MODEL-ORIENTED PROGRAMMING

Leonid Kruglov[1], **Yury Brodsky**[2]

Abstract: The problem of complex multi-component system processing arises in many fields of science and engineering. A system can be described in terms of its components, behavior, and interaction. This work proposes a new declarative Turing complete "model-oriented" programming paradigm based on the concept of "model-component" - a complex structure with well-defined characteristics and behavior, and no external methods. The set of model-components is closed under the union operation of model-components into "model-complex". The proposed approach allows the program to describe the complex system and behavior of its components in a declarative way, possesses a higher level of encapsulation than the object-oriented paradigm, involves a reduced amount of imperative programming, and is naturally focused on parallel computations.

UDC Classification: 004.42, **DOI:** https://doi.org/10.12955/pns.v2.154

Keywords: model-oriented programming, complex systems, model synthesis, simulation.

Introduction

Nowadays, the processing of complex systems plays an important role in many fields of science and engineering – from demographics and economics (Belotelov et al., 2008) to biology, physics (Auyang, 1998), and software development (Jennings, 2001). In many cases, modeling and simulation are the only way to analyze the behavior and characteristics of such systems.

As defined in the paper by Brodsky (2015), a complex system can be viewed as a compound object whose parts can be viewed as systems naturally united together according to certain principles or defined sets of relations. It means that components of a complex system can be complex systems themselves and that the arrangements of system "atoms", their behavior, and the rules of their interaction are well known. Once a formal description of the system is available, computer realization of its simulation model is possible (Brodsky, 2013, 2014). Due to the complexity of the problem, it is preferred that the design and realization of the model are performed in some automated way to reduce the time spent on programming and the number of potential errors.

Simulation modeling

One possible approach to simulation modeling is the deductive approach which builds the model from top to bottom, starting from general ideas and concepts and gradually moving to more specific cases. In software development, this method can be related to object-oriented programming. It allows creating new models by building hierarchies of classes or applying class composition to reflect the structure of the systems under analysis. However, the organization of computational processes requires the application of an imperative paradigm by the developer to create the desired sequence of object method calls.

The inductive approach is the opposite way of solving the problem. It takes advantage of the atomistic nature of complex systems where each component has its well-defined fixed characteristics and behavior, and the components often may not have a strict hierarchy. This approach works from bottom to top, starting from basic primitives and generalizing them into the whole model of a system. In this work, we will focus on the inductive approach and show how it can be applied to simulation modeling and that it can also be considered as a standalone programming paradigm.

Proposed approach

In this work, we propose a new declarative Turing complete "model-oriented" programming paradigm which is based on the concepts of "model-component" and "model synthesis" (Brodsky, 2013). A model-component is a complex structure with well-defined characteristics and behavior and with no external methods. Model-components are used as building blocks to synthesize the model of the whole system. This process is feasible since the set of model-components is closed under the union operation of model-components into a "model-complex". Hence, the whole system can be viewed as a single model-component, and each model-component can be processed by a common universal procedure. The

[1] Moscow Pedagogical State University, Institute of Mathematics and Informatics, Moscow, Russian Federation, lv_kruglov@student.mpgu.edu, ORCID 0000-0001-6510-1740

[2] Federal Research Centre "Computer Science and Control" of RAS, yury_brodsky@mail.ru, ORCID 0000-0002-0565-4957

behavior of each model-component can be described in terms of transitions between its internal states and corresponding internal methods, which can be executed in parallel. Such complex knowledge of the system and the behavior of its components can be programmed in a declarative way, and only internal methods may require limited imperative programming (if convenient).

Model-component

A "model-component" of a system is the main concept of the proposed approach. It can be considered as a building block of the system which has some characteristics and behavior. Several model-components can be combined into an aggregated structure called "model-complex", which can be viewed as a model-component itself. Hence the whole system under analysis can be considered as a model-component.

Each model-component has a set of internal and external characteristics and several internal methods. Internal characteristics can be considered as the states of the model-component. External characteristics are the parameters of the environment that affect the behavior of the model-component. Internal characteristics may depend on each other as well as on external characteristics. In contrast to the object-oriented approach, methods of the model-component cannot be called from the outside. The behavior of the model-component is defined by its internal characteristics and corresponding internal methods, and it can be changed by the occurrence of events – points where the response from the system is required to reflect the changes of the environment (the external characteristics). Several methods can be executed simultaneously (by parallel processes) as they are not allowed to change the same characteristics. So-called "fast" methods occur instantly concerning modeling time and are used to calculate discrete characteristics of the system. "Slow" methods require non-zero modeling time and are used to calculate continuous characteristics of the system.

A model of a system can be formally defined as an algorithm F which determines the values of internal characteristics at the next step of the modeling process (Brodsky & Pavlovsky, 2009):

$$x_{i+1} = F(x_i, a_i, \Delta t),$$

where $x_i$is the vector of internal characteristics at a time t_i, a_i is the vector of external characteristics at a time t_i, $\Delta t = t_{i+1} - t_i$ is the modeling time interval.

