УДК: 167.7
DOI: 10.15372/PS20250617
END: ZENACX

В.Ш. Гумиров, А.В. Гумиров, Д.И. Свириденко

О КОНЦЕПЦИИ СЕМАНТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Часть 1. МЕТОДОЛОГИЯ И ТЕОРИЯ КОНЦЕПЦИИ

Настоящая статья открывает цикл работ, посвященный целостному, «объемному» представлению концепции семантического моделирования, включая ее методологические, логико-математические и технологические аспекты, с упором на практические приложения и взаимосвязь концепции с такими разделами ИТ-индустрии как практика программирования и искусственный интеллект. Именно по этой причине центральное место в данном цикле статей будет занимать описание и обсуждение организационно-технологической схемы практического применения положений концепции. И здесь хотелось бы сделать некоторое пояснение.

Исследование по семантическому моделированию были начаты в конце 70-х – начале 80-х годов прошлого столетия и активно ведутся в настоящее время. Основная тема изысканий – создание и развитие положений методологически и теоретически обоснованной технологии автоматического решения интеллектуальных задач как технологии доверенного искусственного интеллекта и применение этой технологии и ее инструментария для решения практических задач. Главные методологические и теоретические усилия этих исследований направлены на создание, изучение и развитие такого декларативного логико-математического формализма описания задач, процедурная семантика которого допускала бы автоматическое извлечение алгоритма решения задачи из ее декларативной спецификации и последующее исполнение

этого алгоритма компьютером. Описание полученных здесь методологических и теоретических результатов и представляет предмет настоящей статьи. Естественно потребовать, чтобы результаты, полученные в данном направлении, должны найти свое применение на практике, что и найдет свое отражение во второй статье цикла, посвященной описанию созданной и развиваемой авторами данного цикла статей технологии семантического моделирования. А поскольку представляет несомненный интерес проблема дальнейшего совершенствования созданной технологии путем, в том числе, применения некоторых инструментов современного искусственного интеллекта (ИИ), в частности, больших языковых моделей (LLM) и генеративного ИИ, то освящению этого вопроса будет посвящена третья статья цикла. При этом следует отметить, что первые две статьи цикла будут все-таки ориентированы на пояснение возможностей концепции семантического моделирования применительно к такому разделу ИТ-индустрии как практическое программирование. Дело в том, что в последнее время в информатике активно обсуждается тема глубокой автоматизации процесса «ручного» программирования и, в частности, этапа кодирования. Особенно эта тема стала популярной в связи с появлением больших языковых моделей. Оказалось, что концепция семантического программирования также вполне успешно может быть использована для решения данной проблемы. Именно по этой причине обсуждению этой возможности и посвящены будут первые две статьи упомянутого выше цикла. Связь же концепции семантического моделирования и искусственного интеллекта составит предмет третьей работы цикла.

Ключевые слова. Информатика, математическая логика, язык исчисления предикатов, теория конструктивных моделей, семантика, методология, теория программирования, No-code и Low-code, исполнимые спецификации, задача, критерий решенности задачи, контекст решения задачи, семантическое моделирование, онтология, искусственный интеллект.

V.S. Gumirov, A.V. Gumirov, D.I. Sviridenko

ABOUT THE CONCEPT OF SEMANTIC MODELING

Part 1. METHODOLOGY AND THEORY OF THE CONCEPT

This article opens a series of papers devoted to a holistic, comprehensive presentation of the concept of semantic modeling, including its methodological, logical-mathematical, and technological aspects, with an em-

phasis on practical applications and the concept's relationship with such areas of the IT industry as programming practice and artificial intelligence. For this reason, the central focus of this series of articles will be the description and discussion of the organizational and technological framework for the practical application of the concept's principles. A few clarifications are in order. Research on semantic modeling began in the late 1970s and early 1980s and is actively pursued today. The main focus of this research is the creation and development of a methodologically and theoretically sound technology for automatically solving intelligent problems as a trusted artificial intelligence technology, and the application of this technology and its tools to solving practical problems. The primary methodological and theoretical efforts of this research are aimed at creating, studying, and developing a declarative logical-mathematical formalism for problem description whose procedural semantics would allow for the automatic extraction of a problem-solving algorithm from its declarative specification and subsequent execution of this algorithm by a computer. The description of the methodological and theoretical results obtained here is the subject of this article. Naturally, the results obtained in this area should find practical application, which will be reflected in the second article in the series, dedicated to describing the semantic modeling technology created and being developed by the authors of this series. Since the problem of further improving this technology by, among other things, applying certain tools of modern artificial intelligence (AI), in particular, large-scale language models (LLM) and generative AI, is of undoubted interest, the third article in the series will be devoted to this issue. It should be noted that the first two articles in this series will focus on explaining the potential of semantic modeling as it applies to practical programming in the IT industry. The topic of deeply automating manual programming, particularly the coding stage, has recently been actively discussed in computer science. This topic has become especially popular with the advent of large language models. It turns out that semantic programming can also be successfully used to solve this problem. For this reason, the first two articles in this series will be devoted to discussing this potential. The relationship between semantic modeling and artificial intelligence will be the subject of the third article.

Keywords: Computer science, mathematical logic, predicate calculus language, constructive model theory, semantics, methodology, programming theory, no-code and low-code, executable specifications, problem, problem solution criterion, problem solution context, semantic modeling, ontology, artificial intelligence.

ВВЕДЕНИЕ

Современным направлением развития человеческого общества, носящее фундаментальный характер, принято считать цифровую трансформацию, опирающуюся на активное и широкое использование цифровых технологий. Центральное место среди таких цифровых технологий естественно занимают **инструментальные технологии** – те технологии, с помощью и средствами которых создаются разнообразные программные приложения. Применение инструментальных технологий помимо понятного и очевидного их назначения в тоже время зачастую осложняется целым спектром негативных моментов, где ключевыми являются:

- потеря смысла поставленных задач в процессе программирования;
- высокая стоимость разработки и внедрения создаваемого программного обеспечения (ПО);
- повышение стоимости поддержки и сопровождения (владения, роста, устойчивости) ПО с течением времени;
- снижение качества ПО за счет несоответствия ожиданий заказчика и итогового результата;
- повышение стоимости обеспечения качества ПО;
- снижение эффективности ПО, в том числе и потому, что:
  - возникают эффекты, о которых никто не знает, как успешно ими пользоваться и для чего они нужны;
  - с течение времени снижается безопасности ПО, включая то, что недокументированные свойства ПО могут приводить к непредсказуемым сценариям использования ПО.

