International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 24 (2015) pp 45292-45296

About The Development Of New Tools System-Object Simulation Process And Traffic

Sergey I. Matorin, Alexander G. Zhikharev, Karina V. Korchagina, Tatiana V. Saitseva
Belgorod State University
Russia, 308015, Belgorod, Pobedy St., 85

Abstract- This paper deals with a new method of simulation modeling approach using "Node-Function lens" system-object representation method of knowledge, mathematical objects and the theory of language to describe functional units UFO script. Test examples of simulation process and traffic flow performed by a developed a new version of software tools system-object modeling UFO-toolkit.
Keywords: system-object modeling technology "Node-Function lens", simulation, formal presentation of the organizational system, system approach, simulation operation of the process.

Introduction

Since long accomplishments of simulation modelling have resulted in recognition thereof as a promising research area and wide use of its tools for solution of various practical tasks. However, the issues that are natural to any evolving area of science and engineering stimulate continuation of research and development works including the simulation modeling area.

As of today there is no established definition of the term ‘simulation modeling’. This is why for specification of the results proposed in this paper it is needed to clarify the author’s understanding of the essence of this research-practical trend. In general, it corresponds to definitions specified in the interview of the President of the National society for simulation modeling, corresponding member of the RAS, director of the SPIIRAS R.M. Yusupov [1]. At that the most adequate to the trend of the authors’ activity is understanding of simulation modeling that was mentioned in this interview and in the relevant Wikipedia-item as well (*https://ru.wikipedia.org/wiki/Simulation modeling*) as the ‘research method in which the system under investigation is substituted through a model that adequately describes the real system, with the use of which experiments are performed in order to get information about this system’. Relying on this definition demonstrates that research and development of authors seem to lie within the discrete-event-driven approach to simulation modeling.

Procedure

The necessity of substituting the system under investigation through a model adequately describing this system, directly makes to use the system approach for design of such a model. For system modeling the authors use the system-object orientation ‘Node-Function-Object’ [2-4] (**UFO-approach**: *http://ru.wikipedia.org/wiki/Node-Function-Object*). This approach allows describing any system as a ‘Node-Function-Object’ element (**UFO-element**) [4] comprehensively and at the same time – from three perspectives:

– as a *structural element* of the super-system in the form of intersection of links with other systems — **node**;

- as a *dynamic element* fulfilling a specific function in terms of maintaining the super-system by means of balancing this node — **function**;
- as a *substantial element* implementing this function in the form of some tangible formation featuring design, operating and other characteristics -**object**.

UFO-elements compiled in different configurations make the element interaction charts that allow visualizing functionality of elements of a higher-level system. Thus, the system being designed is represented in the form of hierarchy of UFO-elements. Such representation allows taking into account different aspects of consideration of this system (structural, functional, substantial) within a single system-object model – **UFO-model**.

Main part

For the purposes of automation of the UFO-approach use the CASE-tool **UFO-toolkit** has been designed and implemented (certificate of the Russian Agency for Patents and Trademarks №2006612046, *http://www.ufo-toolkit.ru/*). The hierarchy of UFO-elements and their configurations supported by the UFO-toolkit is based on classification of links (flows) the intersections of which form nodes. Modeling of any system starts with specialization of the basic link classification according to the specific object domain. The abstract class "Link (**L**)" in the basic link classification is divided into subclasses 'Material link (**M**)' and 'Data link (**I**)'; the material link class is divided into sub-classes 'Real connection (**S**)' and 'Power connection (**E**)", the data connection class – into the sub-classes 'Data link (**D**)' and 'Control link (**C**)'.

On the basis of the UFO-approach the authors have designed the formalized 'System-Object Knowledge Representation Method' (SOKRM) as a tool for design of the universal models of knowledge of systems of random nature. The essence of this method is described in the work [2]. The formal technique of SOKRM allows describing UFO-elements and links/flow in terms of *the object calculus according to Abadi-Cardelli* [5, 6] as well as λ-calculus [7-9].

