Complex Systems Structural Theory as the Foundation of the End-to-End Technology for the Software Development

Yury Brodsky[a]
Federal Research Centre "Computer Science and Control" of the Russian Academy of Sciences,
44/2, Vavilova str. 119333, Moscow, Russia
yury_brodsky@mail.ru

Keywords: Complex Systems, Species of Structure, Model Synthesis, Model-Oriented Programming, Programming Paradigm.

Abstract: We propose a formal definition of the complex system computer model as N. Bourbaki's construction – the "model" species of structure. The mathematical objects set defined by the "model" species of structure has the following two properties: a complex created by combining mathematical objects of the "model" species of structure according to the certain rules is itself a mathematical object of the same "model" species of structure. The organization process of computation is same for all mathematical objects of the "model" species of structure and therefore can be implemented by a single universal program. These two properties allow us to build a new technology for the specification, system design and software developing of the complex systems models – Model Synthesis and Model-Oriented Programming. The programming paradigm proposed is a complete implementation of CAD methods in the software designing domain.

1 INTRODUCTION

Several approaches to the object programming there were in the history of computer science, with the behavior of objects. Usually, the researchers engaged in the problems of AI originated these approaches (Hewitt, 1977), agent-oriented programming (Shoham, 1993), and ABMS (agent-based modelling and simulation) systems.

The author agrees with the central message of these approaches, but the details of these descriptions of the behavior in (Hewitt, 1977; Shoham, 1993) are too anthropomorphic and related to the AI specific field. The agents here are not universal enough – their behavior is described in human like terms such as "choices," "commitments," "beliefs".

The approaches listed above raise the question a – what should be the modeling agent and its behavior, for being sufficient for any complex system modeling, and together the most closest to the necessary for the modeling conditions?

MISS, multilingual integrated simulation system, elaborated in the Computer Centre of the USSR Sciences Academy (Brodsky, 1991) was the full implementation of the concept proposed. The standard unit of the programming here was a model with completed behavior, besides the functions and characteristics, – the ability to respond by standard way on the standard requests. This system did not have enough theoretical grounding then, in the 90th. For example, some calculations might be parallel, but it was unclear whether this parallelism leads to conflicts, or not.

This paper will propose a minimalistic construction of a universal agent for systems with "reactive" behavior – predefined standard answers to standard internal and external environment queries. Such an agent can be considered a functional analog of the computer's operating system.

The tools for building multi-agent systems based on the proposed agents were first implemented in the late 80th (Brodsky, 1991). A mathematical description of the universal agent structure and the justification of its necessity and sufficiency for the multi-agent systems with the reactive behavior realization were given later, in (Brodsky, 2014), (Brodsky, 2015).

[a] https://orcid.org/0000-0002-0565-4957

2 STRUCTURES

Structuralism bases on revealing relations on base sets (the structure), which preserves during some transformations. So, the attention of the researcher moves from the base sets features to the system properties and relations among the base sets.

The characteristics of the phenomenon attribute under the study change over the time. Still, there is something permanent in these changes – the way of their mutual relations, which is the object of detection and study by the mathematical methods. The modeling bases on an analogy which specifies a mapping of the modeling object attributes into various characteristics of the model.

The developing of a model is revealing and assigning the relations among its base sets in a way reflecting the law of the relations of the modeling object attributes.

An example of the universal description of mathematical objects structures is the concept of N. Bourbaki's species of structure (Bourbaki, 2004).

Since the differential and integral calculus origination, differential equations are the main structures used to describe and model complex systems.

3 DIFFERENTIAL EQUATIONS

Currently, differential equations are the primary tool for constructing mathematical models. The advantages of this tool are apparent: first, if the system has a time derivative, we immediately get its dynamics and prediction; secondly, we can use to investigate the model all the means associated with the concept of the well-posed problem, and this fact is quite important. Indeed, if there is no solution or uniqueness, or continuous dependence upon the initial conditions, or stability in any sense, how can we trust in such a model!?