Основная тенденция решения проблем, указанных выше, – это использование таких инструментальных технологий, которые ориентированы на **глубокую автоматизацию фаз/стадий/этапов жизненного цикла** программных систем. При этом наиболее актуальной и перспективной объявлена тема, связанная с автоматизацией фазы создания программ, начиная от спецификации их назначения (ПОЧЕМУ? ЗАЧЕМ? ЧТО?) и определения пользователей (для КОГО?) и заканчивая этапами внедрения и эксплуатации созданного программного продукта.

Как известно, существует несколько видов организации процессов создания программных систем, среди которых наиболее известны такие подходы, как **каскадный, спиральный** и **инкрементальный (инкрементный)**, включая современный вариант последнего – **непрерывная интеграция Agile,** о котором мы подробнее поговорим позже**.** Знакомясь с этими подходами, можно заметить, что при всей внешней непохожести им всем присуще наличие определенной **инвариантной организационно-технологической схемы действий** по разработке, внедрению и эксплуатации программных систем. Данная «инвариантная» организационно-технологическая схема включает в себя следующие основные виды деятельности:

- ***Постановка задачи*** (Statement of the problem) и ***управление требованиями*** (Requirements management) – на этом этапе обычно уточняются требования к системе, оформляемые в виде так называемого **Технического задания (ТЗ);**
- ***Проектирование*** (Design), одним из результатов которого является **архитектура** будущей системы;
- ***Планирование*** (Planning);
- ***Разработка*** (Developing)/***Кодирование*** (Coding);
- ***Тестирование*** (Testing)/***Обеспечение качества*** (QA – Quality Assurance)/ ***Стабилизация*** (Stabilizing);
- ***Развертывание/Внедрение*** (Deployment); один из важнейших видов программистской деятельности, который, обычно подразделяется на несколько видов работ (смотри, например, https://www.qovery.com/blog/day-0-day-1-day-2-what-are-the-differences/), среди которых отдельно выделяют развертывание и мониторинг, включая работы по оптимизации, куда входят как хорошо зарекомендованные, так и новые подходы типа PGO (Profile Guided Optimization) – оптимизация с использованием информации о статистике работы созданного программного обеспечения.
- ***Эксплуатация*** (Operation);
- ***Вывод из эксплуатации*** (Decommissioning).

При этом требуется, чтобы для каждой фазы/стадии/этапа должно быть определено:

- ***ЧТО является результатом***;

- ***над ЧЕМ работает каждая ролевая модель проектной команды.***

Что же касается **ролевых моделей**, упомянутых выше и которые выполняют участники проектной команды, зачастую совмещая их, то обычно используются следующие:

- ***управление проектом*** *(project manager)* – общее управление проектом в т.ч. планирование, трекинг; руководитель проекта несет полную и неделимую ответственность перед заказчиком;
- ***архитектор*** (*architect*)***;***
- ***разработка*** *(developer)* – разработка приложений и инфраструктуры, технологические консультации;
- ***обеспечение качества и тестирование*** *(QA)* – планирование и реализация регламентов обеспечения качества, в т.ч. планирование, разработка тестов и отчетность по тестам;
- ***удовлетворение заказчика*** *(user experience)* – обучение, эргономика, графический дизайн;
- ***техническая поддержка*** (*technical support/configuration management*) – в т.ч. ***управление релизами*** *(release manager)*, инфраструктурой, сопровождение, бизнес-процессы, выпуск готового продукта, DevOps и /DevSecOps;
- ***управление продуктом*** *(product manager/key account manager)* – бизнес-приоритеты, маркетинг, представительство интересов заказчика;
- ***поддержка методологии в инструментальной технологии*** (*methodologist*).

Как известно, ключевой момент любой организационно-технологической схемы – это **управление проектом**. И здесь одним из самых распространенных инструментов управления проектом является ***иерархическая структура работ (Work Breakdown Structure, WBS***).

Естественно, что ориентация на автоматизацию процессов создания программной системы как решение указанных выше проблем может затрагивать либо разные стадии, либо всю схему в целом. В последнее время наиболее активно обсуждается глубокая автоматизация **стадии/этапа проектирования,** рассматрива-

емая с точки зрения возможности автоматической генерации кода будущей системы, отталкиваясь лишь от различных моделей будущей программы, полученных при ее проектировании. Здесь различают два уровня автоматизации генерации кода – **Low-code** и **No-code**.

Что касается последнего, то речь идет о практически полной автоматизации процесса генерации кода без привлечения какого-либо «ручного» кодирования. В качестве идеального случая приводится пример больших языковых моделей LLM, позволяющих это делать, отталкиваясь просто от спецификации программы, сформулированных на естественном языке. Основную идею подхода No-code можно выразить в виде вопросов - «Зачем вообще что-то программировать? Можно ли обойтись без написания алгоритмов на языке программирования, если задачи простые?». Такие вопросы обычно задают представители компаний, у которых управленческий аппарат небольшой и потому штатный сотрудник со специальными знаниями программирования будет необоснованными затратами, поскольку он просто не будет достаточно загружен для постоянной работы, а его периодическая занятость тоже не всегда эффективна. И предлагаемый здесь выход заключается обычно в использовании **шаблонов** – готовых проработанных решений, допускающих их выбор с помощью уточнения **внешних параметров** типа «роль пользователей/стейкхолдеров», «вид договора», «регламенты согласования» и т.п., и настройки значений **внутренних параметров** шаблона с учетом особенностей решаемой проблемы.

Более интересным на взгляд авторов является уровень автоматизации кодирования Low-code, рассматриваемый как способ разработки программного обеспечения, при котором к кодированию «вручную» прибегают, но стараются это делать в минимальном объеме. С этой целью используются различные инструменты, например, визуальные конструкторы или готовые скрипты как шаблоны решений. Главную ценность при этом подходе составляет возможность обойтись в отдельных случаях вообще без программистов, когда нужно создать или изменить какое-то приложение, модуль или даже весь продукт.

Для этих целей создают специальные инструментальные **Low-code платформы**, позволяющие существенно сократить время на разработку и предоставить больше гибкости в настройке процессов. При этом не нужно планировать архитектуру системы, создавать прототипы, анализировать и разрабатывать пользовательский интерфейс - это все реализовано в самой платформе Low-code. Такие платформы должны уметь интегрироваться с широким набором языков программирования и сервисных систем, а также позволять легко и просто добавлять новые функции в любое приложение. Кроме того, производители Low-code платформ в качестве достоинств говорят об их б*о*льшей безопасности для других приложений и стабильности работы по сравнению с создаваемыми в процессе выполнения проекта конструкциями. Ниже указываются основные виды возможностей, предоставляемые современными Low-code/No-code платформами:

- ***Визуальное моделирование;***
- ***Готовые компоненты, встроенные сервисы;***
- ***Быстрое развёртывание приложений;***
- ***Ориентация на DevSecOps;***
- ***Разработка по шаблонам или абстрактная, без привязки к шаблонам.***

Однако эксплуатация платформ Low-code может вызвать и значительные трудности при реализации сложных проектов, поскольку в них обычно требуется качественная проработка архитектуры системы, обеспечение информационной безопасности, рутинное улучшения взаимодействия с пользователем и т.д. и т.п. Так, например, у Low-code систем, генерирующих код, достаточно часто возникает проблема в том, что код генерируется абсолютно неадекватный, нечитаемый и неподдерживаемый. Данные обстоятельства возлагают на Low-code платформу особую ответственность – она должна гарантировать использующим ее администраторам, аналитикам и экспертам предметных областей, что предоставляемые ею сервисы покрывают значительное число навыков и компетенций, необходимых для разрешения внештатных ситуаций и свойственных опытным программистам. При этом, когда действительно требуется «ручная» коррекция, то Low-

code платформа должна позволять привлечение классических программистов и гарантировать легкое и естественное их встраивание в общий цикл работ.