At that an UFO-element is represented in the form of a specialized abstract object (**узлового объекта**) of this calculation:

$$G = [l?_i = a?_i,\ l!_j = a!_j;\ l_n = F(l?_i)l!_j;\ l_m = b_m],$$ where:

- G – node object;
- $l?_i$ – node object field (may represent a set or domain) containing values of the input flow objects $a?_i$ and, accordingly, features the same data type;
- $l!_j$ - node object field (may represent a set or domain) containing the value of the output flow objects $a!_j$ and features the same data type;
- l_n – node object method (may represent a set or domain) transforming the input flow objects into the output ones;
- l_m - node object method (may represent a set or domain) containing the main features of this object (b_m).

Thus, the node of an UFO-object is described by the sets of input ($a?_i$) and output ($a!_j$) flows of the G object, the function of an UFO-element – by the ($F(l?_i)l!_j$) method of the G object, and the object of an UFO-element – as the set of constants (b_m) characterizing the G object.

Link/flow is also represented as a specialized abstract object (**flow object**) [10] of the above-mentioned calculation:

$$a_i = [l_j = b_j],$$ where:

- a_i – flow object;
- $l_j = b_j$ – flow object fields with some values b_j.

Since computation in the object calculus represents the sequence of calls and overriding methods for which the reduction rules have been assigned, in the SOKRM formal transformations are performed according to the **rule of calling the node object method** of the following kind: $G.l_n\ l!_j\{l?_i \mapsto G\}$.

In terms of the UFO-approach the simulation model of a system, first of all, represents a configuration of the UFO-elements (UFO-model) that adequately describes this system'. Mathematically, in terms of the SOKRM formalism a simulation UFO-model represents a combination of node objects and flow objects interaction of which is determined by the above-

mentioned rule of method calling:

$$a_i = [l_m = b_m]: a_i = a?_i = l?_i \rightarrow G_k.l_n \rightarrow l!_j\{l?_i| \rightarrow G_k\} |\rightarrow$$
$$a_{i+1} = [l_{m+1} = b_{m+1}]: a_{i+1} =$$
$$a?_{i+1} = l?_{i+1} |\rightarrow G_{k+1}.l_{n+1} \rightarrow l!_{j+1}\{l?_{i+1}| \rightarrow G_{k+1}\} \rightarrow a_{i+2} =$$
$$[l_{m+2} = b_{m+2}]: a_{i+2} =$$
$$a?_{i+2} = l?_{i+2} |\rightarrow G_{k+2}.l_{n+2} \rightarrow l!_{j+2}\{l?_{i+2} |\rightarrow G_{k+2}\} \rightarrow a_{i+3} =$$
$$[l_{m+3} = b_{m+3}]: \ldots$$

Thus, functioning of the simulation model is determined by node object methods describing how the input flow objects are transformed into the output flow objects.

The authors are currently developing the new version of the software tools for the system-object modeling UFO-toolkit in which the option of describing the node object methods using the language of description of the functional nodes **UFO-script** that is syntactically similar to the Pascal programming language is implemented.

Let's consider peculiarities of the language proposed in details.

The following expression is used for declaration of variables used within description of functioning of a particular process:

var <variable name 1>,<variable name 2>,<…>: data type.

In the example provided the variables names follow the key word '*var'*, then the comma separated names of variables are specified, after the colon the type of data stored in variables is specified. AN UFO-script allows handling the following data types: *integer* – integer data type; *real* – represents fractional numbers; *string* – string data type; *Boolean* – logical data type.

Each operator UFO-script ends with the symbol ';'. The software code fragments are placed into program brackets as shown by the example below:

begin

…

end

For processing of the разветвляющихся алгоритмов исполнения функций УФО-элементов (node object methods) an UFO-script contains an *if* operator the syntax of which is shown below:

if (condition)

then

begin

// set of operators № 1

end;

else

begin

// set of operators № 1

end;

The *if* operator of the UFO-script language acts like in the other programming languages: if a condition takes a true value – the program fragment № 1 is executed, otherwise, if a condition takes a false value the program fragment № 2 is executed.

