The identification of unique mechanical systems state based on agent-based simulation modelling

O A Nikolaychuk¹, A I Pavlov¹ and A B Stolbov¹

¹Matrosov Institute for Systems Dynamics and Control Theory of Siberian Branch of the Russian Academy of Sciences, 134, Lermontov str., Irkutsk, 664033, Russia

E-mail: nikoly@icc.ru

Abstract. The report addresses the application of ADSkit approach and related software tool for creating agent-based models describing the technical state dynamics of unique mechanical systems (UMS)s. UMSs are manufactured in very few copies and furthermore the conditions of their exploitation can significantly diverse from each other. Also as a part of complex technical systems, UMS implements extreme technological operations. The step by step description of the ABSM development process for UMS is considered in the paper. In particular, the UMS ontology acts as a main source for model entities interpretation whereas their behavior mostly described in a rule-based or workflow manner. The testing simulations are carried out based on the ADSkit functionality related to the computational experiment management tool.

1. Introduction

The current state of information and communication technologies opens up new opportunities and creates niches for the active application of multi-agent approach: multi-agent systems (MAS)s and agent-based simulation models (ABSM)s. It is most effective to use MAS and ABSM for the analysis and development of systems with autonomous, proactive and heterogeneously distributed elements, for example, in such areas as industry, finance, service systems, logistics, et.al. In order to satisfy increasing requests for MAS from domain specialists, that lacks general programming or MAS development-related skills, modern MAS creation tools divide the application agent structure into at least two parts: invariant and domain. The first part includes functions that are common to all agents of the application system. This part is also known as "generic agent". The second part is a specialization of the "generic agent" depending on its subject area. Such approach helps potential users to lower their entry threshold as it hides MAS implementation routines and at the same time forms a framework that guides the MAS development process.

The development of the considered approach ideas was the purpose of the recent authors' works [1]. The special attention was devoted to the ABSM development process and further particularization of the agent structure. In the invariant part, the new two subparts are defined: one related directly to the ABSM methodology and another primarily regarding the functionality of ABSM software implementation tools. It is also assumed that the invariant part has no bindings to a specific software tool and provides a sufficient level of generality to be able to reuse the methods of various existing third-party libraries and platforms for creating ABSM and as a consequence to combine them. The proposed approach and homonymous software package are called the ADSkit (Agent Development Support kit). Thus, at the abstract level the new extended structure of the ABSM element defined by
ADSkit approach will consist of the following parts: 1) invariant; 2) methodological; 3) domain; 4) configuration (software configuration for the ABSM implementation).

The ADSkit methodology is based on a model-driven approach [2, 3] and for this reason introduces into development process a four-level set of models with information about the structural and behavioral characteristics of ABSM elements. To represent and process this information an ontological (conceptual models), rules (IF-THEN statements) and workflow (visual programming paradigm) methods are utilized. Two main top-level methods currently have been developing: the method for describing the ABSM design methodology and the method for applied ABSM creation based on the first one. Metamodels architecture, a set of software tools for ABSM development and relations between these two main ADSkit artifacts are proposed in a previous article [1].

At the abstract level, the scheme of information process for ABMS development defined by MDD transformations of ADSkit models and metamodels is also considered in [1]. The main focus of the current paper is the application of the ADSkit approach to the analysis of a unique mechanical systems (UMS)s, namely the technical state identification problem of UMSs. The unique mechanical systems are the systems manufactured in 1 ... 5 copies, operated in different conditions and implementing extreme technological operations as part of complex technical systems. The UMSs are functioning under extreme conditions and can be characterized as dangerous ones. The original research methods for UMSs analysis in the context of knowledge-based and simulation (discrete event models) systems have been developing for a decade [4].

2. Materials and methods

ABSM Development Process

2.1 A Method For Describing The Design Methodology

From the viewpoint of systems analyst (a specialist in agent-based simulation), the process of describing the ABSM design methodology can be defined by the following sequence:

Stage 1. Creating $M_{St-A}$ – an ABSM structure ontological model. The metamodel for $M_{St-A}$ is well-known $M_{Ont}=\langle$Concept, Attribute, Relation, Instance$\rangle$. At this stage, it is necessary to describe a set of interrelated concepts that will be used as elements (building blocks) to represent the agent architecture, environment, and other ABSM entities in accordance with the chosen by the analyst ABSM design methodology. As an example of $M_{St-A}$ creation, let’s adapt a version of the GRAMS (General Reference Model for Agent-based Modeling and Simulation) model [5]. For brevity, the five of its basic concepts are considered:

$M_{St-A}=\langle$Model, Environment, Object, Event, Agent$\rangle$.