Идея глубокой автоматизации процессов проектирования в режиме Low-code может быть реализована многими способами. Одним из таких подходов и может рассматриваться практическая реализация **концепции семантического моделирования**, выдвинутая еще в 70-80-е годы прошлого столетия советскими математиками академиками Ю.Л.Ершовым, С.С.Гончаровым и д.ф.-м.н. Д.И.Свириденко [1-5]. Эта концепция получила свое практическое воплощение в нескольких инструментальных платформах. В данной статье авторы отталкиваются от практики разработки программных систем средствами платформы **ESDP** с ее базовым языком **d0sl**, созданными более 10 лет тому назад Группой компаний Eyeline под руководством В.Ш.Гумирова (www.eyeline.ru). Эта Группа компаний в настоящее время продолжает успешно эксплуатировать и развивать данную инструментально-технологическую платформу, которая позволяет уже на этапе проектирования сразу писать **исполняемый вариант** модели программного приложения, максимально минимизируя «ручное» кодирование бизнес-логики.

И еще одно замечания, касающееся возможности технологического развития платформы и ее базовых инструментов. Как хорошо известно, в мире в настоящее время только менее трети программных проектов завершаются вовремя и в рамках бюджета. Еще одна пятая часть проектов отменяется до завершения. Другими словами, только около половины программных проектов можно фактически считается «успешными» с точки зрения своевременности и бюджета. Существует много причин, по которым проект может выйти за рамки первоначального времени и бюджета и о некоторых из них речь шла выше. Но при внимательном анализе этих причин выясняется, что большая их часть сводится к **проблеме организации коммуникации** участников проекта, что свидетельствует о том, что традиционный процесс разработки программных систем нуждается в новых методологиях и технологиях разработки, желательно поддержанных серьезной математической теорией. Среди таких подходов, как уже упоминалось

ранее, весьма влиятельными стали методология и технология разновидности инкрементального программирования, известных под названием технология **непрерывной интеграции Agile** (https://www.jetinfo.ru/chto-budet-posle-agile-nepreryvnaya-integratsiya). Напомним, что основные цели Agile заключаются в том, чтобы:

- *постараться максимально сблизить владельцев и будущих пользователей с технической командой проекта,*
- *выпускать программное обеспечение более частыми циклами,*
- *адаптироваться к меняющимся потребностям клиентов, а не придерживаться плана.*

Статистика утверждает, что в результате проекты по разработке программного обеспечения с использованием Agile стали чаще и больше соответствовать ожиданиям заказчика, хотя и сопровождаются рядом недостатков и неприятностей. И здесь может оказаться полезным разумное сочетание концепции семантического моделирования и технологии Agile. Например, могут быть позаимствованы друг у друга некоторые их идеи и инструменты и, тем самым, взаимно усиливая их. В частности, концепция семантического моделирования может постараться освоить и внедрить идею ***инкрементальности,*** присущую Agile. В свою очередь, использование технологией Agile концепции и языкового инструментария исполнимых спецификаций из семантического моделирования способно сократить расходы на разработку программного обеспечения, повысив его качество и ускорив сроки поставки. Это становится возможным, в частности, и в силу того, что технология семантического моделирования уделяет большое внимание организации процесса создания конкретных тестовых примеров еще на этапе проектирования с тем, чтобы уменьшить неоднозначность пользовательских историй и критериев приемки. Вообще говоря, замечания, касающиеся взаимоотношений концепций семантического моделирования и Agile, могут представлять интерес практически для любого программного проекта, требующего общего вклада всех заинтересованных сторон, поскольку, например, тестовые примеры, которые будут созданы средствами семантического тестирования, намного более понят-

ны и значимы для всех в команде, начиная с заказчика. Это отличный способ значительно сократить «отходы производства», которые обычно имеют место в программных проектах. И при этом гарантируя, что проект будет осуществлен вовремя и в рамках бюджета.

Данная статья посвящена изложению методологических и теоретических аспектов концепции семантического моделирования, начиная с ключевых методологических принципов и положений и заканчивая логико-математическими основами концепции семантического моделирования.

1. МЕТОДОЛОГИЯ СЕМАНТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Основным и истинным содержанием деятельности любого творческого человека является **постановка и решение задач.** При этом он может прибегать к самым различным вариантам методологических и инструментальных средств. Естественно возникает вопрос – а что из себя представляет сама задача как некая сущность? На этот вопрос и отвечает оригинальный **задачный подход** [6-12], основные положения которого будут изложены ниже. Но начнем с истории.

Впервые о задачах, как об сущностях, заслуживающих отдельного внимания и тщательного изучения, заговорил знаменитый советский математик академик А.Н.Колмогоров в своей ранней работе [13], опубликованной еще в 1932 году. В этой работе А.Н.Колмогоров, описывая в терминах задач денотационную семантику интуиционистского исчисления высказываний, фактически впервые изложил аксиоматику **исчисления задач**. Согласно интерпретации А.Н.Колмогорова значениями пропозициональных переменных в формулах являются любые задачи. И если p и q задачи, то формулы (p&q), (p∨q), (p⊃q) и ¬p толкуются, соответственно, как задачи: "Решить задачу p и задачу q", "Решить задачу p или задачу q", "Свести решение задачи q к решению задачи p" и "Предполагая задачу p решенной, прийти к противоречию". При такой интерпретации предполагается, что каждой доказуемой

