Исследования и разработки в области новых информационных технологий и их приложений

https://doi.org/10.25559/SITITO.021.202503.445-454
УДК 004.942

Разработка и симуляция цифровых моделей на основе языка DLAA в среде онтологической платформы

Оригинальная статья

Д. А. Гапанович*, В. М. Дубков, М. А. Князев, В. А. Сухомлин

ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова», г. Москва, Российская Федерация

Адрес: 119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1

* dim.gapanovich@gmail.com

Аннотация

Статья посвящена вопросам погружения средств моделирования визуального графического языка диаграмм состояний и переходов конечных автоматов специального вида (DLAA), предназначенного для структурированного описания конфигурации и поведения дискретно-событийных систем, в среду онтологической платформы. Такой вектор развития средств моделирования позволяет создать графический конструктор моделей высокого уровня, поддерживающий непрерывную проверку корректности конструируемых моделей с помощью логического аппарата онтологической базы знаний, а также использовать возможности логической машины вывода онтологической платформы для анализа сложных архитектурных представлений моделируемых систем. В статье рассмотрены основные решения поставленной задачи, а именно, разработка: онтологической схемы языка DLAA в среде онтологической платформы OSA, графического конструктора моделей, конвертора онтологического образа модели в представление модели на базовом языке, интерпретируемое симулятором DLAA.

Ключевые слова: язык графического моделирования, DLAA, теория автоматов, моделирование дискретных событий, конечные автоматы, автоматное моделирование, диаграммы переходов и состояний, онтология, онтологическая схема, онтологическое моделирование, онтологическая платформа

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Для цитирования: Разработка и симуляция цифровых моделей на основе языка DLAA в среде онтологической платформы / Д. А. Гапанович [и др.] // Современные информационные технологии и ИТ-образование. 2025. Т. 21, № 3. С. 445-454. https://doi.org/10.25559/SITITO.021.202503.445-454

Research and Development in the Field of New IT and Their Applications

Development and Simulation of Digital Models Based on the DLAA Language in an Ontological Platform Environment

Original article

D. A. Gapanovich*, V. M. Dubkov, M. A. Knyazev, V. A. Sukhomlin
Lomonosov Moscow State University, Moscow, Russian Federation
Address: 1 Leninskie Gory, Moscow 119991, GSP-1, Russian Federation
* dim.gapanovich@gmail.com

Abstract

The article is devoted to the issues of integrating modeling tools for the visual graphical language of state and transition diagrams for finite automata of a special type, DLAA, intended for the structured description of the configuration and behavior of discrete-event systems, into an ontological platform environment. This direction in the development of modeling tools makes it possible to create a high-level graphical model builder that supports continuous correctness checking of the models being constructed using the logical apparatus of an ontological knowledge base, as well as to use the capabilities of the ontological platform's logical inference engine to analyze complex architectural representations of the modeled systems.
The article considers the main solutions to the task, namely the development of: an ontological schema of the DLAA language within the OSA ontological platform environment, a graphical model builder, and a converter that transforms the ontological image of a model into a model representation in the base language interpreted by the DLAA simulator.

Keywords: graphical modeling language, DLAA, automata theory, discrete-event simulation, finite automata, automata-based modeling, state and transition diagrams, ontology, ontological schema, ontological modeling, ontological platform

Conflict of interests: The authors declares no conflict of interest.

For citation: Gapanovich D.A., Dubkov V.M., Knyazev M.A., Sukhomlin V.A. Development and Simulation of Digital Models Based on the DLAA Language in an Ontological Platform Environment. *Modern Information Technologies and IT-Education*. 2025;21(3):445-454. https://doi.org/10.25559/SITITO.021.202503.445-454

1 Введение

На основе исследований методов и средств описания структурных и поведенческих аспектов систем, основанных на теории конечных автоматов, в работе [1] предложена специальная конфигурация автомата, которая использовалась при разработке языка визуального графического моделирования поведенческих аспектов динамических дискретно-событийных систем DLAA. Данный язык характеризуется компактной концептуальной базой, обладая при этом широкими возможностями для описания динамических аспектов событийных систем, поддержкой структурированной технологии разработки моделей, лёгкостью использования языка. Также он легко реализуется в среде языков программирования методом обогащения проблемно-ориентированными возможностями базовых сред программирования. Для этого языка созданы метатранслятор и симулятор, позволяющие разрабатывать и исполнять поведенческие модели динамических систем в модельном времени в среде базовых языков программирования таких, как C++ и C# [2].

