# LOCAL AND GLOBAL TIME IN MODELING OF EVOLVING SYSTEMS

Igor N. Skopin

Institute of Computational Mathematics and Mathematical Geophysics SB RAS,
630090, Novosibirsk, Russia
Novosibirsk State University,
630090, Novosibirsk, Russia

DOI: 10.24411/2073-0667-2020-10013

The study of evolving systems includes the reflection in the models of changes in the system during its existence. The processes that ensure development begin, execute and end within a certain time frame, which in itself indicates the need for modeling of the time. Not only hours, minutes, etc. are important here, although in some simple cases this attachment of behavior to the time scale may be sufficient, as well as properties such as mutual influence and synchronization of element behavior, development under the influence of events, that are external in relation to the system and its elements produced. In other words, at the model level are significant the interactions of the elements, but not the moments of absolute time when these interactions occur.

One can talk about the local time of an individual element, defining it as a sequence of events in which the element operates during the system evolving. But this is not global time common to all elements. Such a time would become an additional entity of the model without any advantages for synchronizing interactions.

The problem of adequate time reflection in models is very important for any approach to the study of processes and phenomena. But when evolving systems are studied, it takes on special significance. The common task in such cases is the reconstruction or imitation of the behavior of the system as the interaction of its elements in time. As a result, the concept of model time is required, which would reflect reality in concert with interactions.

The article contains definitions of notions related to local for the system elements times, proof of the correctness of constructing models based on the consideration of global time as a partial order relationship on the set of events that occur when the system model is executed. An accurate definition of evolving systems is required to rigorous present the proposed approach to modeling. We give such a definition and prove its correctness, using the notion of the system element life cycle.

From the above it follows that our approach negatively refers to the use of global time as one of the basic notion of modeling. Nevertheless, it does not limit the model developer and allows him to operate with global time, which becomes a concept subordinate to event-drived control and local time of system elements.

The proposed formal definition of concepts related to time cannot be considered as the only possible correct representation of time in models. This approach is adequate to modeling in tasks for which the activity of elements is essential. As important alternative approaches, we consider modeling based on quasi-parallel systems implemented, for example, in the programming languages Simula and Simula 67, as well as a number of other special solutions.

One of the main goals of this paper is the construction and justification of proposed concept the main property of which is the priority of local time of elements before global time. The structure of the

presentation obeys this goal. The next section 2 gives information on various types of representations of time used in modeling, in particular, in the study of evolving systems. Section 3 clarifies the notions that characterize a system as evolving. It also describes the event-driven control of interactions of elements that uses as a method for determining the time in models of evolving systems. The following sections 4–7 are devoted to formalizing proposed in the article concept. Section 4 justifies the use of local time of a system's elements in the event-based organization of interactions in a model system. Section 5 introduces the concept of an evolving element and the associated with this notion of states and transitions from one state to another. Section 6 is devoted to proving the correctness of using partial ordering of events as the basis for determining the local time of an element. Section 7 completes the proof of the formal correctness of the proposed concept and shows how global time can be determined using event-driven control. The Conclusion presents information on other approaches to modeling time, conceptually close to the concept being developed. This part of the article also provides motivation for what tools are needed to support the proposed concept in modeling.

The research presented in this work showed that overcoming the difficult problems of globalization of time based on world clocks is achievable if global time is considered as an entity secondary to local times of elements of the system. On this basis, design patterns can be used that implement fairly general methods for constructing models of evolutionary development. The identification of situations in which such patterns are required, we consider as a promising work.

**Key words:** local and global time; partial order relation on a set of events; events, reaction of elements to events; event protocols.

## References

1. Avgustin A. Ispoved'. Seriya „Pamyatniki religiozno-filosofskoj mysli“. Per. s lat. M. K. Sergeenko. 1991. M.: Izdatel'stvo „Renessans“, SP IVO — SiD. ISBN 5-7664-0472-7.

2. Namestnikov A. M. Razrabotka imitacionnyh modelej v srede MATLAB // Meto-dicheskie ukazaniya dlya studentov special'nostej 01719, 351400. Ul'yanovsk, Ul-GTU, 2004.

3. SHevchenko A. A. Upravlenie vremenem pri proektirovanii imitacionnyh modelej // CHast' sb. Prikladnaya informatika. N 3. 2006. Laboratoriya Matematicheskogo i komp'yuternogo modelirovaniya. S. 113–119.

4. Skopin I. N. Lokal'noe i global'noe vremya pri modelirovanii razvivayushchihsya si-stem. V sb. trudov Sed'moj mezhdunarodnoj konferencii pamyati akademika A. P. Ershova „Perspektivy sistem informatiki“. Rabochij seminar „Naukoemkoe pro-grammnoe obespechenie“. Novosibirsk: OOO „Sibirskoe Nauchnoe Izdatel'stvo“, 2009. S. 255–259.

5. Sistema. Bol'shoj Rossijskij enciklopedicheskij slovar'. M.: BRE. 2003, S. 1437.

6. Smirnov G. A. Okkam, Uil'yam // Novaya filosofskaya enciklopediya / In-t filosofii RAN; Nac. obshchestv.-nauch. fond. 2-e izd., ispr. i dopol. M.: Mysl', 2010. ISBN 978-5-244-01115-9.

7. Skopin I. N. Ierarhicheskie otnosheniya . metodologicheskaya osnova izucheniya po-nyatiya ierarhij // Vestnik Rossijskogo universiteta druzhby narodov. Seriya „Informatizaciya obrazovaniya“. M.: RUDN, 2014. N 1. S. 56–63.

8. Skopin I. N. Subordinacionnye otnosheniya v metodike izucheniya ponyatiya ierar-hichnosti // Vestnik Rossijskogo universiteta druzhby narodov. Seriya „Informatizaciya obrazovaniya“. M.: RUDN, 2014. N 2. S. 35–49.

9. Dejkstra E. Vzaimodejstvie posledovatel'nyh processov. V sb. „YAzyki program-mirovaniya“, pod red. F. ZHenyui. Per. s angl. M.: „Mir“, 1972.

10. Lamport L. Time, clocks, and the ordering of events in a distributed systems // Commun. ACM. 1978. Vol. 21(7). P. 558–565.

11. Chandy K. M., Misra J. Distributed simulation: a case study in design and verification of distributed programs // IEEE Transactions on Software Engineering. 1978. Vol. SE–5(5). P. 440–452.

12. Ferscha A. Parallel and distributed simulation of discrete event systems // Parallel and Distributed Computing Handbook. McGraw-Hill. 1996. P. 1003–1041.

13. Kazakov YU. P., Smelyanskij R. L. Ob organizacii raspredelennogo imitacionnogo modelirovaniya // Programmirovanie. 1994. N 2. S. 45–63.

14. Fujimoto R. M. Parallel and Distributed Simulation Systems // Wiley Interscience, 2000.

15. Fujimoto R. M. Parallel and Distributed Simulation Systems // Proc. of the Winter Simulation Conf. 2001. P. 147–157.

16. Dal O. I., Nyugord K. Simula . yazyk dlya programmirovaniya i opisaniya sistem s diskretnymi sobytiyami // Algoritmy i algoritmicheskie yazyki. Vyp. 2. M.: VC AN SSSR, 1967.

17. Dal O. I., Myurhaug B., Nyugord K. Simula 67 universal'nyj yazyk programmirova-niya // Perevod s angl. K. S. Kuz'mina i E. I. YAkovleva. M.: Mir, 1969.

18. Nygaard K., Dahl O.-J. The Development of the SIMULA Languages // History of programming languages. ACM New York, NY, USA, 1981. P. 439–480.

19. Nepejvoda N. N. Skopin I. N. Osnovaniya programmirovaniya. Izd-vo: Moskva-Izhevsk: Institut komp'yuternyh issledovanij. 2003. ISBN: 5-93972-299-7.

20. Backus J. Can programming be liberated from the von Neumann style? A functional style and its algebra of programs // CACM, 21(8), August 1978. P. 613–641.

21. Malyshkin V. Assembling of Parallel Programs for Large Scale Numerical Modeling. . In the Handbook of Research on Scalable Computing Technologies. IGI Global, USA, 2010. Chapter 13, P. 295–311. ISBN 978-1-60566-661-7.

22. Keene S. Object-Oriented Programming in Common Lisp: A Programmer's Guide to CLOS, 1988, Addison-Wesley. ISBN 0-201-17589-4

23. Haskell 98 Language and Libraries. The Revised Report. Dec. [Electron. Res.]: `https://www.haskell.org/onlinereport/`.