в интуиционистском исчислении высказываний формуле Ψ(p1,p2,...,pn) должен соответствовать класс задач, имеющих **общий метод решения,** не зависящий от конкретного содержания задач p1, p2, ..., pn. При этом формуле (p ∨ ¬ p), не доказуемой в указанном логическом исчислении, поскольку она выражает принцип исключенного третьего, соответствует класс задач вида "Решить задачу p или, предполагая задачу p решенной, прийти к противоречию". Следует заметить, что в год написания А.Н. Колмогоровым упомянутой выше статьи, посвященной интерпретации интуиционистского исчисления высказываний, оставалось еще не выясненным, что следует понимать под «общим методом решения» задач некоторого класса и что значит «сведение решения одной задачи к решению другой». Это было уточнено (и притом различными эквивалентными способами) после того, как начиная со второй половине 30-х гг. прошлого века были предложены различные варианты формализации понятий алгоритма и вычислимой (рекурсивной) функции. Напомним, что появление понятия алгоритма позволило строго математически доказать несуществование общего метода (алгоритма) решения задач из класса, соответствующего формуле (p∨¬p). Действительно, существование такого алгоритма означало бы, в частности, существование алгоритма решения задач вида "Доказать выводимость формулы Ψ в исчислении S или, предполагая формулу доказуемой в нем, прийти к противоречию" (в качестве p взята задача "Доказать выводимость Ψ в исчислении S"). Алгоритм решения задач этого вида при некотором конкретном S является решением проблемы разрешимости исчисления S. Но, как показал Черч в 1936 году [14], эта проблема неразрешима, в частности, для узкого исчисления предикатов.

В последствии концепция А.Н.Колмогорова была перенесена на интуиционистское исчисление предикатов, где формулы также трактовались как задачи, логические связки – как преобразования задач, аксиомы – как задачи, для которых решения считаются известными, а правила вывода – как преобразования решений задач. Семантика же формул задавалась в терминах реализаций: каждой формуле A сопоставляется множество реализаций □A, где под **реализацией задачи** A по-

нимается **решение** этой задачи. Такие множества реализаций задаются согласно следующим определениям:

- ⊕(A&B) = ⊕A × ⊕B, где ⊕(A&B) – это пара множеств реализаций <⊕A, ⊕B>;
- ⊕(A∨B) = ⊕A + ⊕B, где ⊕(A∨B) – реализации подзадач A или B с точным указанием того, какая из этих подзадач решена;
- ⊕¬A = 0 ⇔ ⊕A = ∅, где множество реализаций ⊕¬ A представляет собой некий стандартный символ (например, как в нашем случае, 0), при условии, что задача A неразрешима;
- ⊕∃xA(x) = a∈U⊕A(α), где ⊕∃xA(x) – это множество пар вида <значение a переменной x, реализация задачи A(a)>;
- реализациями (A ⇒ B) являются вычислимые функционалы из ⊕A в ⊕B;
- реализациями ∀xA(x) являются вычислимые функционалы, перерабатывающие каждое α∈ U в реализацию A(α) (здесь U – некий универсум данных).

Формальное изучение задач, начатое в 30-е годы А.Н.Колмогоровым, было независимо продолжено исследователями в других дисциплинах, среди которых, на взгляд авторов, заметное место занимает **Теория Решения Изобретательских Задач** (**ТРИЗ**), предложенная в 50-е годы прошлого столетия советским инженером-патентоведом Г. Альтшуллером [15] (смотри также https://www.livelib.ru/author/194901/top-genrih-altshuller). В его концепции задачный подход формулировался как сугубо **эмпирическая теория**, ориентированная, главным образом, на практическое применение ее положений при решении задач анализа и синтеза **искусственных систем (ИС),** примерами которых могут служить: самолет, автомобиль, телевизор, смартфон, компьютер, программа, организация и т.д.

Согласно ТРИЗ, главной движущей силой развития человеческого общества является непрерывное развитие человеческих потребностей, где под **потребностью** понимается нужда в чем-либо, необходимом для поддержания «жизнедеятельности» индивида/ сущности, социальной группы, организации, общества, внутрен-

ний побудитель активности. Развитие потребностей в свою очередь управляется серией **законов развития**, главным из которых является **закон повышения идеальности потребностей**, который подразумевает, что с каждой совокупностью потребностей может быть связан некий уровень идеальности этой совокупности, а само повышение идеальности проводится определенными действиями (динамизацией, объединением или специализацией с последующим согласованием). **Идеальная потребность** – это такая потребность, которую нет необходимости удовлетворять, поскольку она либо к моменту ее удовлетворения стала ненужной, либо потребность удовлетворяется сама. К основному свойству идеальной потребности относится требование, чтобы потребность всегда гарантированно удовлетворялась в нужный момент и в нужном месте при выполнении необходимых условий. При этом, потребность тем идеальнее, чем качественнее она удовлетворяется - чем меньше затрачивает времени, сил и средств на ее осуществление, чем меньше процесс ее удовлетворения создает отрицательных эффектов (вредных факторов, вредных воздействий), как непосредственно объекту удовлетворения потребностей, так и его окружению. Кроме того, ТРИЗ предложила использовать семейство эмпирических формул, позволяющих алгоритмически оценивать степень идеальности ИС из разных областей и находящихся на разных этапах их создания. Полезное свойство таких формул заключается в том, что их анализ позволяет реально указать те действия, которые помогают повысить идеальность ИС.

Заслугой ТРИЗ является также то, что в рамках этой теории была разработана и обоснована общая схема деятельности по удовлетворению потребности в предположении, что эта схема должна завершиться синтезом нужной ИС: искусственная система, удовлетворяя некую **потребность,** должна выполнять ту или иную **функцию,** осуществляя, тем самым, определенный **процесс**, основанный на конкретном **принципе действия,** на котором и будет базироваться **синтезируемая ИС.** Естественно, что такая ИС будет иметь определенную структуру и потоки (материи, энергии, информации), а также может включать, как искусственные, так и природные элементы. Подобный взгляд на структуру

и содержание этапов схемы **синтеза ИС** (обратная ей схема есть ничто иное, как схема **анализа ИС**) позволил ТРИЗ ответить на фундаментальные вопросы задачного подхода – **откуда берутся задачи, каков их истинный источник и что собой представляет задача?**

Согласно ТРИЗ (смотри Рис.1 ниже) **задача** фактически есть ничто иное, как **потребность преодолеть противоречие,** между желаемым и имеющимися возможностями у ее Решателя на каждом этапе синтеза/анализа ИС.