В статье кратко рассмотрены решения по интеграции средств имитационного моделирования языка DLAA с онтологической платформой OSA [3], включая:

- трансформацию концептуального базиса языка DLAA в соответствующую онтологическую схему на онтологической платформе OSA;
- разработку графического конструктора в виде приложения на платформе OSA, позволяющего проектировать в графическом режиме модели дискретно-событийных систем на языке DLAA и выполнять непрерывную логическую проверку создаваемых моделей с помощью движка логического вывода онтологической платформы, а также формировать онтологический образ проектируемой модели;
- конвертор онтологического образа модели в конструкции базового языка (C++), интерпретируемые симулятором.

2 Онтологическая схема языка DLAA

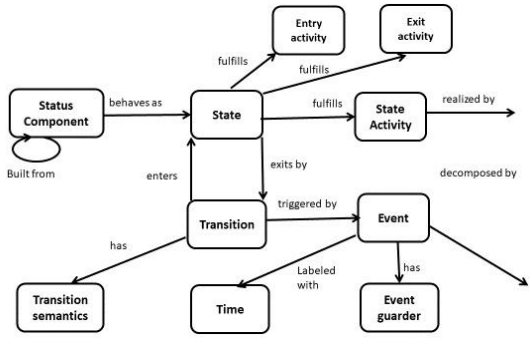
В процессе разработки языка DLAA были разработаны концептуальные модели для статической и динамической структуры моделируемых систем, составляющие базовые концепты языка DLAA.

Статическая структура разрабатывалась на основе типовой архитектуры Интернета вещей [4].

Концептуальная модель динамической структуры включала набор понятий, характерных для описания структурных и поведенческих аспектов динамических систем [5, 6], таких, как, например:

- Time (время)
- Event (событие)
- Event parameters (параметры события)
- State (состояние)
- Transition (переход)
- Transition semantics (семантика перехода)
- Event guarder (предусловие события или сторожевой предикат)
- Ресурс (resource)
- State activity (активность состояния)
- Entry activity (входная активность)
- Do activity (рабочая активность)
- Exit activity (выходная активность)
- Mode (режим) и некоторые другие.

Для выше перечисленных понятий на рис. 1 представлена UML-диаграмма, описывающая концептуальную основу динамического представления моделируемых систем.



Р и с. 1. Концептуальная модель динамической структуры моделей систем
F i g. 1. Conceptual model of the dynamic structure of system models

Источник: здесь и далее в статье все таблицы и рисунки составлены авторами.
Source: Hereinafter in this article all tables and figures were made by the authors.

При разработке языка DLAA за основу принята следующая формальная модель автомата [1]:

$$\{FSM_d=<Ex_d, Ey_d, I_d, S_d, F_d, q_{init}, \delta_{int(d)}, \delta_{ext(d)}, \lambda_{st(d)}, \lambda_{tr(d)}, Ta_d, TS_d, \mu_d, Ю_d, Sem_d>\} \quad (1)$$

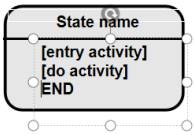
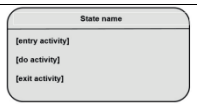
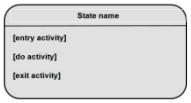
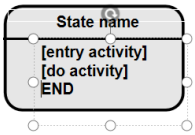
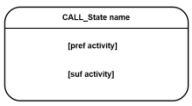
где

Ex – множество внешних входящих событий автомата
Ey – множество внешних исходящих событий автомата
*EI = {i, *, **} – множество внутренних событий автомата*
S – множество состояний автомата
F – множество конечных состояний автомата
qinit – начальное состояние автомата
δint – функция внутреннего перехода
δext – функция внешнего перехода
λst – функция вывода по состоянию
λtr – функция вывода по переходу
Ta – функция длительности пребывания в состоянии
TS – структура времени автомата
μ_d, $Ю_d$, Sem_d - *средства расширение языка семантическими программами*