The advantages of the differential equations as a modeling tool are their drawbacks simultaneously. The main one is that those differential equations are not suitable for the behavior describing. If we have a well-posed problem - our trajectory is unique, continuously depends upon the initial conditions, and is stable - and this leaves to it no behavioral alternatives.

The attempts to overcome this disadvantage are known. For example, Georgy Vasilievich Korenev (1902-1980), a remarkable professor of the MIPT, who can be called Tsiolkovsky of Robotics, proposed introducing special target functions into the right-hand sides of differential equations to describe purposeful actions.

Something similar positional differential games offer. However, positional differential games are very complex mathematical objects, and it is possible to analyze thoroughly only the most uncomplicated cases. Even in these cases, the game's equilibrium may be quite different from the equilibrium states of the system of its differential constraints with the controls fixed.

The practice constantly collides us with systems where hundreds and thousands of agents act purposefully.

4 MODELLING OF THE BEHAVIOUR

With the behavior of complex systems and their components, the author faced modeling complex organizational and technical systems (Brodsky, 1991). Such systems are not like Kantian "thing-in-itself," which needs studying with the help of a model. People created them or are going to create, so we need not invent the behavior of their models – all is determined either by design, project documents, or intelligence data – and for the modeler, it remains only to reproduce it. The complexity of the problem is that usually, it is necessary to synthesize the behavior of the entire complex system as a whole, with all its systemic effects. At the same time, we know only individual agents' behavior and the existing relations between them.

In work (Brodsky, 2014), we found out the class of systems for which building a model is possible – a specific analog of the existence of the solution theorems for the differential equations. A type of model closed at each time point, with piecewise continuous and left continuous trajectories. Continuity from the left is required to avoid the situations like von Neumann's fly (MacRae, 1999), or Thomson's lamp (Thomson,1954), where some system states are not the complete results of their pre-history. The model closeness at a time point means the ability to determine its internal characteristics by the values of its internal and external variables at this point:

If there is a discontinuity of the trajectory, and calculate the gap if it is.

Then determine a positive forecast interval.

Finally, to calculate the continuous evolution of the model's characteristics on this interval.

As for the remaining components of the traditional well-posed problem: uniqueness, continuous dependence on initial conditions, and stability – it is proposed to clarify them experimentally, with the help of simulation on the model.

Further, in work (Brodsky, 2015), the end-to-end technology was offered for the complex systems models specification, design, and software implementation. The basic concept of this technology is a "model" – a representative of the Bourbakian species of structure "model". In addition to the characteristics, the model's base sets include methods that calculate gaps, the continuous trajectory evolution, and the events associated with the completion of the forecast intervals.

The universal program of execution of the models provides dynamics of these structures. It focuses on parallel and distributed calculations and can organize the computation process for any model.

The model is functionally similar to the computer operating system: it can provide standard answers to the standard queries of the environment.

Combining several "models" into a complex does not lead us beyond the set of "models", therefore, this complex is also a "model". These facts make it possible to build very complex fractal constructions without fear of the computational complexity of such structures – the organization of calculations does not change. Moreover, the more complex the model is, the more computations it carries out in parallel.

Thus, we propose a new construction to describe a broad class of systems with the behavior, combining a static N. Bourbaki's structure with a computer program, focused on parallelism of calculations. Finally, let us note that any of differential equations system we can treat as a special case of such a construction. This fact preserves the succession with the previous modeling methods.

5 MODEL SYNTHESIS AND MODEL-ORIENTED PROGRAMMING

The theory of Model Synthesis and the Model-Oriented Programming paradigm as the way of the specification, design, and software implementation of large-scale simulation models, were developed in Dorodnitsyn Computing Centre of the USSR Sciences Academy, since the end of the 80th. The gist of this method was stated in works (Brodsky, 1991 - Brodsky, 2015), and the work (Brodsky, 2014) introduced these names. The concept of the "model" species of structure – the generic modeling agent – was the essence of that theory. This "model" is somewhat similar to the object of the Object Analysis, but in some way is opposite to it, this is reflected in the title of this section and the theory proposed. Still, besides characteristics and methods capable of doing something proper only if we call them, the "model" has an analog of the services of an operating system (Silberschatz, 2018), which always are working, on alert to answer to the usual requests.