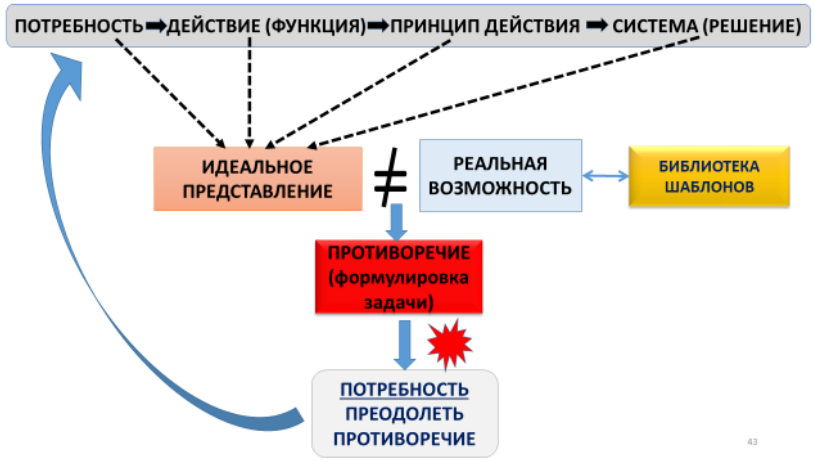

Рис. 1. Источник возникновения задачи

Дальнейшее значительное развитие задачный подход получил в работах академика Ю.Л.Ершова и д.ф.н. К.Ф.Самохвалова, посвященных применению идей и положений задачного подхода к основаниям математики [16,17]. Их анализ понятия задачи применительно к основаниям математики начинался со следующих простых рассуждений: «*Я хочу пить» – что это значит? Нет, конечно, никакой ошибки полагать, что слова «я хочу пить» означают просто вот это, где это – определенное состояние сознания, которое я переживаю сейчас и которое я именую жаждой. Но тогда возникает новый вопрос: как ощущение жажды (хотения) связано с фактическим питьем (удовлетворением хо-*

тения)? Откуда я знаю, что удовлетворить жажду можно питьем? Содержится ли в самом переживании жажды сознание того, чем эту жажду можно удовлетворить? ... Знать желание не означает знать желаемое, а означает способность узнать желаемое». Последнее замечание является ключевым в данном рассуждении и означает ничто иное, как требование наличия **критерия удовлетворения желания**. Таким образом, согласно вышесказанному, задача определена (осмыслена) тогда и только тогда, когда у нас есть **критерий решенности** – критерий проверки того, что действительно ли предъявленное решение является решением задачи. В противном случае любое предлагаемое соображение можно считать решением задачи. В математических теориях таким критерием решенности считается наличие доказательства решения задачи. Но этот критерий применим корректно только тогда, когда в рамках самой формальной системы мы имеем как доказательство решения задачи, так и возможность убедиться средствами самой же системы, что данное доказательство действительно является решением задачи.

Вообще говоря, таких критериев решенности может быть несколько. Так, например, в программировании правильность написанного кода может подтверждаться, например, либо фактом удовлетворения определенному конечному набору **тестов**, либо успешной **верификацией** программы – критерий, аналогичный математическому критерию – наличие доказательства. Заметим, что *каждый такой критерий определяет, тем самым, свое оригинальное понимание смысла задачи*.

Положения, сформулированные и обоснованные в работах Ю.Л.Ершова и К.Ф.Самохвалова, в последующем были развиты и адаптированы коллективом математиков Института математики СО РАН в работах [6–12] применительно к искусственному интеллекту. Потребность практического применения задачного подхода для автоматического решения как можно большего класса конкретных задач искусственного интеллекта заставила авторов более детально рассмотреть само понятие «задача». Результатом такого рассмотрения явилось формулировка этого понятия вначале на методологическом уровне, а затем и на формальном, математическом, что привело к созданию оригинальной логико-

вероятностной теории **семантического моделирования** [1-5,18-36]. Поэтому, прежде чем формулировать положения этой теории, приведем вначале неформальное определение понятия задачи с тем, чтобы в следующем разделе статьи указать его формальный аналог.

Итак, будем считать, что задача определена в том и только в том случае, когда в ее формулировке присутствуют:

- указание **предметной области**, зафиксированное в виде некой модели, включая описание сигнатуры и структуры предметно-ориентированного языка (онтологии), а также общие знания и знания о предметной области: исходные данные, факты/прецеденты, правила, ограничения и гипотезы;
- на какой **запрос (вопрос),** сформулированный в задаче к модели предметной области, мы должны получить **ответ (решение задачи**);
- **критерий удовлетворения запроса** – в каком случае можно считать, что ответ (решение) на запрос (вопрос) действительно получен;
- в каком **контексте** следует искать и интерпретировать ответ (решение) на запрос (вопрос) – какую цель мы преследуем, решая задачу, т.е. что мы ожидаем от полученного результата и каковы его последствия и что делать, если ответ окажется отрицательным.

Данное определение позволило авторам концепции семантического моделирования сформулировать и решить уже на математическом уровне проблему автоматизации постановки и решения задач с предъявленным обоснованием соответствия критерию решенности, ориентируясь на то, что задачи будут формулироваться в терминах **исполнимых (декларативных) спецификаций**.

Опираясь на вышесказанное, мы закончим данный раздел методологическими рекомендациями об организации процесса решения задач в рамках и средствами семантическом моделировании, что будет ориентиром для теоретических изысканий и составит основу описания его технологических аспектов во второй статье упомянутого ранее цикла работ.

Согласно методологическим принципам и положениям, изложенным выше, предлагается при решении задачи в рамках задачного подхода и средствами концепции семантического моделирования придерживаться следующей схемы действий:

ШАГ 1. ***Выявление и анализ потребностей***.

Изучается, уточняется и осмысливается возникшая **потребность** и делается попытка определить возможность использовать тот или иной имеющийся **шаблон**, позволяющих ее удовлетворить. В случае отсутствия подходящего «шаблонного» способа удовлетворения потребности выявляется и изучается связанное с этим возникающее **противоречие** между желаемым и реальным состоянием дел, которое и является истинной причиной и содержанием решаемой **задачи**. Более того, осознанная формулировка потребности и есть, фактически, формулировка задачи.

Заметим, что результаты анализа, предпринятого на этом шаге (*история - почему?, цель - зачем?, от кого какие данные (предусловие) и для кого/кому какие данные (постусловие)?, где?, чем?, кем?, возможные последствия решения/не решения задачи* и т.д.) в последующем составят содержание основных компонент задачи. Кроме того, они могут привести к выводу о необходимости декомпозиции исходной потребности на конечный набор взаимосвязанных **субпотребностей,** что повлечет за собой возникновение целого набора противоречий и необходимость поиска ответов на вопросы**:** каковы **подзадачи-противоречия** и что есть ближайшая **надзадача** для каждой подзадачи-противоречия. В этом случае цикл операций *ШАГ 1* для каждой выявленной субпотребности и подзадачи-противоречия повторяется.