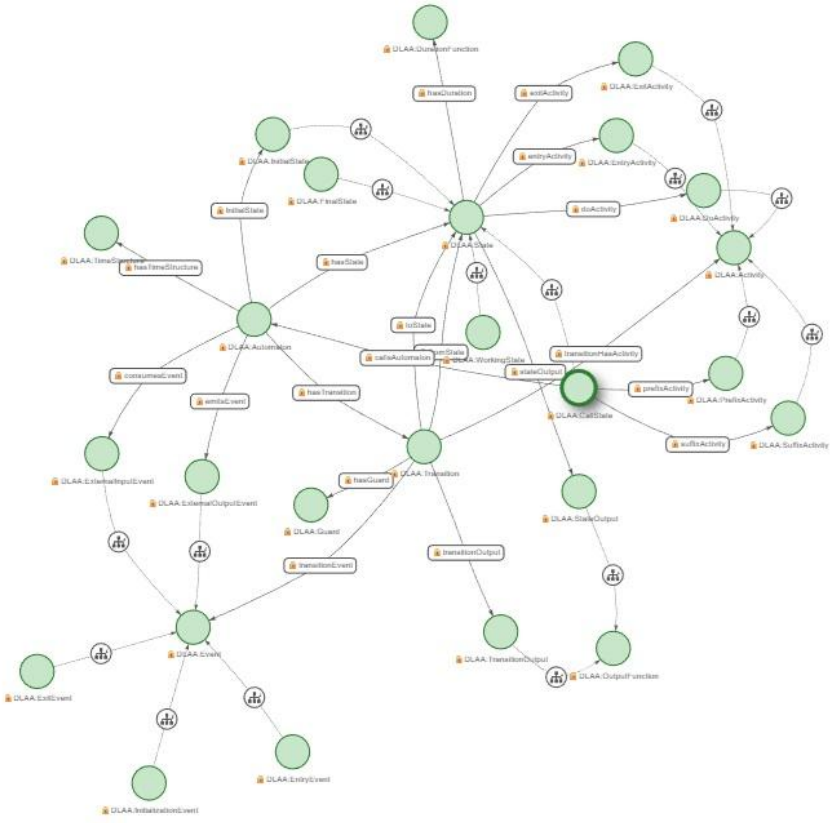
Для трансформации концептуальной модели языка DLAA в онтологическую схему разработаны состав семантических свойств концептов языка и их представление в онтологической схеме с помощью средств платформы OSA, аналогичных возможностям Protege.

Такие свойства отражены в Таблице 1.

Т а б л и ц а 1. **Семантические свойства концептов языка DLAA**
T a b l e 1. **Semantic properties of the DLAA language concepts**

Понятие	**Тип**	**Атрибуты**	**Связи**
Автомат		**имя автомата** **S** – множество состояний автомата **Ex** – множество внешних входящих событий автомата **Ey** – множество внешних исходящих событий автомата EI ={i, *, **} – множество внутренних событий автомата **F** – множество конечных состояний автомата q_{init} – начальное состояние автомата δ_{int} – функция внутреннего перехода δ_{ext} – функция внешнего перехода λ_{st} – функция вывода по состоянию λ_{tr} – функция вывода по переходу T_a – функция длительности пребывания в состоянии TS – структура времени автомата	Связи между автоматами: -внешние исходящие события - внешние входящие события - вызов автомата CALL и возврат по стеку из конечного узла  и 
Состояние (графическое представление состояния в виде прямоугольника с закруглёнными углами (аналогично тому, как принято в языке SysML), внутри которого помещается имя состояния)	Начальное Рабочее Конечное Вызов автомата	**Начальное** 1)имя состояния 2)ta - длительность пребывания в состоянии 3)активность типа entry 4)активность типа do 5)активность типа exit **Рабочее** 1)имя состояния 2)активность типа entry 3)активность типа do 4)активность типа exit **Конечное** 1)имя состояния 2)активность типа entry 3)активность типа do 4)активность типа exit **Вызов** 1)имя начального состояния автомата 2)активность префикс 3)активность суффикс	   
Переход и разметка перехода (стрелка, дуга) **<разметка>** 	Переход по внешнему событию Переход по внутреннему событию	**Разметка перехода:** 1)Состояние-исток 2)Состояние-сток 3)Имя сторожевого предиката 4)Имя внешнего входящего события (переход) - [имя внутреннего события перехода - **] //умалчивается 5) λ_{tr} – функция вывода по переходу::=<список элементов вывода>, где <элемент вывода>: !«текст вывода»| !<имя_переменной>| !!<имя внешнего исходящего события> //возбуждение внешнего исходящего события| **?**<имя внешнего входящего события> //ожидание внешнего события на переходе	Сторожевой предикат Состояние-исток Состояние-сток Имя внешнего входящего события Функция вывода по переходу Имя внешнего входящего события Имя внешнего исходящего события
Событие	внешнее входящее внешнее исходящее *i* - событие инициализации автомата * - событие входа в состояние ** - событие выхода из состояния	1) std::string name - имя события 2) int priority - приоритет события TS time - время планируемого события - динамика Используется симуляторов для входа в автомат Используется симуляторов для входа в новое состояние автомата Используется симуляторов для выхода из текущего состояния автомата	δ_{ext} – функция внешнего перехода **Ex** – множество внешних входящих событий автомата **Ey** – множество внешних исходящих событий автомата
δ_{int}– функция внутреннего перехода	δint: S → S, где S – множество состояний автомата	Формируется на переходе созданием дуги между двумя состояниями	Определяет логически следующее состояние для каждого состояния в автомате