The listed components are concepts in terms of $M_{Ont}$: they can have descendants and are also connected via attributes and relationships to other $M_{St-A}$ concepts:

$\langle$Environment$\rangle=\langle$Concept$\rangle$,

\{Property of environment\},

\{Location\},

\{Update functions of the environment\},

where $\langle$Concept$\rangle \in M_{Ont}$ – the basic concept created by the analyst to identify the environment,

$\langle$Update functions of the environment$\rangle$ contains the description of functions for updating environment properties and locations. $\langle$Object$\rangle$ is set in terms of the concept element of $M_{Ont}$ and will be refined in subsequent stages. $\langle$Event$\rangle$ – with the help of this concept (or its descendants), one can describe instantaneous phenomena that occur at a certain time $t \in T$ and can change the state of the model. Events can be endogenous (occur within the agent or environment and do not affect other ABSM elements) or exogenous.

$\langle$Agent$\rangle=\langle$Concept$\rangle$,\{Sensor\},

\{Effector\},\{Reasoning tool\},

where $\langle$Concept$\rangle \in M_{Ont}$ – the basic concept created by the analyst to identify the agent and to define its properties; $\langle$Sensor$\rangle$ – with the help of this concept (or its descendants), one can describe the
interface through which the agent receives information from the environment and if stated in the methodology, from other agents. <Effector> – through this concept, it is possible to define an interface that allows the agent to interact with the environment and agents and actively achieve their goals. <Reasoning tool> is an internal agent component that provides data processing received by sensors; decision-making and effector control. There are several ways to classify the agent architecture: reactive, deliberative and reflective; single-level, horizontal multi-level, vertical multi-level. Depending on the specific type of agent architecture, a set of the reasoning tools may include, for example, knowledge base, workflow, planner, etc.

Stage 2. ABMS behavior ontological modeling. The M^{Op-A} model is necessary to describe the behavior of active elements with the ADSkit functional components [1] as building blocks: Data Controller, Communication Engine, Inference Engine, Simulation Engine, Imperative Engine. Each method of the functional component is a basic operation Op^{B}, which can be combined into a workflow [6] using sequential process with conditional and cyclic statements to implement a more complex behavior – a composite operation Op^{C}. The operation parameters can be the elements of M^{Out}, and M^{St-A} as a special case. More information about the model M^{Op}=<Op^{B}, Op^{C}> and the features of its software implementation are discussed in [1].

Let’s look at the features of this stage using as an example the Inference Engine component, containing the following methods: LoadInitialState – setting the initial state of working memory and ExecuteReasoning – executing inference process. The first method takes a list of elements in the M^{Out} format as input parameters. The second method returns the final state of the working memory in the M^{Out} format also, and at the input accepts the knowledge base described in terms of the M^{KB} metamodel. Note, that the specific content of the knowledge base is determined at the next stage "Domain-specific behavior for the agent model", whereas at the current stage the developer sets only its ID (for more information, see the example below).

At this stage the proposed GRAMS-based M^{St-A} model it can be clarified in the following way: <Update functions of the environment> and <Reasoning tool> structures will be described and implemented as a set of composite operations Op^{C}. The systems analyst may use the principles of structural design and create an excessive set of interrelated "behaviors". The selection of specific behaviors from this set is carried out at the next stages.

Stage 3. ABMS methodology specifying. At this stage, descriptions of M^{St-A} elements and related behaviors (either in the form of M^{Op-A} or M^{KB}) are coupled into a single metamodel – M^{A} with the following structure of:

\[
\text{<Name, el^{St-A}, Behaviour>,}
\]

where <Name> – identifier for the "ABSM element – behavior" pair. For example, different types of sensors have different behaviors, but they can have the same name for ease of addressing them (see the example below). Thus, the M^{A} metamodel is a formalized description of the design methodology and will later be explicitly used in the development of applied ABSM. As an illustrative example of the element behavior from M^{A} let’s consider the pseudocode of the agent life cycle.