Closeness hypothesis

The existence of such an algorithm means that the system is closed: it is assumed that we have enough knowledge of the system to construct its model which can reproduce its dynamics (i.e. its internal characteristics x_i) at every point of modeling time t_i of some time segment $[0, T]$ if its initial state x_0 and external characteristics a_i are known (Brodsky, 2013).

More formally, the model is called (locally) closed at time point $t_i \in [0, T)$ if there exists $\Delta t > 0, t_i + \Delta t \in (0, T]$ such that:

1. it is possible to infer from internal characteristics $x(t_i)$ and external characteristics $a(t_i)$ if there exists calculable discontinuity (a gap) $\Delta x(t_i)$ at point t_i,
2. on the interval $(t_i, t_i + \Delta t]$ $x(t)$ is a continuous function of time with value $x(t_i)$+$\Delta x(t_i)$ at point t_i.

Hence, "fast" methods of a model-component are used to calculate the gaps, and "slow" methods are used to calculate continuous intervals of $x(t)$ (Brodsky, 2013).

The local closeness of a model is the necessary condition of model realization: if there exists a point at which even a small-time step is not possible, then it is unclear how to build the model. However, this condition is not sufficient (Brodsky 2013, 2014). The counterexample is the fly from the "train fly problem" whose speed has two limits at the time point when both trains meet. The additional requirement of left continuity of internal characteristics $x(t_i)$ at any time point $t_i \in [0, T)$ make the condition sufficient (Brodsky 2013, 2014). This requirement means that any state of the model, except the initial, is the result of its prehistory.

Based on the described conditions (namely functional dependency of the gap of internal characteristics, their continuous evolution, and the need to calculate the gap at each modeling step), a common universal procedure for processing any model-component can be defined.

Events

Events are the points of synchronization between methods of the model-component – points at which the values of internal and external characteristics require the reaction of the model-component. Formally, the event is a function of the model internal and external characteristics $\Delta\tau = E(x, a)$ which returns the predicted time of its occurrence at the beginning of the simulation step. Occurrence of the event at a time t_i is defined by the equation:

$$E(x_i, a_i) = 0.$$

Events control the switch between methods of the model-component: for each ordered pair of switchable methods $\{f_1, f_2\}$ there exists a unique event $E_{\{f_1,f_2\}}$that defines the switch. Simultaneous occurrence of events $E_{\{f_1,f_2\}}$ and $E_{\{f_1,f_3\}}$ indicates an error in the design of the model.

Model behavior

The behavior of a model component is given by a procedure that defines the rules of its simulation calculations. It is assumed that at the beginning of every simulation step, all internal characteristics and current methods are known, and external characteristics can be observed at any time point. An appropriate default modeling time step Δt is fixed. The procedure implements the following algorithm:

1. Compute the events which are associated with each current method f_m: $\Delta\tau_i = E_{\{f_m,f_n\}}(x_i, a_i)$.
2. If any events corresponding to fast methods occur (i.e. some $\Delta\tau_j = 0$), switch current methods accordingly, execute the methods and go to step 1.
3. If any events corresponding to slow methods occur, switch current methods accordingly.
4. Calculate the next modeling time step $\Delta\tau = min(\Delta t, min_i \Delta\tau_i)$.
5. Execute all current slow methods with calculated modeling time step $\Delta\tau$, go to step 1.

The algorithm defines a common universal procedure for the execution process of any model-component.

Model-complex

Several models-components can be combined into a higher-level element called “model-complex”, where some model-components may explicitly model external characteristics of some other model-components. To describe a model-complex it is enough to specify:

1. which models-components are included in the model-complex,
2. the number of included model-components of each type,
3. commutation of models-components inside the model-complex: which internal characteristics of which models-components are the external characteristics of model-components.

When combining model-components into a model-complex, the unambiguity of the computational process may be lost. This might happen if, for some reason, several components calculate the same internal characteristics of the system under analysis. In this case, a new model-component should be included in the model-complex that treats all these characteristics as external characteristics and deterministically calculates the only value, which in turn represents a new internal characteristic of the complex (Brodsky, 2013).

Model-complex synthesis

A model-complex consisting of many models-components can be viewed as a single model-component. The union operation of model-components into a model-complex is defined by the following rules (Brodsky, 2013):

1. The internal characteristics of the complex are the union of the internal characteristics of all its components.
2. The processes of the complex are the union of the processes of all its components.
3. The methods of the complex are the union of the methods of all its components.
4. The events of the complex are the union of the events of all its components.
5. The external characteristics of the complex are the union of the external characteristics of all its components, except the ones that are explicitly modeled by some component of the complex.

The fact that the set of model-components is closed under the union operation (i.e. that a model-complex can be treated as a model-component) is the basic idea of model synthesis, which allows developers to design and implement simulation models of complex multi-component systems of any required nesting and guarantees the existence of the aforementioned universal processing procedure.

Model-oriented paradigm

The proposed model-oriented approach allows designing and building programs from more aggregate elements (model-components) with a higher level of encapsulation than the object-oriented approach: methods of a model-component are internal and cannot be called manually; they are always executed automatically, thus demonstrating the behavior of the model. An important aspect of the approach is that the methods can be executed in parallel at each iteration – it is guaranteed by the fact that the methods represent functional dependencies and are not allowed to change the same internal characteristics.