δ_{ext} – функция внешнего перехода	δext: SxEx → S	Формируется: а) на переходе по истоку и стоку перехода (дуги) и имени внешнего входящего события б) по входной стрелке в начальное состояние	Определяет переход к новое состояние по внешнему входящему событию
λst – функция вывода по состоянию		!«текст вывода»| !<имя_переменной>| !!<имя внешнего исходящего события> //возбуждение внешнего исходящего события| выводимая информация с помощью функции вывода при входе в состояние будет записываться в кавычках, перед которыми ставится восклицательный знак: • выводимая текстовая информация заключается в кавычки • вместо имени переменной выводится ее значение • сообщение об исходящем событии заключается в скобки, предваряемые восклицательным знаком	Выводит элементы вывода при входе в состояние
λ_{tr} – функция вывода по переходу		λ_{tr} – функция вывода по переходу::=<список элементов вывода>, где <элемент вывода>: !«текст вывода»| !<имя_переменной>| !!<имя внешнего исходящего события> //возбуждение внешнего исходящего события| **?**<имя внешнего входящего события> **//**ожидание внешнего события на переходе • выводимая текстовая информация заключается в кавычки, перед которыми ставится восклицательный знак • вместо имени переменной выводится ее значение • сообщение об исходящем событии предваряется двумя восклицательными знаками	Состояние-исток Состояние-сток Имя внешнего входящего события Функция вывода по переходу Имя внешнего входящего события
Ta – функция длительности пребывания в состоянии	функции продвижения Времени Ta = S→T	ta - атрибут состояния	Определяет атрибут длительности для каждого состояния в автомате
Функция распределения активностей по состояниям μ	μ: S -> <Aentry, Ado, Aexit>| <Apref, Asuf	<Aentry, Ado, Aexit>| <Apref, Asuf>	<Aentry, Ado, Aexit>| <Apref, Asuf>



Р и с. 2. Онтологическая схема базовых концептов языка DLAA
F i g. 2. Ontological schema of the basic concepts of the DLAA language

На рис. 2 представлена итоговая онтологическая схема языка DLAA.

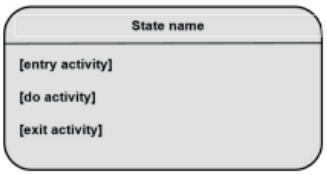
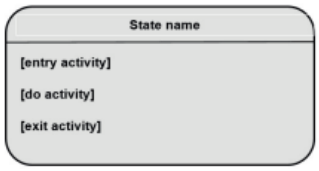
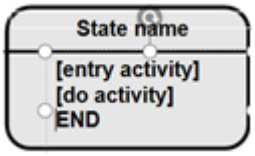
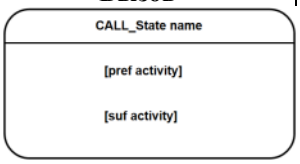
3 Графический конструктор в виде приложения на платформе OSA

Графический конструктор предназначен для построения в режиме графического редактора диаграммы состояний и переходов автомата, моделирующего некоторый элемент исследуемой системы, посредством размещения на рабочем поле экрана графических элементов языка DLAA, ввода их атрибутов и элементов разметки.