24. Akhmed-Zaki D., Lebedev D., Malyshkin V., Perepelkin V. Automated Construction of High Performance Distributed Programs in LuNA System // PaCT-2019 proceedings, LNCS 11657, Springer, 2019, P. 3–9. DOI: 10.1007/978-3-030-25636-4_1.

25. Ahmed-Zaki D. ZH., Lebedev D. V., Malyshkin V. E., Perepelkin V. A. Avtomati-zaciya konstruirovaniya raspredelennyh programm chislennogo modelirovaniya v si-steme LuNA na primere model'noj zadachi // Problemy informatiki, N 4, 2019. S. 53–64. DOI: 10.24411/2073-0667-2019-00017.

26. Okol'nishnikov V. V. Predstavlenie vremeni v imitacionnom modelirovanii // Vychislitel'nye tekhnologii. 2005. T. 10, N 5. S. 57–80.IEEE Std P1516.

27. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) — Framework and Rules. N.Y.: Institute of Electrical and Electronics Engineers, Inc., 2000.

# ЛОКАЛЬНОЕ И ГЛОБАЛЬНОЕ ВРЕМЯ ПРИ МОДЕЛИРОВАНИИ РАЗВИВАЮЩИХСЯ СИСТЕМ

И. Н. Скопин

Институт вычислительной математики и математической геофизики СО РАН,
630090, Новосибирск, Россия
Новосибирский государственный университет,
630090, Новосибирск, Россия

УДК 51
DOI: 10.24411/2073-0667-2020-10013

Понятия глобального и локального времени обсуждаются с точки зрения их реализации в моделях развивающихся систем. Предлагается поход к изучению таких систем в рамках событийного механизма управления в имитационных моделях. Показана корректность использования времени в сочетании с этим механизмом.

**Ключевые слова:** локальное и глобальное время; отношение частичного порядка на множестве событий; события, реакция элементов на события; протоколы событий.

*Что я измеряю время, это я знаю, но я не могу измерить будущего, ибо его еще нет; не могу измерить настоящего, потому что в нем нет длительности, не могу измерить прошлого, потому что его уже нет. Что же я измеряю? Время, которое проходит, но еще не прошло?*
*— Аврелий Августин (354–430). Исповедь [1]*

**Введение.** Исследование развивающейся системы методами моделирования всегда предусматривает, что в модельном представлении развития отражаются изменения реальной системы, происходящие в некоторые моменты времени ее существования. Процессы, обеспечивающие развитие, начинаются, выполняются и завершаются в определенных временных рамках, что само по себе указывает на необходимость моделирования времени. Однако здесь важны не столько часы, минуты и т. п., сколько такие свойства, как взаимовлияние и синхронизация поведения элементов системы, развитие под воздействием событий, внешних по отношению к моделируемой системе или продуцируемых ее элементами. Иными словами, на модельном уровне более существенны взаимодействия элементов, чем моменты абсолютного времени, когда эти взаимодействия происходят.

Проблема адекватного отражения времени в моделях очень важна для любого подхода к изучению процессов и явлений. Но когда изучаются развивающиеся системы, она приобретает особое значение. Общей задачей в таких случаях является реконструкция или имитация поведения системы как взаимодействия ее элементов во времени. Как следствие, требуется концепция модельного времени, которая отражала бы реальность согласованно с взаимодействиями.

Одной из главных целей предлагаемой работы является построение и обоснование такой концепции. Этой цели подчиняется структура изложения. В следующем разделе 1 приводятся сведения о различных типах представлений времени, используемых при моделировании, в частности, при изучении развивающихся систем. Раздел 2 уточняет понятия, характеризующие систему как развивающуюся. В нем также описывается механизм событий, управляющий взаимодействием элементов, как метод определения времени в моделях развивающихся систем. Последующие разделы 3–6 посвящены формализации предлагаемой концепции оперирования информацией, относящейся к времени в моделях развивающихся систем. Раздел 3 обосновывает использование локального времени элемента системы при событийной организации взаимодействий в модельной системе. В четвертом разделе вводится понятие развития элемента и связанных с ним переходов из одного состояния в другое. Раздел 5 посвящен доказательству корректности использования частичной упорядоченности событий в качестве основы определения локального времени элемента. Шестой раздел завершает доказательство формальной корректности предлагаемой концепции и показывает, какое глобальное время можно определять при событийном управлении. В Заключении представлены сведения о других подходах к моделированию времени, концептуально близких к развиваемой концепции. В нем также приводится мотивация того, какой инструментарий требуется для поддержки предлагаемой концепции при моделировании.

**1. Варианты представления времени при моделировании.** В любом подходе к моделированию необходимо различать три варианта представления времени, так или иначе используемых при построении моделей и проведении расчетов[1]:

— *Реальное время*, в котором происходит функционирование изучаемого объекта, — это время реальной системы, модель которой строится для проведения исследования;

— *Модельное время*, в масштабе которого организуются экспериментальные расчеты модели, — это отражение реального времени в конструируемой модели (его часто называют системным временем, что, на наш взгляд, может быть понято не совсем корректно);

— *Машинное время*, отражающее затраты времени вычислителя на проведение имитации, — этот атрибут особенно важен при использовании моделей, адекватность которых существенно зависит от объема вычислительных ресурсов, задействованных в расчетах.

Реальное время для исследования является атрибутом или наблюдаемой характеристикой моделируемого объекта в целом, которая фиксирует рамки развития системы, представленной в виде поведения ее элементов. В большинстве случаев реальное время используется как источник значений временны**х** параметров модели в целом и ее элементов, отражая причинно-следственные связи. При необходимости моделировать реальное или, что то же, *абсолютное время* реальной системы естественно представлять его структурой данных, формируемой по информации о порядке взаимодействий элементов, т. е. как структуру, глобально доступную всем элементам системы при выполнении их локальных действий.

Выбор подхода к заданию модельного времени очень неоднозначен (см., например, [3]), но если развитие системы связывать с активностью и взаимодействиями элементов, то становится очевидной необходимость согласовано определять два вида модельного времени: *локальное время* для каждого *элемента* и *глобальное время системы* в целом. Ключевое требование здесь — согласованность.

[1]Мы даем определения вариантов представления времени из [2], уточнив их для большего соответствия предлагаемой концепции.

Единое для всеобщего употребления время человеческие сообщества придумали еще на заре своего существования, и поэтому использование его для синхронизации событий, происходящих при функционировании модельной системы и ее элементов, кажется естественным. К сожалению, глобальное время при каком бы то ни было его определении страдает избыточностью, обусловленной тем, что приходится отслеживать не только события, но и ход времени, т. е. реально несуществующего генератора времени. Но хуже другое. Этот механизм не справляется с задачей согласования действий элементов системы. Во-первых, теряется свойство актуальности событий — всегда приходится сверяться с отчужденными от локальной последовательности действий элемента „мировыми часами“, а во-вторых, не достигается требуемая точность указания одновременности и последовательности выполнения нескольких действий, зависящих друг от друга, т. е., когда одни из них производят данные для других. В таких случаях справиться с зависимостями не помогают ни соглашение о „мгновенности“ действий, ни рассмотрение недетерминированного поведения — в реальной жизни именно этого достаточно. Поэтому приходится дополнять модель времени специальными корректирующими согласованиями. Если так, то не проще ли отказаться от исчисления разных времен, которое мало что дает для согласования порядка модельных действий и событий? Более подробное обсуждение проблем глобального времени можно найти в работе [4].

При построении моделей развития машинное время, затрачиваемое на проведение модельных расчетов, лимитирует количество вариантов траекторий развития, которые удается проверить. Кроме того, когда в модели используется имитация непрерывных процессов, описываемых дифференциальными и интегральными уравнениями, вместо точного решения которых вычисляется его аппроксимация, качество результата можно повышать за счет использования более производительного вычислительного оборудования. Указанные обстоятельства влияют на адекватность проводимого исследования.

**2. Моделирование развивающихся систем и событийное управление.** Система (др.-греч. *σύστημα* „целое, составленное из частей; соединение“), согласно определению Большого российского энциклопедического словаря, „это множество элементов, находящихся в отношениях и связях друг с другом, которое образует определенную целостность, единство“ [5]. В приведенном толковании понятия системы отражена общность разнообразных существующих его определений: можно сказать, что практически всегда системы составляются из элементов и связей, суть которых фиксируется в дальнейших уточнениях.