ШАГ 2. ***Планирование.***

Разрабатывается план осуществления действий по формулировке (предпочтительно на естественном языке) выявленной задачи или набора подзадач, включая:

- создание модели той **предметной области**, к которой принадлежит данная задача,
- описание **запроса (запросов)**, требующего ответа(ов),
- формулировку **критерия успешности** преодоления выявленного противоречия; обычно этот критерий формулиру-

ется как набор **тестовых примеров,** выступающий как некий прототип **критерия решенности задачи;** возможны и другие критерии успешности, например, в виде верификационного алгоритма,

- анализ условий организации процесса решения задачи, рассматривая полученные при этом результаты как начальный вариант ее **контекста**.

ШАГ 3. ***Формирование видения (концепции) проекта.***

На естественном языке описываются общие и специальные, оценочные знания, имеющее отношение к поставленной задаче(ам), собираются релевантные факты и другие прецеденты, строится онтологическая модель проблемной области (понятия, отношения, свойства и т.п.), формулируется в общих и онтологических терминах класс возможных запросов к проблемной области, раскрывающих содержание поставленной общей задачи и ее подзадач, формулируется вариант критерии решенности задачи и ее подзадач. Выделяется та часть представлений о задаче и ее предметной области, которая требует использования либо внешних вычислителей как оракулов, либо «базовых» сущностей, уже полученных или получаемых в процессе выполнения проекта «ручным способом», которые, кстати, тоже можно воспринимать как оракулы. При этом естественно нужно стремиться к минимизации набора оракулов, требующих «ручного» исполнения в рамках и средствами традиционного программирования. Заметим, что если компетенции и профессионализм специалиста(ов), занимающихся спецификацией общей задачи и ее подзадач, позволяют, то можно сразу переходить на *ШАГ 4*.

ШАГ 4. ***Формализация семантического предметно-ориентированного языка (sDSL) предметной области и постановки задачи в терминах этого языка***

Строится формальная модель проблемной области задачи и в этих же формальных терминах формулируется запрос и критерий решенности задачи с учетом контекстных условий. Программируется выделенный набор отсутствующих оракулов и решается вопрос коммуникации с уже существующими оракулами. Если выбран критерий решенности в виде тестирования с помо-

щью семантических тестовых примеров, то пишутся их формальные спецификации.

Заметим, что если *ШАГ 4* осуществляется в рамках и средствами соответствующей цифровой **инструментальной платформы** семантического моделирования, то мы можем сразу получить компьютерный вариант системы, ориентированной на решение задачи.

ШАГ 5. ***Отладка***

Используя выбранный вариант проверки выполнимости критерия решенности задачи (семантическое тестирование с помощью тестовых примеров или верификация спецификаций и самописанных «ручным» способом оракулов или имитационное моделирование) проверяется интерпретируемость, выполнимость, обоснованность и/или объяснимость функционирования компьютерной системы.

ШАГ 6. ***Эксплуатация***.

С помощью компьютерной системы в режиме ее эксплуатации находятся ответы на возникающие запросы, составляющие потенциальное содержание решаемого класса задач.

2. МАТЕМАТИЧЕСКИЕ ОСНОВЫ СЕМАНТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Чтобы перейти к формальному уточнению понятия задачи необходимо было предварительно проанализировать основные активно использующиеся на практике различные модели вычислимости. Эти модели естественно классифицируются на **императивные,** в которых вычисление есть **процесс** управления состоянием памяти Вычислителя, и на **декларативные,** где вычислимость предстает как **выводимость**. Так, например, ключевой идеей функционального и логического программирования является как раз концепция, согласно которой программа рассматривается как **теория**, а вычисление, ею генерируемое, представляет собой **вывод** в этой теории. Понятно, что чем меньше ограничений накладывается на допустимые способы построения таких теорий,

тем проще пользователю специфицировать задачу. Но, к сожалению, имеет место и другое наблюдение – чем меньше ограничений, тем менее эффективным будет алгоритм интерпретации теорий и, следовательно, алгоритм организации поиска вывода. Данное противоречие заставляет сторонников и апологетов декларативного программирования искать разумный компромисс между удобством и выразительностью существующих декларативных языков и их эффективностью. Такой поиск может осуществляться либо в рамках существующих парадигм функционального и логического программирования, либо в их критическом осмысливании и, возможно, создании новой парадигмы.

Как известно, фундаментальным изучением понятия вычислимости в математике занимается математическая логика и осуществляет это в рамках двух основных подходов: **аксиоматическом** и **теоретико-модельном**. Как было замечено выше, именно аксиоматический подход и составляет математическую основу существующего декларативного программирования. Но ведь в качестве концептуальной основы технологии компьютерного решения задач можно выбрать и **теоретико-модельный подход**, взяв, скажем, в качестве математической базы концепцию **формульной определимости вычислимости на конструктивных моделях** [36], расширив ее, скажем, **оракульной** (относительной) **вычислимостью,** а общий процесс вычислимости мыслить, например, либо как **процесс проверки истинности формулы** на конструктивной модели, либо как **вычисление конкретного терма,** если в качестве языка спецификации задач выбран некий достаточно выразительный фрагмент языка исчисления предикатов**.** Эта идея и была положена в основу **семантического моделирования,** а предлагаемый вариант модели вычислимости получил название **семантической вычислимости.** Коротко, суть концепции семантического моделирования заключаться в следующем.

Пусть нам дана исходная многосортная **базовая модель** $\mathfrak{M}$, выступающая как ядро некого **Вычислителя**, средствами которого далее будут специфицироваться и решаться задачи. При этом предполагается, что все объекты данной модели (элементы, функции, предикаты) и логические операции над ними являются **конструктивными,** т.е. **вычислимыми.** Кроме того, предлагает-

ся эту модель рассматривать только вместе с ее **списочной надстройкой HW(𝔐),** состоящей из наследственно-конечных списков, порожденных элементами базовой модели, что, кстати, позволяет работать практически с любыми структурами данных, используемыми в традиционном программировании. Кроме того, это важнейший инструмент повышения уровней абстракции, что крайне необходимо при решении реальных задач.

Более точно, это формулируется так. Пусть M некоторое множество. Определим по индукции множество HW(M) наследственно конечных списков над M:

$HW0(M) = \{\emptyset\}$;

$HW_{n+1}(M) = HW_n(M) \cup Lisp_\omega(HW_n(M) \cup M)$;

$HW(M) = \bigcup_{n=0}^{w} HWn\ (M)$,

где $Lisp_\omega(X)$ есть множество всех конечных списков, состоящих из элементов множества X.

Пусть теперь 𝔐 – это многосортная конструктивная модель сигнатуры σ, где M – ее базисное множество и пусть (HW(M) ∪ M) – базисное множество расширенной модели HW(𝔐) сигнатуры σ* = σ ∪ {nil, head, tail, conc, cons, =, ϵ, ≤, U}, где новые функциональные и предикатные символы имеют естественную интерпретацию и где U(HW(M)) = M. Будем называть модель *HW*(𝔐) сигнатуры σ* **надстройкой наследственно конечных списков для модели 𝔐** или просто **списочной надстройкой над 𝔐** (Рис.2).