Основными графическими элементами языка DLAA являются:

- состояния и их разметка
- переходы и их разметка

Состояния и их разметка

Начальное	Рабочее	Конечное	Вызов
			
Начальное состояние 1) имя состояния 2) ta - длительность пребывания в состоянии 3) активность типа entry 4) активность типа do 5) активность типа exit	**Рабочее состояние** 1) имя состояния 2) активность типа entry 3) активность типа do 3) активность типа exit	**Конечное состояние автомата** 1) имя состояния 2) активность типа entry 3) активность типа do 4) активность типа exit	**Вызов автомата** 1) имя начального состояния автомата 2) активность префикс 3) активность суффикс

Переходы и их разметка

Графическое представление перехода это стрелка, соединяющая два состояния автомата, состояние-истока и состояние-стока.

<разметка>

Разметки перехода включает:

Разметка перехода:

1) Состояние-исток
2) Состояние-сток
3) Имя сторожевого предиката
4) Имя внешнего входящего события (переход)
- [имя внутреннего события перехода - **]
//умалчивается
5) λ_{tr} – функция вывода по переходу::=<список элементов вывода>, где
<элемент вывода>:
!«текст вывода»|
!<имя_переменной>|
!!<имя внешнего исходящего события> //возбуждение внешнего исходящего события|
?<имя внешнего входящего события> //ожидание внешнего события на переходе

В процессе работы конструктора формируется онтологический образ автомата, корректность которого проверяется непрерывно логическими средствами онтологической платформы [7-13].

Когда проектирование автомата завершается и онтологическая корректность образа подтверждается, онтологический образ передаётся на вход конвертору, задача которого трансформировать онтологический образ модели в конструкции базового языка (C++), интерпретируемые симулятором.

4 Конвертор онтологического образа модели в семантически эквивалентные конструкции базового языка

Трансформация конвертором онтологического образа модели в конструкции базового языка осуществляется на основе метасемантической таблицы 2, устанавливающей соответствие между элементами онтологической схемы языка и конструкциями базового языка, реализующими семантику языка при исполнении этих конструкций симулятором.

Полученное в результате трансформации онтологического образа модели в конструкции базового языка представление, определяет структурно-поведенческие аспекты исследуемой системы в терминах базового языка программирования и называется динамической структурой модели (ДСМ). Эта часть модели затем используется для компоновки с помощью конфигуратора полной модели на базовом языке посредством интеграции в единую программу (модель) ДСМ, исполняемую симулятором [14-25].

Т а б л и ц а 2. **Метасемантическая таблица соответствия элементов онтологического образа автомата конструкциям базового языка**

T a b l e 2. **Meta-semantic correspondence between elements of the automaton ontological image and base-language constructs**

Понятие	**Тип**	**Атрибуты**	**Конструкции языка**
Автомат		**имя автомата** **S** – множество состояний автомата **Ex** – множество внешних входящих событий автомата **Ey** – множество внешних исходящих событий автомата EI ={i, *, **} – множество внутренних событий автомата **F** – множество конечных состояний автомата q_{init} – начальное состояние автомата δ_{int} – функция внутреннего перехода δ_{ext} – функция внешнего перехода λ_{st} – функция вывода по состоянию λ_{tr} – функция вывода по переходу T_a – функция длительности пребывания в состоянии TS – структура времени автомата	Тип Имя //Определение std::string name //имя автомата std::vector<State> S //вектор состояний std::pair<State, TS> q_init //начальное состояние std::map<State, State> f_int //внутренние переходы std::map<State, std::string> lambda_st //вывод по состоянию std::map<std::pair<State, std::string>, std::string> lambda_tr //вывод по переходу std::map<State, TS> Ta //времена состояний std::vector<std::string> E_y //внешние исходящие события std::vector<std::string> E_x //внешние входящие события std::map<std::pair<State, std::string>, State>f_ext //переходы по внешним событиям std::map<std::pair<State, std::string>, State> new_events //новые события
Событие	внешнее входящее внешнее исходящее *i* - событие инициализации автомата * - событие входа в состояние ** - событие выхода из состояния	EventTime -динамическая характеристика EventName - имя события Priority - приоритет события EventType - тип события Sender = < MachineN> -динамическая характеристика Receiver = <MachineN> -динамическая характеристика	1) std::string name - имя события 2) int priority - приоритет события TS time - время планируемого события - динамика