For execution of cyclic algorithms the UFO-script language contains two kinds of the loop organization. The first one – the counting loop the syntax of which is shown below:

for i:=<initial value of the counter> *to* <final value of the counter> *do*

begin

<*operator* 1>;

<*operator* 2>;

…

<*operator* n>;

end;

Except for the above-specified example, there is also an option to operate the cycle with the condition:

while <cycle execution condition > *do*

begin

<*operator* 1>;

<*operator* 2>;

…

<*operator* n>;

end;

The above-mentioned cycle will be executed as long as the cycle execution condition is true, this is why the exit condition shall be provided in the cycle body.

Except for the standard structures, an UFO-script features a number of pre-defined procedures and

functions meant for simulation of operation of functional nodes. The pre-defined functions have been designed for obtaining the values of input flows of the node to which a function described by means of an UFO-script refers:

getlinki (‘name of the input flow object’) – input flow of integer type;

getlinkr (‘name of the input flow object) – input flow of real type;

getlinkb (‘name of the input flow object) – input flow of Boolean type;

getlinks (‘name of the input flow object’) – input flow of string type.

Upon execution of the necessary actions the values of the output flow objects shall be set for which the following procedures have been designed:

setlinki (‘name of the output flow object’, value) – output flow of integer type;

setlinkr (‘name of the output flow object’, value) – output flow of real type;

setlinkb (‘name of the output flow object’, value) – output flow of Boolean type;

setlinks (‘name of the output flow object’, value) – output flow of string type.

Conclusions

The specified structures of the language describing functional nodes allow simulating functioning of the process to the fullest extent.

In order to use the created system simulation model the initial model parameters shall be set. These parameters are determined by the input flow objects of the context system model. After initial parameters have been set the model is started up for execution. Upon launching of simulation the designed model is translated into the program that is executed according to the following schedule: the function of the node the flow objects of which are initialized by specific values of their parameters is executed.

For simulation of the process functioning in the UFO-toolkit there is a separate module that enables starting the model, setting the model parameters, stopping the model operation, visualizing the results of simulation modeling.

Summary

The demonstrated options of the software tool UFO-toolkit speak of versatility of the proposed simulation modeling method that is achieved due to description of functional nodes with the use of the UFO-script. Surely, one may speak here of excessive efforts by script writing, however, these efforts ensure flexibility of such approach and possibility to simulate processes of various nature.

References

1. Russian National Society for Simulation Modeling – начало пути. Interview of R.M. Yusupov, corresponding member of the RAS, director of the SPIIRAS \\ CAD/CAM/CAE Observer. – 2012. - №2 (70).- PP. 10-18

2. S.I. Matorin, A.G. Zhikharev. Method of formalization of organizational knowledge // Artificial intelligence and decision-making. – 2011. – № 2 – PP. 12-18.

3. A.G. Zhikharev, S.I. Matorin, E.M. Mamatov, N.N. Smorodina. About the system-object method of representation of organizational knowledge // Scientific journal of the Belgorod State University. Series History. Political science. Economics. Informatics. – 2013. – № 8 (151). - Issue 26/1.

4. A.G. Zhikharev. About the new technology of knowledge representation for the decision support systems // Scientific journal of the Belgorod State University. Series History. Political science. Economics. Informatics. Информатика. – 2011. – № 19 (114). - Issue 20/1. – PP. 151-156.

5. Abadi, M., and Cardelli, L., 1996, “A Theory of Objects”, Springer-Verlag.

6. Hindley, J. R., 1997 “Basic Simple Type Theory”, Cambridge Tracts in Theoretical Computer Science Cambridge University Press, Cambridge, Volume 42.

7. Hindley, J. R., and Jonathan P. S., 1986, "Introduction to Combinators and λ-Calculus", London Mathematical Society Student Texts. Cambridge University Press, Volume 1.

8. Huet, G., 1975 "A unification algorithm for typed λ –calculus". Theoretical Computer Science, 1, PP. 27–57

9. Luo, Z., 1994 "Computation and Reasoning: A Type Theory for Computer Science". Number 11 in International Series of Monographs on Computer Science. Oxford University Press.

10. S.I. Matorin, A.G. Zhikharev. About the new formalized method of representation of organizational knowledge by means of computing tools // Series The issues of radio electronics. – 2011. – Issue 1. – PP. 120-131.