DataController → RetrieveProperty(Agent, "Sensor", Sensors);
FOREACH (Sensors as Sensor) DO
ImperativeEngine → Execute(Sensor, "Process And Get Data", DataOfAllSensors);
DataController → RetrieveBehaviour(Agent, "Analyze Inputs", KnowledgeBase1);
InferenceEngine → LoadInitialState(Agent, DataOfAllSensors);
InferenceEngine → ExecuteReasoning(KnowledgeBase1, ResOfReasoning1);
DataController → RetrieveBeahviour(Agent, "Make Effectors", KnowledgeBase2);
InferenceEngine → LoadInitialState(ResOfReasoning1);
InferenceEngine → ExecuteReasoning(KnowledgeBase2, EffectorsList);
DataController → RetrieveProperty(ABM, "Environment", Environment);
DataController → RetrieveBehaviour(Environment, "Constraints For Effectors", EnvKnowledgeBase);}
DataController → LoadInitialStat(EffectorsList);
DataController → ExecuteReasoning(EnvKnowledgeBase, CheckedEffectorsToExecute);
SimulationEngine → Execute(ABM, "Execute action", Agent, CheckedEffectorsToExecute);

Here, input parameters of the functional components methods are shown in italics, and output parameters are shown in bold. To simplify development the DataController component provides the following auxiliary methods: RetrieveProperty – returns the property value of a concept or instance by its name, RetrieveBehaviour – returns the behavior of M^4 by its name for a concept from M^{St,A} or its corresponding instance. Agent, Sensor, ABM, Environment are instances from M^{St,A} related to specific ABSM elements at the model execution stage (runtime phase). The final state of the working memory for the reasoning component in the form of M^{Out} is returned via the following variables ResOfReasoning1, EffectorsList, CheckedEffectorsToExecute.

The considered algorithm means the following. At the beginning, a composite operation with the equal name "Process And Get Data" is performed for each sensor. The operation returns information DataOfAllSensors in terms of M^{Out}. For example, for a sensor type that reads messages that came directly to an agent, the function of Simulation Engine component can be utilized: RetrieveNotifications(Agent, Notifications). Then, using the appropriate knowledge bases (KnowledgeBase1, KnowledgeBase2, EnvKnowledgeBase), the following actions are performed: sensors are analyzed sequentially, a set of possible agent reactions is formed; suitable effectors are selected; effectors are analyzed by the environment; allowed effectors are activated.

2.2 An ABSM Design Method

The ABSM design method provides a step-by-step transformation of the original conceptual model of the domain in the ontological form into the ABSM specification in accordance with the methodology description (in the M^1 form) created during the first three stages. Let's look at the development process defined by the ABSM design method from the viewpoint of a non-programming user on the example.

The research object is a unique mechanical system "tubular reactor" which is a component of "low-density polyethylene production". The purpose of this UMS is to provide heat transfer in conditions of high pressure of the working environment of the pipe space and the contacting medium in the inter-tube space (in the jacket). UMS consists of two types of assembly units – "bent pipe in a jacket" and "straight pipe in a jacket". In turn, each of the assembly units accordingly consists of the parts – "heat exchange bent pipe" or "heat exchange straight pipe". The operating conditions of the UMS are determined by the presence of media and their characteristics: a heat exchange medium circulates in the inter-tube space and an explosion and fire hazardous gas moves in a thick-walled pipe. The heat exchange medium is mechanically and chemically purified water that circulates at a pressure of 3…4 MPa and a temperature of 290…300ºС. Explosion and fire hazardous gas flows at a pressure of up to 250 MPa and a temperature of 260…300ºС. The result of the UMS operation is a product of gas polymerization – polyethylene. The described process parameters and processed substances are characterized as dangerous and allow us to classify the considered UMS as dangerous. To develop the "tubular reactor" ABSM, the domain expert must complete the following sequence of stages, continuing the ABSM developing process:

Stage 4. Subject domain ontological modeling: creating M^{Out-D} model in the form of M^{Out}. Here UMS; assembly unit; part; technical characteristic (material, medium, stresses), requirements, conditions; levels of states (functional, technical, physical, degradation processes), classes of state on levels (perfect state, up state, down state, hazardous state) are examples of main concepts for M^{Out-D}.

Stage 5. The selection of ABSM methodology described in the form of a metamodel M^A.

Stage 6. Domain-specific model structure for the agent model. At this stage, the descriptions of the structure elements – M^{St,A} and domain ontology M^{Out-D} are coupled into a single M^{St,A-D} model with the following structure:

<el^D, el^A>, el^A ∈ {Concept|Instance} ⊂ M^{St,A}, el^D ∈ {Concept} ⊂ M^{Out-D},

where el^* – the model element, * – the model type.
In a fragment of correspondence between $M^{St-A}$ и $M^{Ont-D}$ elements (Fig. 1) rectangles with a bold border represent concepts of the domain, while others represent concepts related to ABSM methodology. The arrows with the dotted line show the relationship between $M^{St-A}$ and $M^{Ont-D}$, and with the normal line inside $M^{St-A}$ (the black arrow head means inheritance).