5 Симулятор моделей

Исполнение программной модели, построенной конфигуратором на базовом языке, осуществляется симулятором, в состав которого входят следующие основные блоки:

EventPlaner – планировщик событий, реализующий интерпретацию структуры времени системы TS с помощью специальной структуры – синхромножества *(SynchroSet)*, представляющей собой список ссылок на уведомления запланированных событий, упорядоченный по времени наступления событий и приоритетам событий в случае сингулярного планирования последовательностей событий. При включении нового уведомления в синхромножество планировщих обеспечивает упорядоченность структуры по времени наступления событий.

Select – функция разрешения конфликтов в случае сингулярного планирования последовательностей событий в системе, результатом которой является ссылка на уведомление выбранного события.

EventExecutor – исполнитель цикла моделирования, который выбирает очередное уведомление о событии с наименьшим запланированным временем наступления (возможно с использованием функции Select), извлекает из него атрибуты события и исполняет его интерпретацию, следуя операционной семантике, определённой в логической модели симулятора. После чего процесс повторяется до тех пор пока в сихромноженстве имеются ссылки на уведомления о событиях или завершится время моделирования.

Основной цикл моделирования осуществляется с помощью функции **EventExecuter**, которая:

- выбирает из SynchroSet уведомление о событии с минимальным запланированным временем, если таких уведомлений больше одного, то они передаются функции Select, которая на основании стратегии выбора Select_STR определяет уведомление текущего события
- из уведомления текущего события извлекается имя события, время наступления события, имя автомата-назначения и тип события (CurrentEventName, Time, CurrentMachine, EventType):

```
Time = EventNotificat.EventTime, //
определение времени наступления события
CurrentEventName =
EventNotificat.EventName //имя
исполняемого события
CurrentMachine =
EventNotificat.RecieverMachineN //имя
автомата-назначения
EventType= EventNotificat.EventType//
тип планируемого события
CallingMachine=EventNotificat.SenderMach
ineN // автомат-источник сообщения
```

- анализируется тип события EventType и вызывается метод текущего автомата, соответствующий типу события

```
если EventType = TypeInit then
CurrentMachine.EvInit() // предикат
Init(i, t0)
если EventType = TypeStar then
CurrentMachine.EvStar() // предикат
Entr(*)
если EventType = TypeStarStar then
CurrentMachine.EvStarStar() // предикат
Do(**)
если EventType = TypeX then
CurrentMachine.EvX()// предикат Ext(x)
```

Заключение

Статья посвящена решению актуальной задачи, а именно интеграции средств имитационного моделирования дискретно-событийных систем для языка DLAA и возможностей онтологической платформы, что обеспечивает непрерывность проверки корректности диаграмм языка в процессе их построения с помощью логических средств онтологической платформы, создание онтологического образа проектируемой модели, а также возможность использования логической машины вывода онтологической платформы для анализа сложных архитектурных представлений моделируемых систем.

Рассмотрены основные решения этой задачи, а именно: создание онтологической схемы визуального графического языка диаграмм состояний и переходов конечных автоматов DLAA в среде онтологической платформы OSA, разработка графического конструктора моделей дискретно-событийных систем и конвертора онтологического образа модели в представление модели на базовом языке программирования, интерпретируемое симулятором, описана операционная семантика симулятора.

Развитие средств имитационного моделирования в направлении интеграции их с возможностями онтологической платформы выполнено с целью создания инструментов моделирования для поддержки проектной деятельности по курсу «Системная инженерия» программ магистерской подготовки факультета ВМК МГУ.