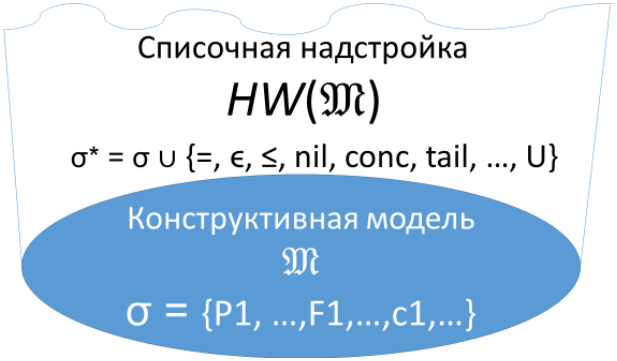

Рис.2. Модель и ее списочная надстройка

При этом допускается расширение сигнатуры σ модели ***HW*(𝔐)** до сигнатуры σ* набором предикатных и функциональных переменных, которые интерпретируются как **оракулы,** вычислимые некоторыми **внешними** Вычислителями.

Термы и атомные формулы нашего языка определяются обычным образом, как это делается в языке исчисления предикатов первого порядка. Класс **Δ0–формул** расширенной сигнатуры σ* – это наименьший класс формул, содержащий все атомные формулы и замкнутый относительно ¬, ∨, ∧, →, ∀x∈a, ∃x∈a, ∀x≤a, ∃x≤a, где «∈» означает «быть элементом списка», а «≤» – «быть начальным подсписком». Множество **∑ – формул** сигнатуры σ* определяется как наименьший класс формул, содержащий все Δ_0 – формулы и замкнутый относительно ∨, ∧, ∀x∈a, ∃x∈a, ∀x≤a, ∃x≤a and ∃x. Отметим, что в нашем языке мы вполне корректно можем использовать выражения вида «∀x=t» и «∃x=t», где t – терм. Очевидно, что на такие выражения можно смотреть как некий эквивалент оператора присваивания «x:=t».

Большую роль в концепции семантического моделирования играет **формульная определимость** между моделями и, в частности, **Δ_0–определимость**. Будем говорить, что модель $\mathfrak{R}$ = (R; **P**), где **P –** это набор предикатов, является **Σ–определимой (Δ_0–определимой) в *HW*(𝔐),** если существуют Σ–подмножество (Δ_0–подмножество) S ⊆ HW(𝔐) и отображение μ: S → R такие, что μ–прообразы предикатов из **P** и =, а также μ–прообразы их дополнений являются Σ–отношениями (Δ_0–отношениями) в HW(𝔐) (обычно с параметрами). Далее будем считать, что отношение является **∑–отношением в *HW*(𝔐)** если оно является ∑–определимым и **Δ_0–отношением в *HW*(𝔐)** если оно Δ_0–определимо. Частично определенная функция является **∑–функцией (Δ_0–функцией) в *HW*(𝔐)** если ее график ∑–определим (Δ_0–определим).

Понятие формульной определимости играет ключевую роль в семантическом моделировании, поскольку позволяет исходным условиям задачи сопоставлять смешанное формульно-термальное, т.е. фактически логико-функциональное определение новой модели с помощью Σ-формул (Δ_0–формул) и Σ-термов (Δ_0–термов). В языке допускается также рекурсивное описание

определяемых предикатов (теорема Ганди). Более точно это формулируется следующим образом. Пусть $\varphi(x_1, \ldots, x_n; Q+)$ есть $\sum$-формула с позитивным вхождением предиката $Q(x_1,\ldots, x_n)$ и пусть предикат Q является n-местным отношением в списочной надстройке $HW(\mathfrak{M})$. Определим оператор G на $HW(\mathfrak{M})$ следующим образом:

$$G(Q) = \{(a_1, \ldots, a_n) | HW(\mathfrak{M}) \models \varphi(a_1, \ldots, a_n; Q+)\}.$$

Тогда оператор G является монотонным и имеет наименьшую $\sum$-определимую неподвижную точку. В [30] было показано, что при определенных условиях теорема Ганди имеет место и для Δ_0–формул.

Практическое использование семантического моделирования показало, что можно ограничиться классом Δ_0-формул и Δ_0-термов, но при этом более активно использовать оракульную вычислимость. Причина заключается в желании, с одной стороны, ограничиться с самого начала эффективным классом вычислений, т.е. вычислениями заданной, например, **полиномиальной сложности,** а, с другой стороны, не потерять выразительную силу языка и его потенциальную полноту. В работе [37], посвященной описанию денотационной и процедурной семантике языка Σ– и Δ_0–определений, была доказан эквивалентность этих семантик, что послужило надежным обоснованием возможности практической его реализации. Что же касается выразительной силы, то заметим, что язык Δ_0–определений допускает мощное расширение **условными** и **рекурсивными** Δ_0 **–термами** [18-20]. Это позволяет рассматривать так обогащенный язык Δ_0–определений как весьма выразительный **логико-функциональ-ный язык моделирования**. И что очень важно – такое обогащение сохраняет исходную полиномиальную вычислимость. Более того, в [30-34] было показано, что имеет место и обратное утверждение – каждая полиномиально вычислимая функция является Δ_0–определимой. Таким образом, можно утверждать, что Δ_0–фрагмент многосортного языка исчисления предикатов первого порядка адекватно соответствует концепции полиномиальной вычислимости или, другими словами, полиномиальная вычислимость является Δ_0–формульно определимой.

Теперь можно подвести некоторые итоги. Построенная логико-математическая теория семантического моделирования предлагает для описания предметной области использовать ее формульно-термальное представление в терминах Δ_0–языка в виде Δ_0–определимой конструктивной модели, при необходимости используя внешние оракулы, и отвечать на Δ_0–запросы, либо проверяя их относительную истинность на этой модели, если речь идет о Δ_0–формулах, либо вычисляя значения Δ_0–термов в этой модели. При этом, как было показано в работах [23-29], такой язык спецификаций можно эффективно обогащать вероятностными конструкциями, что существенно расширяет выразительные возможности языка декларативных спецификаций. Практика показала, что данный логико-вероятностный подход позволяет вполне адекватно представлять и практически **полностью сохранять исходную семантику задачи**. Именно поэтому концепция и носит название **семантического моделирования**.

Заканчивая данную статью, напомним, что технологические, инструментальные и практические аспекты концепции семантического моделирования будут представлены во второй части цикла работ авторов.

Литература

1. *Goncharov S.S., Sviridenko D.I.,* Theoretical aspects of Σ-programming// Lecture Notes in Computer Science. 1986, № 215. P. 169–179.