5.1 A "Model" – the generic modelling agent

The main purpose of the Model Synthesis is a formalization of a complex system simulation methods, The "model" species of structure (Brodsky, 2015) in the sense of the Structure Theory by N. Bourbaki (Bourbaki, 2004), is defined, as a universal simulation agent. Here we try to give a descriptive and explanatory variant of its specification.

As the Object Analysis object, the "model" has its characteristics and methods. But the object has not its own behavior (though the concept authors sometimes say that it has) – the object is only a storage of different abilities – its methods. We may form the software system behavior calling them along with the skills of the other objects. The model differs from the object by its behavior. We cannot call methods of the model from the outside – all of them will be activated automatically according the model's program in a proper time, like the methods of computer operating system services. This time comes when a proper conjunction of the model's characteristics happens. As in a computer OS, the services work in parallel to meet standard requests, either in the model, processes run simultaneously. These processes consist of sequentially alternating actions of the model, programmed as the mathematical functions of its characteristics. The events are mathematical functions that calculate if the proper combinations of the model characteristics occur, and in the case of the event occurrence set the switching of the process elementary actions.

Let us classify the elements – elementary actions of the processes, in comparison with the step of modeling:

1. Fast or instant elements. They happen immediately concerning the modeling step, – the way to calculate gaps in the model's trajectory or discrete characteristics.

2. Slow or distributed elements. They take some positive model time, that is comparable with the modeling step selected. Any suitable interval of the model time produces correspondent result for these elements. The slow elements – are the means of the model's continuous characteristics calculating.

The events – are the entities that we cannot miss during our modeling. They control the elements switching in the process and the synchronization points where the stage of the model requires the corresponding response.

Now let us give the rules of the model's behavior typical to all the models, – the rules of the simulation calculations (Brodsky, 2015).

We choose a modeling time step Δt. If a step is the very first one, we know the internal variables start values, and the processes start elements. On the further steps, we have processes' current elements and the internal variables at the step of simulation beginning. All the external variables we supposed to be able to observe at any moment.

Then at the beginning of the step

1. The model's behavior program (switching rules) determines the correspondence between the events and the current elements of the processes. So, we compute all the events correspondent to the processes' current elements. All events may be calculated parallelly, but to proceed the calculations further, we are to wait the completion of the last of them. If there are occurred events, we look for changes to the fast elements. If such switches take place – we run the fast elements. They can be also parallelly calculated, waiting the completion of all, to proceed further. Then let us go to the item 1. If no switchovers to the fast elements – we change to the slow ones and move to the beginning of our algorithm (item 1).
2. If there are no occurred events, we choose the minimal $\Delta\tau$ from their predictions.
3. If our modeling step Δt is lesser or equal then this minimal prediction time $\Delta t \le \Delta\tau$, – slow elements are to be calculated (in parallel, expecting the completion of the latter) with the usual step Δt. If not, – they are calculated with the reduced one $\Delta\tau$.
4. For executing the next modeling step – we increase the model time by the past modeling step and move to the beginning of our algorithm (item 1).

We may develop a rather simple program under these rules, which may organize the running for any object of the "model" species of structure class.

5.2 A Complex of Models

We can unite some models into the complex of them, and it may turn out (but not mandatory) that some models in it calculate the external variables of some others as their internal ones.

To describe the complex of models, we need to specify:

1. The types and the numbers of exemplars of the models included.
2. Connections among the models inside the complex, so, which the internal variables of which components match the external ones of which other models.

Then, the complex turns out to be an object of the "model" species of structure (Brodsky, 2015). We may enter the following operation, uniting the objects of "model" species of structure into the complexes:

1. The set of the complex internal variables is the union of the ones of its models.
2. The set of the complex processes is the union of the ones of its models.
3. The set of the complex methods is the union of all the ones of its models.
4. The set of the complex events is the union of all the ones of its models.
5. The set of the complex external variables is the union of all the ones of its models, except those that are explicitly calculated by them.