References

1. Gapanovich D.A., Sukhomlin V.A. Visual Graphical Interpreted Language for Discrete Modeling DLAA. *Modern Information Technologies and IT-Education.* 2025;21(1):76-89. (In Russ., abstract in Eng.) https://doi.org/10.25559/SITITO.021.202501.76-89
2. Gapanovich D.A., Sukhomlin V.A. Tools for Constructing Production Digital Twin Models Based on an Algebraic Approach and a Graphical State Language Extended by Functional and Operational Semantics. In: Balandin D., Barkalov K., Meyerov I. (eds.) Mathematical Modeling and Supercomputer Technologies. MMST 2024. *Communications in Computer and Information Science*. Vol. 2363. Cham: Springer; 2025. p. 3-16. https://doi.org/10.1007/978-3-031-80457-1_1
3. Volokitin Yu.I. Ontological Platform OSA. Available at: https://mbse.devtas.ru (accessed 29.07.2025). (In Russ.)
4. ISO/IEC 17789:2014. Information technology – Cloud computing – Reference architecture. Geneva: ISO; 2014.
5. Gapanovich D.A., Sukhomlin V.A. Algebra of Finite Automata as a Mathematical Model of the Digital Twin of Smart Production. *Modern Information Technologies and IT-Education*. 2022;18(2):353-366. (In Russ., abstract in Eng.) https://doi.org/10.25559/SITITO.18.202202.353-366
6. van Tendeloo Y., Vangheluwe H. DEVS: Discrete-Event Modelling and Simulation for Performance Analysis of Resource-Constrained Systems. In: Carreira P., Amaral V., Vangheluwe H. (eds.) Foundations of Multi-Paradigm Modelling for Cyber-Physical Systems. Cham: Springer; 2020. p. 127-153. https://doi.org/10.1007/978-3-030-43946-0_5
7. Zeigler B.P. Closure under coupling: concept, proofs, DEVS recent examples (wip). In: Proceedings of the 4th ACM International Conference of Computing for Engineering and Sciences (ICCES'18). New York, NY, USA: ACM; 2018. Article number: 7. https://doi.org/10.1145/3213187.3213194
8. Zeigler B.P., Muzy A., Kofman E. Theory of Modeling and Simulation: Discrete Event and Iterative System Computational Foundations. 3rd ed. London: Academic Press; 2018. https://doi.org/10.1016/C2016-0-03987-6
9. Gruber T.R. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*. 1993;5(2):199-220. https://doi.org/10.1006/knac.1993.1008
10. Uschold M., Gruninger M. Ontologies: Principles, Methods and Applications. The Knowledge Engineering Review. 1996;11(2):93-136. https://doi.org/10.1017/S0269888900007797
11. Musen M.A. The Protégé Project: A Look Back and a Look Forward. *AI Matters*. 2015;1(4):4-12. https://doi.org/10.1145/2757001.2757003
12. Gennari J.H., Musen M.A., Fergerson R.W., Grosso W.E., Crubézy M., Eriksson H., Noy N.F., Tu S.W. The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. *International Journal of Human-Computer Studies.* 2003;58(1):89-123. https://doi.org/10.1016/S1071-5819(02)00127-1
13. Silver G.A., Hassan O.H., Miller J.A. DeMO: An Ontology for Discrete-Event Modeling and Simulation. *Simulation*. 2011;87(9):747-773. https://doi.org/10.1177/0037549710386843
14. Hong K.J., Kim T.G. DEVSpecL: DEVS Specification Language for Modeling, Simulation and Analysis of Discrete Event Systems. *Information and Software Technology*. 2006;48(4):221-234. https://doi.org/10.1016/j.infsof.2005.04.008
15. Harel D. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*. 1987;8(3):231-274. https://doi.org/10.1016/0167-6423(87)90035-9
16. Al-Fedaghi S. Modeling the Semantics of States and State Machines. *Journal of Computer Science*. 2020;16(7):891-905. https://doi.org/10.3844/jcssp.2020.891.905
17. Wainer G.A. Discrete-Event Modeling and Simulation: A Practitioner's Approach. Boca Raton: CRC Press; 2017. https://doi.org/10.1201/9781420053371
18. OMG. Systems Modeling Language (OMG SysML), Version 1.6. Object Management Group; 2019. Available at: https://www.omg.org/spec/SysML/1.6/ (accessed 29.07.2025).
19. Grieves M., Vickers J. Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In: Kahlen F.-J., Flumerfelt S., Alves A. (eds.) Transdisciplinary Perspectives on Complex Systems. Cham: Springer; 2017. p. 85-113. https://doi.org/10.1007/978-3-319-38756-7_4
20. Lim K.Y.H., Zheng P., Chen C.-H. A State-of-the-Art Survey of Digital Twin: Techniques, Engineering Product Lifecycle Management and Business Innovation Perspectives. *Journal of Intelligent Manufacturing*. 2020;31(6):1313-1337. https://doi.org/10.1007/s10845-019-01512-w
21. Liu Y., Peng Y., Wang B., Yao S., Liu Z. Review on Cyber-Physical Systems. *IEEE/CAA Journal of Automatica Sinica.* 2017;4(1):27-40. https://doi.org/10.1109/JAS.2017.7510349
22. Schluse M., Priggemeyer M., Atorf L., Rossmann J. Experimentable Digital Twins – Streamlining Simulation-Based Systems Engineering for Industry 4.0. *IEEE Transactions on Industrial Informatics*. 2018;14(4):1722-1731. https://doi.org/10.1109/TII.2018.2804917
23. Redelinghuys A.J.H., Basson A.H., Kruger K. A Six-Layer Architecture for the Digital Twin: A Manufacturing Case Study Implementation. *Journal of Intelligent Manufacturing*. 2020;31(6):1383-1402. https://doi.org/10.1007/s10845-019-01516-6