Применительно к задачам моделирования следует определять одновременно два вида систем:

— реально существующая изучаемая система — *реальный объект* исследования и

— *модельная система* — артефакт, отражающий свойства и поведение реальной системы, определенные для исследования.

В рамках предлагаемой концепции считается, что реальная система является *развивающимся объектом*. Развитие — это изменение системы, которое осуществляется под воздействием активности ее элементов и влияний окружения. Изменяются как элементный состав системы, так ее свойства и свойства ее элементов, в том числе связи между элементами. В этой связи в качестве целей исследования системы может быть выбрано построение прогноза или выявление закономерностей изменений системы, т. е. зависимость ее развития от свойств элементов, их взаимодействия между собой и с окружением, иначе — со средой функционирования.

*Модельная система* — это объект, конструируемый для имитации развития реального объекта, которая проще для выявления закономерностей за счет *абстрагирования* от элементов и связей, рассматриваемых как несущественные. По сравнению с реальным объектом информация, получаемая о модельной системе с помощью наблюдений, измерений, экспериментов, оказывается более доступной для аналитической работы. Если абстрагирование от несущественного проведено адекватно требованиям к исследованию, а наблюдения выполняются корректно, то перенос на реальный объект выявляемых закономерностей развития модельной системы будет правомерен.

Элемент модельной системы выделяется в ней своим *атрибутивным представлением*, допускающим передачу данных и методов для выполнения действий, необходимых при оперировании с моделью. В атрибутивном представлении сохраняется информация об отнесении элемента к определенному типу, а также о связях между элементами — отношениях с другими элементами. Типы связей элементов реального объекта очень разнообразны и переменчивы, но еще до начала исследования отбираются те элементы и связи, которые принимаются за существенные — существование остальных аспектов реальных объектов игнорируется. В результате строится, а точнее, выбирается требуемая для изучения реального объекта элементная *декомпозиция*, которую в дальнейшем будем называть изучаемой *реальной системой*.

В модельном объекте элементы, их свойства и связи не выбираются, а конструируются с использованием понятий и объектов уже существующей системы понятий и инструментов их сочетания. Таким образом строится модельная система. Критерием отбора составляющих для модельной системы элементов является то, что, используя их свойства и связи, реальная система, представляющая изучаемый объект, может быть реконструирована как самостоятельная структура, допускающая взаимно-однозначное соответствие между изучаемой реальной и модельной системами.

Когда объектом исследования является развивающаяся система, то постулируется, что элементы системы могут выполнять определенные *действия*: создание и уничтожение элементов, изменение своих свойств и связей, а также свойств и связей других элементов. Существенно то, что при развитии системы доступные для выполнения действия меняются. Поэтому нужно говорить о последовательности *состояний* элемента. Каждое из них характеризуется набором допустимых действий, которые элемент, находясь в этом состоянии, может выполнять. Состояние — это особого рода свойство элемента, динамически меняющееся в ходе функционирования системы.

Понятие состояние элемента модельной системы хорошо согласуется с таким же понятием реальной развивающейся системы. Более того, само развитие системы можно трактовать как объединение последовательностей смен состояний всех ее элементов. В реальных системах такое объединение достигается за счет привязки смен состояний к единой временно̀й оси, что, как было показано выше, недостаточно для синхронизации взаимодействий в модельных системах. Здесь требуется отражать в модели причины перехода элемента из одного состояния в другое: в ходе проведения модельных расчетов элемент должен получить сигнал о необходимости выполнения этого перехода. Адекватным способом такой сигнализации является достаточно общий *механизм событий*, применимый не только для изменения состояний элементов. Суть его сводится к следующим операциям, выполняемым в указанном порядке:

1) *Генерация события* некоторым элементом модельной системы или ее окружением (применительно к смене состояний элементов это источник события, причина возможного изменения состояний некоторых элементов);

2) *Распознавание события* теми элементами, которые должны получить сигнал (это идентификация сигнала некоторыми элементами как причины возможного изменения их состояний);

3) *Реагирование на событие* каждым из элементов, распознавших событие, выполнением определенных для этого события действий (это следствие распознанного элементом события как причины принятия решения об изменении его состояния).

Важно заметить, что реакция на событие, трактуемая как требование выполнить некоторое действие, может оказаться отличной от этого действия. Решение о том, какая реакция будет выполнена, принимает элемент на основе всей доступной для него информации, а не только исходя из распознавания события. Так, если требование перейти в новое состояние обусловливается не одним событием, а некоторой их последовательностью (примером может служить ситуация с накоплением какого-либо ресурса), то оно принимается только в результате реагирования на заключительное событие последовательности. Отметим, что события с неоднозначно выбираемыми реакциями одного и того же элемента легко ликвидировать: достаточно для такого состояния в качестве реакции задать специальную функцию, генерирующую для каждого из вариантов первоначальных реакций вспомогательное событие с единственной реакцией.

Реагирование на событие необязательно является немедленным следствием распознавания. Реакция может быть *отложена* до выполнения условий срабатывания, а в реальной системе или при реализации глобального времени модели она, кроме того, может быть назначена на определенное время. Как и в случае неоднозначных реакций, отложенное реагирование легко имитируется с помощью специального вспомогательного события, генерирующегося при выполнении условий срабатывания. При этом необязательно генерировать это событие с явным назначением отложенной реакции на определенное время. Необходимо лишь, чтобы оно произошло *после* отложенной реакции и *до* зависящих от нее событий.

События связывают элемент системы с другими элементами: само порождение элемента и его дальнейшие состояния зависят от его действий, действий других элементов и окружения, точнее — от результатов этих действий. Все эти действия есть следствия соответствующих событий, являющихся их причинами. Некоторые из реальных событий при моделировании игнорируются — исследователь либо не знает о них, либо абстрагируется от их существования. В обоих случаях причинность действия элемента и, в том числе, изменения его состояния *приписывается* другим событиям или же процессам, инициированным такими событиями. Как причинность, так и приписывание причинности имеют прямое отношение к понятию модельного времени: пока причинное событие (возможно, несколько необходимых событий или одно из них — это зависит от семантики модели) не произойдет, не могут произойти соответствующие реакции. Понятно, что приписывание не должно нарушать причинно-следственные отношения, и, как следствие, запрещается приписывание причинности *позднее* наступления его следствиям и, в том числе, событиям-следствиям.

Проиллюстрируем введенные понятия на примере фрагмента условной модели чаепития. В качестве одного из элементов модели рассматривается электрический чайник со следующими альтернативными свойствами: пустой и заполненный водой, выключенный

и включенный. Допустимые комбинации этих свойств характеризуют состояния чайника. При корректной эксплуатации чайника событие включения приводит к активизации процесса кипячения. Будем считать, что событие включения чайника без воды контролируется, т. е. для пустого чайника включения не влекут никаких действий, в частности, не меняют его состояние. В случае заполненного водой чайника событию включения приписывается причинность: в модели чаепития полагается, что вода в чайнике закипает из-за него, а закипание до включения не допускается. При этом игнорируется, что закипание невозможно без нагревания, процессов генерации электричества и превращения его в тепло, а событие закипания происходит через некоторый промежуток времени после включения без каких бы то ни было специальных действий. В реальной жизни вопрос, когда это случится, не возникает, но на модельный уровень факт закипания переносится как утверждение о том, что это происходит *после* включения. Чтобы событие закипания произошло, может потребоваться новое включение, например, когда возникло принудительное прекращение нагрева в промежутке между первоначальным включением и окончанием процесса. Но информация о величине этого временно́го интервала фактически никак не используется.

Осознанное приписывание причинности — естественный прием моделирования, который позволяет корректно абстрагироваться от несущественных деталей изучаемой системы. В то же время, неосознанное приписывание может приводить к нарушению принципа Бритвы Оккама: „Не следует множить сущее без необходимости“ [6] и, как следствие, к неадекватности создаваемых моделей. Это особенно важно учитывать в подходах, которые появляются, когда исследуемые процессы и явления требуют совместного построения набора моделей:

— конкретная система рассматривается как часть, элемент другой системы более высокого уровня, называемой *надсистемой*;

— некоторые элементы системы рассматриваются как состоящие из других элементов, т. е. как системы более низкого уровня, называемые *подсистемами*.