As each synthesized model-complex (and hence the whole system itself) can be viewed as a model-component, the computational process of model behavior can be implemented as a common universal procedure that allows parallelization of most calculations. All this allows creating models of systems with deep and complex nesting.

The model-oriented paradigm splits the process of programming into two main steps:

1. Declarative programming of model-components and model-complexes.
2. Functional (or imperative if convenient) programming of internal methods.

Hence the structure and interaction of components of the system under analysis are programmed in a declarative way which makes it possible to significantly reduce the time spent on programming and the number of potential errors compared to the imperative approach (Brodsky, 2013). The proposed approach paradigm is Turing complete which means that it can be successfully applied to a wider class of problems than just modeling of complex systems.

Turing completeness

We will show Turing completeness of the proposed model-oriented approach by the realization of Turing machine with semi-infinite tape (which is equivalent to Turing machine with finite tape as shown by Karpov (2003)) using the concept of model-component.

1. Set of values $Q = \{q_0, q_1, \dots, q_m\}$ of internal characteristic x_q of the model-component represent the set of *states* of the Turing machine.
2. $x_q = q_0$ is the *initial state*.
3. $F \subseteq Q$ is the set of *final states*.
4. Internal characteristics $X = \{x_0, x_1, \dots\}$ represent the *semi-infinite tape*.
5. Internal characteristic $x_c \in \{0,1, \dots\}$ indicates the current cell of the tape, $x_{x_c} \in X$.
6. $x_c = k, k \in \{0,1, \dots\}$ indicates the initial cell of the tape.
7. The set Γ of allowed values of internal characteristics X represents the set of *tape alphabet symbols*.
8. $b \in \Gamma$ is the *blank symbol*.
9. The set $\Sigma \in \Gamma \backslash \{b\}$ of allowed values of internal characteristics X represent the set of *input symbols*.
10. The set $\underline{X} = \{x_i, \dots, x_j\} \subset X$ represents the initialized tape cells, $x_b \notin \underline{X}$ are initially blank.
11. *The transition function* is represented by the "fast" internal method $f_t(x_q, x_t, X)$ which changes the value of the current cell for the current state x_q and value x_{x_c} of current cell x_c, updates the value of the current state and either increments, decrements, or does not modify the index x_c of the current cell.
12. Execution of transition method f_t is triggered by an empty event E_{f_t,f_t} at the beginning of each modeling step.
13. Execution is terminated if the value of the current state characteristic represents some final state, i.e. $x_q = f \in F$

This construction provides the Turing machine with semi-infinite tape, thus proving Turing completeness of the proposed model-oriented programming paradigm.

Conclusion

In this paper, a new model-oriented programming paradigm has been proposed. It is a declarative approach that possesses a higher level of encapsulation than the object-oriented paradigm, involves a reduced amount of imperative programming, and is naturally focused on parallel computations. The

proposed approach is Turing complete and is applicable for a wide class of problems, including simulation modeling of complex multi-component systems.

The model-oriented paradigm has been successfully used to implement models of complex systems (Brodsky, 2013), and several tools for system simulation have been created (Brodsky & Lebedev, 1991; Brodsky, 2010). The scope of future work includes further development of the programming system for the model-oriented paradigm and extending its mathematical basis.

References

Auyang, S. Y. (1998). Foundations of complex-system theories: in economics, evolutionary biology, and statistical physics. Cambridge University Press. https://doi.org/10.1017/CBO9780511626135

Belotelov, N. V., Brodsky, Yu. I, & Pavlovsky, Yu. N. (2008). Komp'yuternoe modelirovanie demograficheskih, migracionnyh, ekologo-ekonomicheskih processov sredstvami raspredelennyh vychislenij [Computer modeling of demographic, migrational and ecological-economical processes by means of distributed calculations]. Moscow: CC RAS.

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

Brodsky, Yu. I. (2010). Raspredelennoe imitatsionnoe modelirovanie slozhnych sistem [Distributed simulation of complex systems] Moscow: CC RAS.

Brodsky, Yu. I. (2013). Model'nyj sintez i model'no-orientirovannoe programmirovanie [Model synthesis and model-oriented programming]. Moscow: CC RAS.

Brodsky, Yu. I. (2014). Model synthesis and model-oriented programming-the technology of design and implementation of simulation models of complex multicomponent systems. In the World of Scientific Discoveries, Series B, 2(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, 55(1), 148-159. https://doi.org/10.1134/s0965542515010054

Brodsky, Yu. I., & Pavlovsky, Yu. N. (2009). Razrabotka instrumental'noj sistemy raspredelennogo imitacionnogo modelirovaniya [Development of instrumental system of distributed simulation modeling]. Journal of Information Technologies and Computing Systems, (4), 9-21.

Jennings, N. R. (2001). An agent-based approach for building complex software systems. Communications of the ACM, 44(4), 35-41. https://doi.org/10.1145/367211.367250

Karpov, Yu. G. (2003). Teoriya avtomatov: Uchebnik dlya vuzov [Automata theory: Course book for higher educational institutions]. Saint Petersburg: Piter.