91–115, 2016.

[6] A. Garro, and W. Russo, “EasyABMS: a domain-expert oriented methodology for agent-based modeling and simulation,” Simulation Modelling Practice and Theory, vol. 18(10), pp. 1453–1467, 2010.

[7] R. Siegfried, Modeling and Simulation of Complex Systems: A Framework for Efficient Agent-Based Modeling and Simulation, Berlin: Springer, 2014.

[8] N. Russell, A. H. M. Hofstede, D. Edmond, and W. M. P. Aalst, Workflow Data Patterns, Brisbane: Queensland University of Technology, 2004.

[9] J.L. Pereira and D. Silva, “Business Process Modeling Languages: A Comparative Framework” in New Advances in Information Systems and Technologies. Advances in Intelligent Systems and Computing,

vol. 444, A. Rocha, A. Correia, H. Adeli, L. Reis and M. M. Teixeira, Eds. Cham: Springer, 2016, pp. 619–628.

[10] O. Gutknecht and J. Ferber, "The madkit agent platform architecture," in LNCS, vol. 1887, T. Wagner and O. F. Rana, Eds. Heidelberg: Springer, 2000, pp. 48–55.

[11] A. F. Berman, O. A. Nikolaychuk, A. Y. Yurin, and A. I. Pavlov, "A methodology for the investigation of the reliability and safety of unique technical systems," Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, vol. 228, No. 1, pp. 29–38, 2014.

[12] O. A. Nikolaychuk, A. F. Berman, A. I. Pavlov, "Predicting the technical state of hazardous objects via simulation modeling," Journal of Machinery Manufacture and Reliability, vol. 46, No. 2, pp. 209–218, 2017.