Варианты построения модели, состоящей из нескольких уровней, начиная от элементов низкого уровня путем собирания из них нужных подсистем надсистемы (*восходящий*, т. е. *снизу-вверх подход)*, или в обратном направлении (*нисходящий*, т. е. *сверху-вниз подход)*, формально эквивалентны. Оба подхода могут использоваться в практике моделирования. И если выбирается один из них, то второй стоит рассматривать в качестве проверочного инструмента. Однако следует отметить, что как трудоемкость построения системы моделей, так и эффективность проведения расчетов во многом зависят от выбора направления конструирования.

Оба подхода позволяют строить как *строго иерархические системы*, когда каждая из подсистем может быть частью, т. е. элементом только одной надсистемы, так и *нестрого иерархические системы*, когда подсистемы-элементы надсистем могут входить в разные надсистемы[2].

Наблюдаемую реальность, в которую погружен объект изучения, можно рассматривать как систему: она состоит по крайней мере из двух частей: изучаемый объект и его окружение. С точки зрения исследователя уже это разделение является абстракцией: игнорируется то, что не требуется для целей исследования, и фиксируются части-элементы

[2]Свойства иерархичности и способы построения иерархий на базе двух видов бинарных отношений: эквивалентности и субординации подробно изучаются в работах [7, 8]. В них строго и нестрого иерархические системы названы, соответственно, классификационными и субординационными.

создаваемой системы и обычно их наименования. Вот примеры таких систем, уровень которых для соответствующих исследований мог бы оказаться самым высоким: объект земля — это суша, вода и воздух, все остальное — окружение; объект дерево — это крона, ствол и корни, все остальное — окружение; объект научная школа — это учителя-основатели, поколения учеников, периоды роста, апогея, стабилизации и затухания и др., все остальное — окружение.

Именно на уровне наблюдаемой реальности появляется время как особый атрибут системы в целом. Временной порядок, в котором возникают и отрабатываются события наблюдаемой реальности, необязательно имеет прямое отношение к порядку модельных событий подсистем и их элементов, и это может вводить в заблуждение. Рассмотрение реальности как системы, состоящей из элементов, не означает отождествление ее с модельной системой. В частности, события и их временной порядок в реальности определяются огромным множеством факторов, влияющих на развитие объекта исследований, а не только теми из них, которые остались после отсева игнорируемых или незамеченных, и, конечно, в реальности просто нет места приписываниям и искусственно конструируемым причинно-следственными связям. Реальность, в которую погружен объект изучения, является цельной, единой и неделимой на части.

Как же тогда добиться согласованности модельной концепции времени с реально существующей или только субъективно ощущаемой, субстанциональной или лишь реляционной сущностью времени[3], интерес к которой зародился на заре человечества и не прекращается до сих пор? Не претендуя на универсальность, мы предлагаем решение проблемы времени при моделировании развивающихся систем, поведение которых существенно зависит от активности их элементов. Основой решения является организация взаимодействия элементов, опосредованная событийным управлением вычислениями.

Последовательность событий, которые распознает элемент модели и на которые он реагирует в ходе развития системы, далее называется *протоколом поведения элемента*. Именно протокол естественно отождествлять с *локальным временем* элемента. Протоколы могут пересекаться, т. е. иметь общие события, что означает синхронизованное поведение элементов системы — они вовлечены в действия, которые следует считать общей реакцией элементов на распознанное ими событие.

Совокупность всех протоколов всех элементов модельной системы некоторого уровня частично упорядочивает множество всех событий этой системы. И именно этот частичный порядок имеет смыл считать представлением наблюдаемого времени реальной системы, т. е. *глобальным временем модельной системы*. При совместном построении моделей разных уровней это представление является атрибутом системы самого верхнего уровня. Переход от системы некоторого уровня вниз по иерархии модели дает возможность согласования локальных времен подсистем с временем их общей надсистемы.

Отметим, что протоколы не имитируют в модели реальное наблюдаемое время (они не претендуют, например, на отражение длительности интервалов между событиями), но выполняют такую важную для развития функцию, как задание бинарных отношений „раньше“, „позже“ и „несравнимы“ на множестве всех событий.

Требование имитации длительности может оказаться полезным в некоторых видах моделей, например, для апостериорного, исторического анализа, но не в случаях моделирования развития, для которого прямое соответствие между реальным наблюдаемым време-

[3]В Приложении к данной работе приведена схема взаимосвязей различных концепций времени, представленная в 2000-летней истории развития естествознания.

нем объекта моделирования и модельным его представлением означало дополнительную сущность без каких бы то ни было преимуществ — еще одно нарушение принципа Бритвы Оккама.

**3. События, протоколы, локальное время.** В данном разделе формализуются понятия, относящиеся к событийной концепции времени. Формализация фиксирует тот минимально необходимый уровень требований, достаточный для точного задания предлагаемой концепции времени и взаимодействий.

Определение 1. Под термином *событие* понимается сущность поведения системы, которая обладает следующими свойствами. События

a) *вырабатываются* окружением или элементами системы;

b) могут *отрабатываться* некоторыми элементами, про которые говорят, что они *реагируют* на событие, т. е. *идентифицируют* или *распознают* событие и активизируют обработку, т. е. *программу реакции* на событие;

c) могут *отслеживаться*, т. е. фиксироваться для каждого элемента в последовательности, в которой он реагирует на события системы;

d) *не повторяются*, т. е. в последовательности событий, на которые реагирует каждый из элементов, не существует двух или более одинаковых событий.

Тогда для каждого элемента системы *локальное время* определяется как последовательность всех событий, на которые этот элемент реагирует. Эта последовательность называется *протоколом поведения элемента.*

Комментарий к определению 1.

(a). Появление событий, на которые реагируют элементы, — неопределяемое понятие. Достаточно знать лишь то, что элементы системы взаимодействуют посредством реакций на события, которые возникают в результате внешних воздействий или поведения каких-либо элементов. Не исключается возможность элемента объявить событие „для себя“, т. е. такое, на которое он реагирует сам, возможно, совместно с другими элементами.

Множество событий, на которые реагирует элемент, меняется в процессе функционирования системы. В результате различных реакций динамически меняется набор распознаваемых событий, а значит, меняется реакция элемента на события (неважно, внешние для системы или внутренние). Это допущение позволяет говорить о развитии элементов и системы в целом.

(b). Отработка события элементом, — это идентификация события и выполнение программы реакции, которая использует информацию, связанную с событием. Поведение элемента, который не реагирует на событие, не зависит от этого события. Он про него ничего не знает. Можно сказать, что такие события для элемента не существуют. В то же время, неизвестные элементу события могут косвенно влиять на его поведение, поскольку другие события в ходе своих реакций могут изменять атрибутивные описания элементов и общий контекст выполнения действий элементов.

(b.I). Постулируется *мгновенность идентификации события* элементом:

— событие не влияет ни на какие на действия системы и ее элементов, кроме как на те из них, которые обнаруживают событие;

— в ходе идентификации другие события не возникают.

Идентификация — это спусковая функция (предикат), которая предписывает выполнение или невыполнение программы реакции.

(b.II). Выполнение программы реакции на идентифицированное событие может быть *мгновенным*, если в ходе выполнения невозможно возникновение других событий. Если

это неверно, то выполнение реакции называется *длительным*. Последнее означает допустимость изменения поведения, в том числе и самой реакции из-за появления и отработки других событий, реакция на которые меняет атрибутивные описания элементов. Если есть тому основание, то задавать подобные изменения можно в качестве событий, предписывающих элементу осуществить их как реакцию на это событие. Тогда у элемента появляется возможность планировать изменения: выполнять их немедленно или после того, как его текущее действие завершится.

(c). Отслеживание элементом событий, на которые он реагирует, приводит к возможности доступа к истории элемента, т. е. к его протоколу, в котором отмечаются в порядке появления все события, связанные с элементом. Поэтому протоколы рассматриваются как *ретроспективное локальное время элемента*. Можно рассматривать и *перспективное*, или *прогнозное локальное время элемента*, понимая под этим совокупность всех возможных продолжений текущего состояния протокола, тем самым пополнив ретроспективное время будущими вариантами продолжений протоколов.

Протоколы могут пересекаться. Это означает, что два или более элемента реагируют на одно и то же событие совместно, т. е. никак не определяются ни порядок выполнения программ обработки, ни побочные эффекты, возможные в результате такой совместности. При моделировании необходимо уточнение поведения в подобных случаях с тем, чтобы при интерпретации не возникали коллизии, связанные с параллелизмом (см., например, [9]).