![Figure 1. Relationships between ontological model of the UMS and the structural elements of the agent-based methodological model.](image)

Stage 7. Domain-specific behavior for the agent model. At this stage, the $M^{KB-D}$ knowledge base is developed in terms of the $M^{KB}$ metamodel for the considered subject domain, including rules and fact templates that are created based on concepts, attributes, and relationships from the $M^{Ont-D}$. The $M^{KB-D}$ model can be developed from scratch or based on an existing knowledge base created without taking into account agent model specifics. In the latter case, $M^{KB-D}$ is supplemented or refined by rules using elements from $M^{St-A}$ and $M^{Ont-D}$ regarding their connection via $M^{St-A-D}$. For example, a knowledge base describing the mechanisms and kinetics of the processes is utilized to determine the type of degradation process that occurs on the UMS. At this stage, a set of external computing modules is also formed. These modules also can be used for implementing domain behavior. For example, calculating the parameters of a crack (length, depth, etc.) formed as a result of a degradation process.

Stage 8. Here the agent model specification $M^{A-D}$ for a given subject domain is created by combining all the models generated in the previous stages.

Stage 9. ABSM initial conditions. At this stage, by creating instances of concepts from $M^{A-D}$, the initial conditions are set, the parameters of the computational experiment are configured, including the setups of callings to the computational modules.

The model time interval [0, 26,000 hours] and the step – 10 hours is defined for the considered UMS. After starting the modeling process, agents of the "Part", "Assembly unit" and "Mechanical system" types consistently identify their technical condition, receiving information about the identification results from dependent agents, where dependencies are defined according to the structure of the UMS. Identification of the technical state is based on a given model of agent behavior, implemented as a result of the interaction of software components of knowledge processing and
computational models. The result of modeling is a state tree, where each vertex of the tree is described by a set of parameters and their values at each model time.

3. Results

Computational Experiment Management Tool

Implementation of a simulation experiment based on the ABSM specification generated during the stages 1 – 9 is performed using a special ADSkit means – computational experiment management tool (CEMT). With the help of CEMT end-users can create ABSM initial conditions and save simulation results, as well as directly control the simulation process. Currently the following assumptions are used for forming the specification of a computational experiment:

- to simplify the implementation stage all external modules registered for usage in ADSkit are virtually Java subprograms. In the future, it is planned to provide interaction with external modules based on the SOAP and Websocket protocols.
- since the concept of a conceptual model can be associated with many knowledge bases and other components of a knowledge-based system a value constraint is added: a single domain element can only be associated with one element from the methodology. Otherwise, additional information models have to be created to resolve possible conflicts.
- knowledge bases that describe the subject behavior of the ABSM elements do not use information related to the specifics of agent modeling, i.e. the relationships between the agent and domain ABSM parts are explicitly and completely contained in the system of models that define the ABSM specification.

Taking into account the described limitations, CEMT provides a web interface that allows end-users to initialize the model (the number of agents, initial parameters of agents, etc.) and control the modeling process (start/stop the experiment, change the parameters of the model, agents, etc.). The specification interpreter provides direct execution of the ABSM. In this case, Madkit [7] is utilized as the modeling engine, and Drools [8] – is used for logical output.

4. Conclusion

The original approach and related software (ADSkit) designed by the authors is utilized for developing agent-based simulation models of unique mechanical systems (UMS)s. As an illustrative example “tubular reactor” UMS is considered. The ADSkit approach takes into account features that are specific as to the UMS itself, as for weakly structured tasks in general. This functionality is achieved in ADSkit by the combined use of approaches from different problem areas: multi-agent systems, simulation, ontologies, rule-based systems and workflow systems.

5. Acknowledgments

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

References

[1] Nikolaychuk O A, Pavlov A I and Stolbov A B 2019 Advances in Intelligent Systems Research 166 201–206
[2] Kardas G 2013 The Knowledge Engineering Review. 28(4) 479–503
[3] Garro A and Russo W 2010 Simulation Modelling Practice and Theory 18(10) 1453–1467
[4] Berman A F, Nikolaychuk O A, Yurin A Y and Pavlov A I 2014 Part O: Journal of Risk and Reliability 228 (1) pp. 29–38
[5] Siegfried R 2014 Modeling and Simulation of Complex Systems: A Framework for Efficient Agent-Based Modeling and Simulation (Springer Vieweg)
[6] Russell N, Hofstede A H M, Edmond D and Aalst W M P 2005 LNCS 3716 353–368
[7] Gutknecht O and Ferber J 2000 LNCS 1887 48–55
[8] Salatino M, Aliverti E and De Maio M N 2016 Mastering JBoss Drools 6 for developers (Birmingham: Packt Publ)