This operation with some clarifications (Brodsky, 2015), turns the complex of "model" species of structure objects into a mathematical object of the same "model" species of structure. So, we can build very complex fractal self-similarly constructions, and only the developer's desire limits its complexity.

5.3 Model-Oriented Programming

Here we propose a new programming paradigm for large-scale systems simulating – a Model-Oriented one, and the theory of Model Synthesis is the base of this technology. The Model Synthesis "model" is a structure more complex than the Object Analysis object and its encapsulation level is higher. The most significant difference from the object – its own complete behavior, – the ability to respond on standard requests in a standard manner, while the object has only potential behavior – the set of the methods – the skills which are to be activated from the outside to manifest. Moreover, the arrangement of

the model's behavior is permanent for all the models, regardless how huge and complicated it is, no matter of the model's domain.

The proposed programming paradigm has only one base concept – a "model" species of structure, – a minimum of bases. It has also an auxiliary idea – a complex, which turns out to be of the same "model" species of structure itself, hence we may include it in the covering complexes as a component.

Any model runs under the permanent rules, so, the universal program of models' execution can perform it, once developed and debugged. In addition, this program is oriented on parallelization of calculations. The developing of the software system breaks down

1. Into the set of declarative specifications of models and complexes.
2. On the programming of the methods – elements of the processes, which are mathematical functions (unique mappings: no states, nor side effects), so they may be developed using functional programming methods.

So, the software development completely excludes the imperative paradigm – the most hard as when elaborating, either at the debugging stage. In addition, once the model is realized and debugged, we can forget about its internal construction, and use it as a ready CAD functional block in all the future designs. The only condition is needed – correctly connect outputs with the inputs – and it will operate correctly, and will behave exactly how it used to in any complexes which include it.

Everything happens the same as in the CAD systems, for example if our chip is contained in a micro assembly, and a circuit board contains this micro assembly. If all the commutations are correct at each project level, the chip will function adequately. The fee is a level of encapsulation of the "model", which grew up in comparison with the object-oriented concept, so, the MO-programming usually pays off only in large projects. As opposed to the object paradigm, it is forbidden (because of useless) to call the methods "manually" from outside the model with "alien", even fitting arguments. The execution program automatically calls all the methods, with the part of model's characteristics as arguments only, following the "model" behavior program, described in its declarative specification.

We offered the LCCD programming language (the language for components and complexes descripting) to define model components and model complexes. The work (Brodsky, 1991) describes this language in detail. It is a declarative one. The specifications of LCCD are to be compiled into the model database – not in the machine code. That fact removes all the questions about the quality of the resulting code – only the compilation correctness is important when compiling into the database.

The MISS tool system (Brodsky, 1991) was an implementation of the Model-Oriented programming paradigm, proposed here. This system won the first prize in the programist contest undertaken in the former USSR by the ASCII corporation (ASCII, 2023). – Japanese company, which had leading positions in computer games in 90th, but now is a Manga publishing company division. The research works of elaborating the MO-programming system oriented on high-performance and distributed computations are running in the Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences.

6 CONCLUSIONS

It turns out, no matter how complex is the domain of a large-scale complex model, its conceptual model may be rather simple and permanent for a very wide class of domains. It contains a constant dynamic part – a four-step execution program for the calculations organizing, and a static part – a database, which changes quantitatively, but not qualitatively with the model changes – the types of most tables in it are invariable. Such an organization of the conceptual model implements the well-known formula "One in all and all in One".

The properties of this universal conceptual model make it possible to propose a through-the-line technology for describing, synthesizing, and software developing of large-scale models – Model Synthesis and Model-Oriented Programming.

The concept of MO-programming differs from the known concepts of model-driven software development: MDA – model-driven architecture (MDA, 2023), MDE – model-driven engineering (Shmidt, 2006), and MDD – model-driven development (Beydeda, 2005), no matter the likeness of names. All these concepts are only superstructures over the object-oriented programming concept, so, they could not make the programming a technology instead of the art, – the ultimate tools for the software development are still imperative Java- or C-like languages in these concepts.