24. Tsinarakis G., Sarantinoudis N., Arampatzis G. A Discrete Process Modelling and Simulation Methodology for Industrial Systems within the Concept of Digital Twins. *Applied Sciences*. 2022;12(2):870. https://doi.org/10.3390/app12020870
25. Karabulut E., Pileggi S.F., Groth P., Degeler V. Ontologies in Digital Twins: A Systematic Literature Review. *Future Generation Computer Systems.* 2024;153:442-456. https://doi.org/10.1016/j.future.2023.12.013

Поступила 29.07.2025; одобрена после рецензирования 11.09.2025; принята к публикации 02.10.2025.

Submitted 29.07.2025; approved after reviewing 11.09.2025; accepted for publication 02.10.2025.

Об авторах:

Гапанович Дмитрий Антонович, ведущий программист лаборатории открытых информационных технологий кафедры информационной безопасности факультета вычислительной математики и кибернетики, ФГБОУ ВО «Московский государственный университет имени М.В. Ломоносова» (119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1), **ORCID: https://orcid.org/0000-0002-3222-694X,** dim.gapanovich@gmail.com

Дубков Владимир Михайлович, магистрант кафедры информационной безопасности факультета вычислительной математики и кибернетики, ФГБОУ ВО «Московский государственный университет имени М.В. Ломоносова» (119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1), **ORCID: https://orcid.org/0009-0001-6131-7840**

Князев Максим Алексеевич, студент кафедры информационной безопасности факультета вычислительной математики и кибернетики, ФГБОУ ВО «Московский государственный университет имени М.В. Ломоносова» (119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1), **ORCID: https://orcid.org/0009-0005-9475-2506**

Сухомлин Владимир Александрович, заведующий лабораторией открытых информационных технологий кафедры информационной безопасности факультета вычислительной математики и кибернетики, ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова» (119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1), доктор технических наук, профессор, **ORCID: https://orcid.org/0000-0001-9468-7138,** sukhomlin@mail.ru

Все авторы прочитали и одобрили окончательный вариант рукописи.

About the authors:

Dmitry A. Gapanovich, Lead Software Developer of the Open Information Technologies Lab, Department of Information Security, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1 Leninskie Gory, Moscow 119991, GSP-1, Russian Federation), **ORCID: https://orcid.org/0000-0002-3222-694X,** dim.gapanovich@gmail.com

Vladimir M. Dubkov, Master degree student of the Department of Information Security, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1 Leninskie Gory, Moscow 119991, GSP-1, Russian Federation), **ORCID: https://orcid.org/0009-0001-6131-7840**

Maxim A. Knyazev, Student of the Department of Information Security, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1 Leninskie Gory, Moscow 119991, GSP-1, Russian Federation), **ORCID: https://orcid.org/0009-0005-9475-2506**

Vladimir A. Sukhomlin, Head of the Open Information Technologies Lab, Department of Information Security, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1 Leninskie Gory, Moscow 119991, GSP-1, Russian Federation), Dr. Sci. (Tech.), Professor, **ORCID: https://orcid.org/0000-0001-9468-7138,** sukhomlin@mail.ru

All authors have read and approved the final manuscript.