(d). Невозможность повторения событий означает, что, когда требуется реакция элементов на событие, совпадающая с реакцией на какое-либо событие, уже представленное в протоколах, это всегда новый экземпляр события, пусть даже функционально идентичный прежнему. При необходимости можно определять *типы событий* как классы эквивалентности по отношению идентичности реакции элементов системы.

Из прагматических соображений будем считать, что *поведение* каждого *элемента конечно*, т. е. он не может реагировать на потенциально бесконечное число событий. Или по-другому: все протоколы — конечные последовательности. Вместе с тем, множество потенциально возможных событий может оказаться бесконечным из-за появления в системе новых элементов, а значит, и новых будущих продолжений протоколов.

**4. Цикл функционирования элемента.**

Определение 2. Создание, активизация и деактивизация элемента.

Пусть на какое-то событие реагируют некоторые элементы системы таким образом, что в результате оказывается выполненным:

— *появление* атрибутного представления элемента, отвечающего его типу и отражающего включение элемента во все представленные в системе структуры;

— порождение *пустого* протокола нового элемента;

— объявление в качестве единственного события, идентифицируемого элементом, *события активизации*, планируемая реакция на которое начинает активное поведение элемента (фактическое *рождение*).

Такое событие называется *созданием элемента*. Создание — общее событие нового элемента и всех элементов, реагирующих на него.

Декларируется, что для каждого элемента системы определено событие, реакция на которое включает в себя прекращение его активности, т. е. он перестает реагировать на другие события. Такое событие называется *деактивизацией* элемента.

События создания и активизации являются первыми членами протокола нового элемента. Событие деактивизации — последний член протокола. Эти три события могут быть как внешними, так и внутренними для системы. Считается, что элементы системы, имеющиеся в начале ее функционирования, созданы фиктивным элементом, неактивным в дальнейшем.

Активизация может возникнуть либо сразу же после создания, либо оказаться отложенной, но до тех пор, пока она не произойдет, новый элемент не в состоянии реагировать на другие события. Это обеспечивается инициализацией атрибутивного описания элемента, которая рассматривается в качестве реакции на событие активизации.

Задачи изучения развивающейся системы можно подразделить на *исторические* и *прогнозные*. Первые связаны с анализом реализованных протоколов, вторые — с возможными вариантами продолжений протоколов. Если изучение касается только реализованных протоколов, то можно указать два ограничения для рассмотрения истории: начало функционирования системы и ее текущий момент. Технически удобно считать их формальными событиями, генерируемыми внешним образом. Будем обозначать их, как $\vdash$ и $\dashv$ соответственно.

На *начальное событие* $\vdash$ реагируют все активные элементы, которые представлены в системе первоначально. Их реакцией на него является инициализация соответствующих атрибутивных описаний. Т. е. для первоначальных элементов начальное событие является активизацией.

Событие $\dashv$ является *завершающим* либо *приостанавливающим* в зависимости от того, предполагается или нет возобновление функционирования системы. На него реагируют активные элементы, запоминая свое текущее состояние и замораживая все действия. Таким образом, завершающее/приостанавливающее событие понимается как событие деактивизации.

Возобновление целесообразно обеспечивать для выполнения анализа сложившейся конфигурации системы. Принудительное изменение конфигурации допустимо, но только в тех случаях, когда измененная конфигурация достижима естественным развитием системы из ее начальной конфигурации. В противном случае картина развития будет заведомо нереалистична.

В связи с возобновлением уместно расширить трактовку начального события, допуская его не только как „сотворение“ системы, но и в качестве средства возобновления функционирования в приостановленной конфигурации. В новой трактовке на $\vdash$ реагируют элементы, представленные в приостановленной конфигурации. Это не отменяет понимания начального события как активизации, но требует коррекции содержания реакции на него, которая обеспечивает восстановление атрибутивных описаний. Корректность возобновления гарантируется, когда *все* элементы, активные при приостановке, реагируют на событие $\vdash$. В противном случае дальнейшее поведение системы может оказаться недостижимым из начальной конфигурации.

Событие $\vdash$ начинает протокол любого элемента, представленного в системе первоначально. Приостановка приводит к дополнению всех протоколов событием $\dashv$, а возобновление — к дополнению их событием $\vdash$. Чтобы не противоречить требованию неповторяемости событий (см. условие (d) в определении 1), считается, что два последних члена протокольной последовательности $\dashv$ и $\vdash$ взаимно уничтожают друг друга в ходе возобновляющей инициализации элемента. Тем самым достигается „незаметность“ приостановки.

Определение 3. Исторические и перспективные события.

Множество всех событий, происходивших в течение функционирования системы, т. е. зафиксированных в протоколах всех ее элементов, обозначается как $\Omega$. Это *исторические* или *реализованные события*. $\Omega$ формируется путем пополнения возникающими событиями первоначально пустого множества.

Множество всех событий, которые могут произойти в течение какого бы то ни было функционирования системы, называется *множеством всех потенциально возможных событий*. Оно обозначается как $\Omega^*$.

Множество $\Omega$ состоит из событий, на которые реагирует хотя бы один элемент. Оно не включает потенциально возможные события, на которые была бы реакция при другом стечении обстоятельств, а также прогнозные события из вариантов продолжений протоколов. $\Omega$ всегда конечно. Именно с ним связано решение исторических задач изучения системы. Что касается задач прогнозных, то они относятся к изучению множества потенциально возможных событий $\Omega^*$. Но требуется не все это множество, а только та его часть, которая содержит в себе продолжения протоколов, содержащих лишь реализованные события. Для коротких прогнозов можно воспользоваться вариантами продолжений развития приостановленной конфигурации системы, но для изучения тенденций это слишком тяжеловесный инструмент.

Мы отождествляем протоколы и ретроспективное локальное время, но не отказываемся от синхронизированных взаимодействий. Следовательно, необходимо понятие, которое отражает то, что на бытовом уровне рассматривается как шкала времени, и которая дает необходимые и достаточные средства для описания взаимодействий.

**5. Временной порядок событий и глобальное время.**

Определение 4. Отношение временного частичного порядка на множестве событий и глобальное время.

Пусть $s_1, s_2 \in \Omega$ — два события, а отношение $\prec$ на множестве всех событий задается следующим образом:

a) Если $s_1$ и $s_2$ принадлежат одному протоколу, то $s_1 \prec s_2$ ($s_1$ „*раньше*“ $s_2$, обратное отношение — $s_2$ „*позже*“ $s_1$) в соответствии с их порядком в протокольной последовательности;

b) Если $s_1 s_2$ принадлежат разным протоколам, имеющим общие события, то $s_1 \prec s_2$ тогда и только тогда, когда существует общее для двух протоколов событие $s$ и $s_1 \prec s$, а $s \prec s_2$;

c) Отношение *транзитивно* $\prec$, т. е.

$$\forall s_1, s_2, s_3 (s_1 \prec s_2) \,\&\, (s_2 \prec s_3) \supset (s_1 \prec s_3)$$

(это условие необходимо постулировать только для таких троек $s_1$, $s_2$ и $s_3$, у которых первое и последнее события принадлежат разным протоколам, а $s_2$ — общее событие);

d) Для некоторых событий $s_1$ и $s_2$ отношение $\prec$ может задаваться *принудительно*, если такое задание не нарушает асимметричность отношения:

$$\forall s', s'' \in \Omega (s' \prec s'') \supset \neg (s' \prec s'')$$

(достаточно требовать этого только для тех $s'$ и $s''$, одно или оба из которых совпадают с $s_1$ или с $s_2$);

e) Во всех других случаях отношение $\prec$ между $s_1$ и $s_2$ не устанавливается.

Тогда $\prec$ называется *отношением временно́го частичного) порядка.* Это отношение объявляется *глобальным временем системы.*

Утверждение 1

Отношение $\prec$ является строгим частичным порядком на множестве всех реализованных событий $\Omega$. Цепями такого порядка являются все протоколы и все *ветвления протоколов*, т. е. последовательности событий, которые строятся следующим образом. Начало цепи берется из одного протокола или из ранее построенной цепи, а продолжение — из другого, имеющего общее событие с началом.

Заметим, что для множества всех потенциальных событий $\Omega^*$ подобное утверждение не имеет места, что хорошо согласуется с интерпретацией: при различных стечениях обстоятельств одни и те же события могут реализовываться в различном порядке.