The MO-programming completely implements the CAD methods in the software design and coding, with all the advantages and drawbacks inherent in these methods. The programming at all the stages and levels of the project are completely declarative, – this fact greatly simplifies the following debugging.

To give a system designer more human-oriented tool then the species of structure, for specifying the model and its parts, a declarative programming language was elaborated. Its name is LCCD – language for components and complexes describing. It describes the "models" and complexes built of them. An essential specificity of the MO-programming is that the LCCD programs are compiled not into the computer code, but database tables. Poor code quality after double compilation kills the UML (Booch, 2005), (Fowler, 2004) and model-driven software development concepts linked with it. In the MO-programming, the code quality is not a problem – the language specifications are compiled into the DB tables either right or wrong. The efficiency of computations depends upon a calculations organizing universal program and the code quality of the processes' elements and events. The DB tables are the third way of the "model" description, more fitting the computations organizing program.

We should note that the technology proposed for specification, designing, and software realizing for the simulation of large-scale systems is not a kind of unrelated theories. On the contrary, all began from the complete MO-programming system realization (Brodsky, 1991), while the first theoretical justification of the method proposed appeared later, in (Brodsky, 2014), (Brodsky, 2015).

The proposed formal language of complex systems structural description can be applied to the formalization of descriptive sciences problem. We can use it to describe almost any agent–based social system – a country, a political party, a social stratum, an army or its subdivision, thanks to the algorithmic completeness of the basic construction of the language – the "model" species of structure in N. Bourbaki sence.

REFERENCES

Hewitt, C., 1977. Viewing Control Structures as Patterns of Passing Messages. *Journal of Artificial Intelligence.* 6.

Shoham, Y., 1933. Agent-oriented programming. *Artificial Intelligence,* vol. 60. 51-92.

Brodsky, Yu.I., Lebedev, V.Yu., 1991. *Instrumental'naya sistema imitatsionnogo modelirovaniya MISS* [Instrumental Simulation System MISS]. CC AS of the USSR, Moscow.

Brodsky, Yu.I., 2014. Model synthesis and model-oriented programming – the technology of design and implementation of simulation models of complex multi-component systems. *In the World of Scientific Discoveries, Series B*, vol. 2, No 1, 12-31.

Brodsky, Yu.I., 2015. Bourbaki's Structure Theory in the Problem of Complex Systems Simulation Models Synthesis and Model-Oriented Programming. *Computational Mathematics and Mathematical Physics*, vol. 55, No. 1, 148-159.

Bourbaki, N., 2004. *Elements of Mathematics, Theory of Set. Springer*, Heidelberg.

MacRae, N., 1999. *John von Neumann: The Scientific Genius Who Pioneered the Modern Computer, Game Theory, Nuclear Deterrence, and Much More.* Providence, 2nd ed. RI: Amer. Math. Soc., 406 p

Thomson, J.F., 1954. Tasks and Super-Tasks. *Analysis,* 15(1), 1-13.

Silberschatz, A., Galvin, P., Gagne, G., 2018. *Operating System Concepts. 10th edition*, Willey & Sons.

ASCII Corporation. https://en.wikipedia.org/wiki/ASCII_Corporation (Retrieved 4 June 2022).

MDA Guide revision 2.0 https://www.omg.org/cgi-bin/doc?ormsc/14-06-01.pdf (Retrieved 4 June 2022).

Schmidt, D.C., 2006. Model-Driven Engineering. *IEEE Computer,* 39 (2).

Beydeda, V, Book, M., Gruhn, V., 2005. *Model-Driven Software Development.* Berlin, Heidelberg: Springer-Verlag, 464 p.

Booch, G., Rumbaugh, J., Jacobson, I., 2005.*UML. User Guide*, 2nd ed.. Addison-Wesley.

Fowler, M., 2004.*UML Distilled*, 3-ed., Addison-Wesley, 192 p.