— **Периоды возникновения событий и синхронизация протоколов.** Глобальное время можно чисто механически использовать для построения иерархий множества всех событий. Однако это построение довольно искусственно. В частности, не имеет под собой интерпретационной основы разнесение исторических событий по уровням, осмысленным для системных иерархий. События сами по себе не образуют систему, они могут являться лишь проявлением развития системы элементов. Иерархия событий представляет не структуру системы как набор ее элементов со связями, а лишь дает некоторую информацию о поведении системы в целом и ее элементов. Ее можно (и нужно!) использовать при рассмотрении системы как черного ящика, если предполагается, что связь между событиями и поведением элементов априори не известна (и стоит задача раскрытия этой связи), или как серого ящика, когда известны некоторые сведения о структуре, например, сами элементы. Иерархия событий в состоянии определять пределы, вне которых (раньше и позже) событие произойти не может — *периоды возникновения событий.*

Определение 5. Период возникновения событий.

Пусть $\prec$ — отношение временно́го порядка на множестве реализованных событий $\Omega$, $x$ — произвольное событие из $\Omega$. Тогда множества $L(x)$ и $H(x)$ определяются следующим образом:

$$L(x) = \{a \in \Omega | (a \prec x)\ \&\ \neg\exists b \in \Omega((a \prec b)\ \&\ (b \prec x))\};$$

$$H(x) = \{b \in \Omega | (x \prec b)\ \&\ \neg\exists a \in \Omega((x \prec a)\ \&\ (a \prec b))\}$$

и называются *нижним и верхним пределами периода* возможного *возникновения события* $x$ соответственно.

Утверждение 2.

Определение 5 корректно, т. е. для любого реализованного $x$ множества $L(x)$ и $H(x)$ не пересекаются и задают по крайней мере по одному событию, раньше и позже которых х произойти не может (наличие элементов в $L(x)$ и $H(x)$ следует из того, что $\vdash\prec x$ и $x \prec\dashv$; если бы эти множества пересекались, то для их общего элемента a было бы одновременно верно $x \prec a$ и $a \prec x$).

Определение 6. Синхронизация событий протоколов.

Пусть на множестве реализованных событий $\Omega$ определено отношением временно́го порядка $\prec$, а $E_1$ и $E_2$ — элементы системы, а $m(E_1)$ и $m(E_2)$ — протоколы этих элементов: $m(E_1) = \{s_1, \ldots, s_{n_1}\}$, $m(E_2) = \{s_1^{'}, \ldots, s_{n_2}^{'}\}$ соответственно.

Пусть, далее, $x$ — событие, на которое реагирует элемент $E_1$, т. е. $m(E_1) = \{s_1, \ldots, s_{i-1}, x, s_{i+1}, \ldots, s_{n_1}\}, 1 \leqslant i \leqslant n_1$.

Тогда считается, что

a) $x$ *синхронизован с* $m(E_2)$ *слева*, если в $m(E_2)$ имеется подпоследовательность $s'_1, \ldots, s'_{j_l}$ такая, что $s'_k \prec x$ при $k \leqslant j_l$ и $\neg(s'_k \prec x)$ при $k > j_l$. В этом случае *событие* $x$ *не может произойти ранее события* $s'_{j_l}$. Если при этом $s'_{j_l} \in L(x)$, то $x$ *синхронизован с* $m(E_2)$ *слева точно*;

b) $x$ *синхронизован с* $m(E_2)$ *справа*, если в $m(E_2)$ имеется подпоследовательность $s'_{j_r}, \ldots, s'_{n_2}$ такая, что $x \prec s'_k$ при $k \geqslant j_r$ и $\neg(x \prec s'_k)$ при $k < j_r$. В этом случае *событие* $x$ *не может произойти позднее события* $s'_{j_r}$. Если при этом $s'_{j_r} \in H(x)$, то $x$ *синхронизован с* $m(E_2)$ *справа точно.*

Считается, что $x$ *синхронизован с* $m(E_2)$ *полностью*, если $x$ синхронизован с $m(E_2)$ слева и справа, т. е. $\exists s'_{j_l}, s'_{j_r} \in m(E_2)$, удовлетворяющие условиям (a) и (b). Если при этом $x$ синхронизован с $m(E_2)$слева и справа точно, т. е. $s'_{j_l} \in L(x)$ и $s'_{j_r} \in H(x)$, то $x$ *полностью синхронизован с* $m(E_2)$ *точно.*

Если $x \in m(E_1)$ и $x \in m(E_2)$, то $x$ — *общее событие протоколов* $m(E_1)$ и $m(E_2)$, т. е. для некоторого $k$, $k < n_2$,

$$m(E_2) = \{s'_1, \ldots, s'_{k-1}, x, s'_{k+1}, \ldots, s'_{n_1}\}.$$

Тогда, по определению, $s'_{j_l} = s'_{k-1}$ и $s'_{j_r} = s'_{k+1}$. В этом случае происходит *совместная реакция элементов* $E_1$ *и* $E_2$ *на общее событие* $x$.

Утверждение 3.

Определение 6 корректно, т. е. для любого $x$, на который реагирует хотя бы один элемент, и любого протокола $m$ некоторого элемента либо $x \in m$, либо существуют события этого протокола $s'_{j_l}$ и $s'_{j_r}$, которые указывают на синхронизацию $x$ с $m$ слева и справа. Существуют протоколы, для которых эти свойства выполняются точно.

Для доказательства достаточно заметить, что любое событие происходит не ранее $\vdash$ и не позднее $\dashv$. Эти события — первые кандидаты на $s'_{j_l}$ и $s'_{j_r}$. Последовательный просмотр выбранного протокола отбраковывает события, не удовлетворяющие требуемым свойствам. Если все члены протокола за исключением начального и завершающего события оказываются отбракованными, то искомыми остаются события $\vdash$ и $\dashv$. Для элементов, которые возникают в ходе активного функционирования системы, процедура отбраковки остается осуществимой, поскольку всегда есть возможность выбора родительского элемента, ветвление которого означает создание нового элемента. Пропуск при отбраковке всех членов родительской протокольной последовательности до события создания элемента правомерен, т. к. в этой части цепи заведомо нет требуемых событий. С учетом того, что любой протокол является цепью, т. е. его члены упорядочены линейно, нужные события определяются однозначно.

Резюмируя сказанное выше, можно утверждать, что корректное глобальное время — это частичный порядок на множестве событий. Если при моделировании этого недостаточно, то желаемую шкалу времени можно задать с помощью специального элемента системы, который естественно трактовать как *генератор времени* или *системные часы.* Его роль — постоянное порождение моментов времени. На некоторые из них реагируют другие элементы, что означает привязку их активности к временной шкале: элемент „знает“, в какой момент он должен выполнить соответствующую реакцию. Некоторые события принудительно ставятся в отношении временного порядка с моментами времени. Это корректно, если сохраняется условие асимметричности (d) из определения 4 и выполнено транзи-

тивное замыкание для расширения частичного порядка. Таким построением дозировано достигается все, что требуется от традиционного глобального времени, и сохраняется неопределенность там, где привязка к времени является лишь гипотетической.

**Заключение.** Понятия глобального и локального времени, временно́го частичного порядка, синхронизации и др. в рамках представленной концепции удается определить точно, и это дает возможность проверять формальную корректность взаимодействия элементов, которые совместно реагируют на общие события. Мы показали достаточность введенной системы понятий для обеспечения корректного моделирования взаимодействий элементов системы в рамках механизма событий без привлечения глобального времени в виде мировых часов.

Общепринято считать, что механизм синхронизации событий должен удовлетворять требованию локального ограничения причинной связи (local causality constraint), обеспечивающего для модельного времени имитацию естественного порядка событий „от причины — к следствию“ [10]. Понятно, что никакой формализм не в состоянии гарантировать автоматическое выполнение этого или ему подобного требования. Наша формализация не исключение. Однако, используя ее, разработчик модели будет точно знать, какие свойства модельной системы нужно проверять для верификации ее корректности. Организация такой проверки выходит за рамки рассмотрения настоящей работы. В связи с этим стоит упомянуть публикации [11–15], в которых специальное внимание уделяется алгоритмам синхронизации взаимодействия процессов и верификации моделирования времени.

Наша концепция времени как частичного порядка на множестве событий не претендует на роль единственно возможного формального определения системы понятий для моделирования развития. Исторически первым и в некотором смысле альтернативным решением из этого ряда следует считать так называемую квазипараллельную систему дискретных событий языка Симула [16], а затем и Симулы 67 [17], предложенную О. И. Далом и К. Нюгордом еще в 60-х годах.[4] Это решение примечательно тем, что доказывает возможность отображения взаимодействия автономных процессов, о которых программист может и должен думать как о параллельных, в схему последовательно выполняемых программных модулей. При этом эффект параллелизма сохраняется. Вычислительное оборудование того времени не позволяло рассчитывать на реализацию параллельных расчетов, и поэтому пришлось строить строго последовательное выполнение моделей и имитировать параллелизм. Быть может, именно это ограничение позволило авторам проекта Симула предложить сбалансированное решение, которое не перестало быть актуальным и сегодня.

В отличие от модельного времени, определяемого на основе отношений между событиями, квазипараллельная система Симулы и Симулы 67 задает или, лучше сказать, конструирует прогнозируемое модельное глобальное время, упорядочивая элементы системы в соответствии с динамически формирующимся планом выполнения программ этих элементов и главной программы модели, симулирующей поведение внешней среды системы. Для упорядочения выполнения модельных расчетов используется специальная структура данных, называемая *управляющим списком*, в который вносятся элементы (ссылки на них), планируемые для активизации. В каждый момент модельных расчетов единственным активным элементом является тот, на который ссылается первый элемент управляющего списка. Как только элемент перестает быть активным, он удаляется из управляющего

[4]История развития Симулы, его концепций, пользовательских средств и других особенностей представлена в очень интересной публикации авторов языка [18].

списка. Все последующие элементы, представленные в списке, ожидают своей очереди на выполнение.

Процесс, выполняемый элементом, может находиться в одном из четырех состояний:

a) *активном*, когда (реально) выполняется программа процесса;

b) *приостановленном*, когда выполнение программы процесса прервано, но запомнена точка возобновления и процесс находится в управляющем списке;

c) *пассивном*, когда процесс не выполняется и не находится в управляющем списке, но точка возобновления активности запомнена;

d) *завершенном*, когда выполнение его программы прервано, и точка возобновления активности не запомнена.

Конструкция управляющего списка имитирует время. Первый процесс единственный активный. Когда он прерывает свое выполнение, следующим активным становится следующий за ним приостановленный процесс.

Процесс может быть вставлен в управляющий список (перед каким-либо процессом в списке или после него, через определенное время) или удален из него. Процесс также может быть назначен на определенное время. Это означает, что он вставляется перед тем процессом, время выполнения которого —минимальное время, превосходящее назначаемое. Возможно случайное (псевдослучайное) действие по вставке процесса в то или иное место управляющего списка. Можно считать, что с помощью управляющего списка процессам задаются относительные приоритеты. Постулируется, что все оперирование с управляющим списком и с состояниями активности процессов является следствием событий, дискретно происходящих в системе. Пока не произошло какое-либо из событий, реакция на которое предусматривает соответствующие изменения, состояние процесса и его положение в управляющем списке не могут изменяться.

Самое существенное в системе с дискретными событиями Симулы — это то, что ее средства оперирования позволяют программисту думать об элементах системы как о параллельно действующих агентах в условиях неявно определенного глобального времени. Фактически элементы выполняют свои программы последовательно и в строгом соответствии с динамически меняющимся управляющим списком, который обеспечивает глобальное время как полностью упорядоченное множество уже возникших и планируемых к возникновению событий. Вычислительное оборудование времени, когда создавалась Симула, не позволяло рассчитывать на реализацию параллелизма, а потому решение использовать специальную структуру данных, поддерживающую полное упорядочивание активизации элементов модельной системы, предложенное Далом и Нюгордом, вполне оправдано (анализ этой концепции представлен в монографии [19]).

Сегодня хотелось бы видеть развитие систем с дискретными событиями, которое ориентировало бы вычисления на продуктивное использование многопроцессорного ресурса в сочетании с эффектом параллелизма, обеспеченного управляющим списком. Однако эта проблема не так проста, как может показаться при поверхностном рассмотрении. Дело в том, что строго упорядоченная последовательность событий создает видимость дополнительной зависимости процессов, выполняемых как реакции на события. Задача распознавания того, какие из процессов можно выполнять параллельно, очень трудоемка — она требует проведения анализа связей между процессами на уровне всех контекстов, к которым обращаются процессы, что для программ моделирования практически нереально. Нужны подходы, которые отказываются от прямолинейного следования упорядочивания вычислений на базе глобального времени модельной системы.

Очевидный способ организации вычислений, способствующий сокращению расходов на анализ зависимостей, — это сокращение связей процессов через общий глобальный контекст. Применительно к событийному управлению в развивающихся системах это прямо указывает на необходимость отказа от моделирования глобального времени. И здесь надо говорить о развитии языковых моделей вычислений, нацеленном на реализацию удобных и понятных средств поддержки задания локального оперирования. В качестве перспективных направлений в этой области можно назвать языки функционального стиля программирования и системы фрагментированного программирования[5]. К первому из этих направлений относятся чистый Лисп с поддержкой объектно-ориентированного программирования (система CLOS [22]), Haskel [23] и др., ко второму — язык и система программирования Luna [24, 25]. Оба направления доказали свою эффективности при конструировании программ во многих прикладных и системных областях, однако адекватность их использования при моделировании развития систем еще предстоит исследовать.

Современное состояние проблематики моделирования времени отражает подробный обзор В. В. Окольнишникова [26], содержащий достаточно полную информацию о подходах к решению имитации развития. Ценность этой публикации в том, что автор указывает на побудительные причины выбора того или иного способа отражения времени в моделях, исходя из потребностей решения реальных задач моделирования. В частности, он обосновывает причины появления стандарта архитектуры верхнего уровня для систем моделирования и симуляции [27], а также сопутствующих ему стандартов, которые были разработаны в IEEE для унификации методов решения задач в этой отрасли. Автор обзора отмечает, что все рассматриваемые им модели имитационного времени в настоящее время используются в различных областях применения моделирования, а их различия связаны с особенностями этих областей.

Исследование, представленное в настоящей работе, показало, что преодоление трудных проблем глобализации времени на основе мировых часов достижимо, если глобальное время рассматривать как сущность, вторичную по отношению к локальным временам элементов системы. На этой основе можно строить шаблоны проектирования, которые реализуют достаточно общие методы построения моделей эволюционного развития. Выявление ситуаций, в которых требуются такие шаблоны, мы рассматриваем в качестве перспективной работы.

**Приложение. Методологические концепции времени и их связь с видами научно-исследовательской деятельности.**

Понятие времени и его соотношения с понятием пространства занимала исследователей с древнейших времен и до наших дней. В представленной схеме (рис. 1.) представлена классификация сложившихся концепций времени. Основой построенной иерархии служат научно-исследовательские направления, с которыми можно соотнести те или иные концепции. В результате определилось пять кластеров: (1) естественно-научное направление; (2) экономические науки; (3) системные исследования; для направлений

[5]Несмотря на то, что первому функциональному языку Лиспу почти столько же лет, сколько и Фортрану, стартом широкого обсуждения функциональности стал 1978 год после публикации Дж. Бэкусом лекции, которую он прочитал при присуждении ему тьюринговской премии [20]. Фрагментированное программирование [21] не имеет столь же давней истории. Тем не менее, опыт конструирования программ путем подготовки фрагментов, которые для задания требуемых вычислений комбинируются с учетом свойств и особенностей, как составляемых алгоритмов, так и используемой реальной среды вычислений, показывает очень хорошие результаты.


Рис. 1. Классификация сложившихся концепций времени

(4) и (5) сформулировать адекватные названия не удалось. На схеме выделены имена представителей направлений, суждения которых цитируются (в кратком пересказе).

## Список литературы

1. Августин А. Исповедь. Серия „Памятники религиозно-философской мысли“. Пер. с лат. М. К. Сергеенко. 1991. М.: Издательство „Ренессанс“, СП ИВО СиД. ISBN 5-7664-0472-7.

2. Наместников А. М. Разработка имитационных моделей в среде MATLAB // Методические указания для студентов специальностей 01719, 351400. Ульяновск, УлГТУ, 2004.

3. Шевченко А. А. Управление временем при проектировании имитационных моделей // Часть сб. Прикладная информатика. № 3. 2006. Лаборатория Математического и компьютерного моделирования. С. 113–119.

4. Скопин И. Н. Локальное и глобальное время при моделировании развивающихся систем. В сб. трудов Седьмой международной конференции памяти академика А. П. Ершова „Перспективы систем информатики“. Рабочий семинар „Наукоемкое программное обеспечение“. Новосибирск: ООО „Сибирское Научное Издательство“, 2009. С. 255–259.

5. Система. Большой Российский энциклопедический словарь. М.: БРЭ. 2003, С. 1437.

6. Смирнов Г. А. Оккам, Уильям // Новая философская энциклопедия / Ин-т философии РАН; Нац. обществ.-науч. фонд. 2-е изд., испр. и допол. М.: Мысль, 2010. ISBN 978-5-244-01115-9.

7. Скопин И. Н. Иерархические отношения — методологическая основа изучения понятия иерархий // Вестник Российского университета дружбы народов. Серия „Информатизация образования“. М.: РУДН, 2014. № 1. С. 56–63.

8. Скопин И. Н. Субординационные отношения в методике изучения понятия иерархичности // Вестник Российского университета дружбы народов. Серия „Информатизация образования“. М.: РУДН, 2014. № 2. С. 35–49.

9. Дейкстра Э. Взаимодействие последовательных процессов. В сб. „Языки программирования“, под ред. Ф. Женюи. Пер. с англ. М.: „Мир“, 1972.

10. Lamport L. Time, clocks, and the ordering of events in a distributed systems // Commun. ACM. 1978. Vol. 21(7). P. 558–565.

11. Chandy K. M., Misra J. Distributed simulation: a case study in design and verification of distributed programs // IEEE Transactions on Software Engineering. 1978. Vol. SE–5(5). P. 440–452.

12. Ferscha A. Parallel and distributed simulation of discrete event systems // Parallel and Distributed Computing Handbook. McGraw-Hill. 1996. P. 1003–1041.

13. Казаков Ю. П., Смелянский Р. Л. Об организации распределенного имитационного моделирования // Программирование. 1994. № 2. С. 45–63.

14. Fujimoto R. M. Parallel and Distributed Simulation Systems // Wiley Interscience, 2000.

15. Fujimoto R. M. Parallel and Distributed Simulation Systems // Proc. of the Winter Simulation Conf. 2001. P. 147–157.

16. Дал О. И., Нюгорд К. Симула — язык для программирования и описания систем с дискретными событиями // Алгоритмы и алгоритмические языки. Вып. 2. М.: ВЦ АН СССР, 1967.

17. Дал О. И., Мюрхауг Б., Нюгорд К. Симула 67 универсальный язык программирования // Перевод с англ. К. С. Кузьмина и Е. И. Яковлева. М.: Мир, 1969.

18. Nygaard K., Dahl O.-J. The Development of the SIMULA Languages // History of programming languages. ACM New York, NY, USA, 1981. P. 439–480.

19. Непейвода Н. Н. Скопин И. Н. Основания программирования. Изд-во: Москва-Ижевск: Институт компьютерных исследований. 2003. ISBN: 5-93972-299-7.

20. Backus J. Can programming be liberated from the von Neumann style? A functional style and its algebra of programs // CACM, 21(8), August 1978. P. 613–641.

21. Malyshkin V. Assembling of Parallel Programs for Large Scale Numerical Modeling. In the Handbook of Research on Scalable Computing Technologies. IGI Global, USA, 2010. Chapter 13, P. 295–311. ISBN 978-1-60566-661-7.

22. Keene S. Object-Oriented Programming in Common Lisp: A Programmer's Guide to CLOS, 1988, Addison-Wesley. ISBN 0-201-17589-4

23. Haskell 98 Language and Libraries. The Revised Report. Dec. [Electron. Res.]: `https://www.haskell.org/onlinereport/`.

24. Akhmed-Zaki D., Lebedev D., Malyshkin V., Perepelkin V. Automated Construction of High Performance Distributed Programs in LuNA System // PaCT-2019 proceedings, LNCS 11657, Springer, 2019, P. 3–9. DOI: 10.1007/978-3-030-25636-4_1.

25. Ахмед-Заки Д. Ж., Лебедев Д. В., Малышкин В. Э., Перепелкин В. А. Автоматизация конструирования распределенных программ численного моделирования в системе LuNA на примере модельной задачи // Проблемы информатики, № 4, 2019. С. 53–64. DOI: 10.24411/2073-0667-2019-00017.

26. Окольнишников В. В. Представление времени в имитационном моделировании // Вычислительные технологии. 2005. Т. 10, № 5. С. 57–80.

27. IEEE Std P1516. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) — Framework and Rules. N.Y.: Institute of Electrical and Electronics Engineers, Inc., 2000.

**Скопин Игорь Николаевич**, тел: +7 (983) 126-26-86; e-mail: `iskophin@gmail.com` — старший научный сотрудник Института вычислительной математики и математической геофизики (ИВМ и МГ) СО РАН; доцент механико-математического факультета Новосибирского государственного университета (НГУ);

Интерес **Игоря Николаевича** к проблематике разработки языков программирования и к конструированию компиляторов восходит к 1968–70 годам, когда он был студентом механико-математического факультета НГУ и проходил производственную практику в Новосибирском филиале Института точной ме-

ханики и вычислительной техники АН СССР (ныне Новосибирский институт программных систем). В этой организации он принимал участие в разработке ряда систем программирования и инструментов. В 1974–76 годах он руководил группой специалистов, которая конструировала специализированную систему поддержки обработки текстовой информации. Эта работа послужила основой кандидатской диссертации „Функциональное структурирование текстовой информации, его реализация и приложения“ (научный руководитель А. П. Ершов), которую И. Н. Скопин защитил 1981 году.

В 1986 году И. Н. Скопин получил приглашение заведовать лабораторией в Институте информатики и вычислительной техники Академии педагогических наук СССР (после реорганизации — Институт компьютерных систем в обучении). В качестве целей работы этой лаборатории были обозначены исследования и разработка базовых программных средств поддержки преподавания. Проблемы образовательной информатики пришлось решать в период становления института, а потому одной из существенных задач стала организация команды программистов, способной эффективно работать в новой прикладной области. Результативности деятельности И. Н. Скопина в этой области способствовало его преподавание в НГУ. Эта работа всегда была и остается одним из главных направлений деятельности И. Н. Скопина.

Среди других наиболее значимых мест работы И. Н. Скопина следует указать Новосибирский Центр исследований и разработки корпорации Интел, где он участвовал в решении задач, связанных с пакетом математических программ MKL.

В настоящее время И. Н. Скопин является сотрудником Лаборатории синтеза параллельных программ ИВМиМГ СО РАН. Основное направление его исследований связано с проблемами разработки инструментария для решения задач вычислительной математики и математического моделирования.

**Igor Skopin**, tel: +7 (983) 126-26-86; e-mail: iskophin@gmail.com. Senior Researcher of Institute of Computational Mathematics and Mathematical Geophysics (ICMMG) SB RAS; Associate Professor, Mechanics and Mathematics Department of Novosibirsk State University (NSU);

The interest of **Igor Skopin** in problems of programming languages developing and in the construction of compilers dates back to 1968–70, when he was a student in the Mechanics and Mathematics Department of NSU. He had practices at the Novosibirsk branch of Lebedev Institute of Precision Mechanics and Computer Engineering with Academy of Sciences of the USSR (now it is called as the Novosibirsk Institute of Software Systems), where he took part in the development of programming systems and tools. In 1974–76 he led a team of programmers who designed a specialized support system for processing text information. This work served as the basis for his dissertation „Functional structuring of textual information, its implementation and applications“ (supervisor A. P. Ershov), which Skopin defended 1981.

In 1986 Igor Skopin received an invitation to head the laboratory at the Institute of Informatics and Computer Engineering of the USSR Academy of Pedagogical Sciences (after the reorganization it was called as the Institute of Computer Systems in Education). As the goal of this laboratory, research and development of basic teaching support software were identified. The problems of educational informatics had to be solved during the formation of the institute, and therefore one of the essential tasks was the organization of a team of programmers capable of working effectively in a new applied field. The performance of Igor Skopin in this area contributed to his teaching at NSU. This activity has always been and remains one of the most businesses of Skopin.

Among the other most significant places of work Skopin should be pointed out by the Novosibirsk Research and Development Center of Intel Corporation, where he participated in solving problems associated with the MKL mathematical software packages.

Currently Igor Skopin is an employee of the Laboratory of Computational Physics, ICMMG. The main direction of his researches is related to the problems of developing tools for solving problems of computational mathematics and mathematical modeling.

*Дата поступления — 08.06.2020*