2. *Ershov Yu. L, S. S. Goncharov S.S., Sviridenko D.I.* Semantic programming, Information processing. // Proc. IFIP 10 th World Comput. Congress. Dublin. 1986., Vol. 10. P. 1113–1120.

3. *Гончаров С.С., Свириденко Д.И.* Математические основы семантического программирования. // Доклады АН СССР. 1986. № 289:6. С.1324-1328.

4. *Ershov Yu. L., Goncharov S.S., Sviridenko D.I.* Semantic foundations of programming. // Lecture Notes in Computer Science. 1987. Vol. 278. P. 116–122.

5. *Goncharov S.S., Sviridenko D.I.* Σ-programming // Transl., II.Ser., Am. Math. Soc. 1989. Vol. 142. P. 101–121.

6. *Витяев Е.Е., Гончаров С.С., Свириденко Д.И.* О задачном подходе в искусственном интеллекте // Сибирский философский журнал. 2019. Т. 17, № 4. С. 5-25.

7. *Витяев Е.Е., Гончаров С.С., Свириденко Д.И.* О задачном подходе в искусственном интеллекте и когнитивных науках // Сибирский философский журнал. 2020. Т. 18, № 2. С. 5-29.

8. *Vityaev E.E., Goncharov S.S., Sviridenko D.I.* Task-driven approach to artificial intelligence. // Cognitive Systems Research. 2023. Vol. 81. P. 50-56.

9. *Vityaev, E.E., Goncharov, S.S., Gumirov, V.S., Mantsivoda, A.V, Nechesov, A.V., Sviridenko, D.I.* Task approach: on the way to trusting artificial intelligence // World Congress: Systems Theory, Algebraic Biology, Artificial Intelligence: Mathematical Foundations and Applications. Selected works. 2023. P. 179-243.

10. *Goncharov S.S, Sviridenko D.I., Vityaev E.E.* Task Approach to Artificial Intelligence. // Proceedings of the Workshop on Applied Mathematics and Fundamental Computer Science 2020, Omsk, Russia, April 23-30, 2020. CEUR Workshop Proceeding. Vol. 2642.

11. *Goncharov S.S., Vityaev E.E., Sviridenko D.I.* Problem-solving approach in artificial intelligence // Applied Mathematics and Fundamental Informatics. 2020. T. 7. No. 2. P. 4-9.

12. *Sviridenko D.I., Vityaev E.E.* A task-based approach to artificial intelligence and its theoretical and technological base. // 18 National conference on artificial intelligence with

international participation (CII-2020), (October 10-16, 2020, Moscow, Russia). Conference proceedings. M: MFTI, 2020. P. 36-44.

13. *Kolmogorov A. N.* Grundbegriffe der Wahrscheinlichkeitrechnung, in Ergebnisse der Mathematik. Berlin, 1933.

14. *Church A.* An Unsolvable Problem of Elementary Number Theory // American Journal of Mathematics. 1936. Vol. 58, №. 58. P. 345–363

15. *Альтшуллер Г. С.* Найти идею: Введение в ТРИЗ – теорию решения изобретательских задач/ М.: Альпина Паблишер. 2010.

16. *Ершов Ю. Л., Самохвалов К. Ф.* О новом подходе к философии математики // Структурный анализ символьных последовательностей. Новосибирск: Изд-во СО РАН. 1984. Вып. 101. С. 141 - 148.

17. *Ершов Ю.Л., Самохвалов К.Ф.* Современная философия математики: недомогания и лечение. – Новосибирск: Параллель, 2007.

18. *Goncharov S.S.* Conditional terms in semantic programming // Siberian Mathematical Journal. 2017. Т. 58. № 5. P. 794-800.

19. *Гончаров С. С., Свириденко Д. И.* Рекурсивные термы в семантическом программировании // СМЖ. 2018. Т. 59, № 6. С. 1279–1290.

20. *Гончаров С. С., Свириденко Д. И.* Логический язык описания полиномиальной вычислимости // Доклады РАН. 2018. Т. 485. № 1. С. 11–14.

21. *Goncharov S., Sviridenko D.* Semantic Modeling and Hybrid Models // 2019 International

Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON), Novosibirsk, Russia, 2019, P. 987-990.

22. *Goncharov S., Ospichev S., Ponomaryov D., Sviridenko D*. The expressiveness of looping

terms in the Semantic Programming. Siberian Electronic Mathematical News. 2020. P. 380-394.

23. *Vityaev E.* The logic of prediction. In: Proceedings of the 9th Asian Logic Conference. World Scientific Publishers. 2006.P.263-276.

24. *Vityaev E, Smerdov S.* On the Problem of Prediction // KONT/KPP. 2007, LNAI 6581. 2011. P. 280-296.

25. *Demin A.V., Vityaev E.E.* Learning in a virtual model of the C. elegans nematode for locomotion and chemotaxis // Biologically Inspired Cognitive Architectures. 2014. Vol.7. H.9–14.

26. *Demin A.V., Vityaev E.E.* Adaptive control of multiped robot // Procedia Computer Science 145C. 2018. P. 629-634.

27. *Demin A.D., Vityaev E.E.* Adaptive Control of Modular Robots // Biologically Inspired Cognitive Architectures (BICA) for Young Scientists, Advances in Intelligent Systems and Computing 636. 2018. P. 204-212.

28. *Vityaev E.E., Perlovsky L.I., Kovalerchuk B.Y., Speransky S.O.* Probabilistic dynamic logic of cognition. // Biologically Inspired Cognitive Architectures. Special issue: Papers from the Fourth Annual Meeting of the BICA Society (BICA 2013). 2013. Vol.6. P. 159-168.

29. *Vityaev, E., Odintsov, S*. How to predict consistently? Trends in Mathematics and Computational Intelligence. // Studies in Computational Intelligence, 796. 2019. P. 35-41.

30. *Goncharov, S. Nechesov, A.* Polynomial Analogue of Gandy's Fixed Point Theorem. // Mathematics. 2021. 9. 2102. Режим доступа: https://www.mdpi.com/2227-7390/9/17/2102

31. *Goncharov S., Nechesov A.* Solution of the Problem P = L. // Mathematics. 2022. 10. 113. Режим доступа: https://www.mdpi.com/2227-7390/10/1/113/notes

32. *Свириденко Д.И.* Задачный подход к построению цифрового двойника организации // Теория систем, алгебраическая биология, искусственный интеллект: математические основы и приложения: Тезисы докладов. М.: Наука. 2023. С.770-773.

33. *Nechesov A.* Functional variant of Polynomial Analogue of Gandy's Fixed Point Theorem.// arXiv. 2024. doi: 10.48550/arXiv.2407.02515

34. Nechesov A. Semantic Programming and Polynomially Computable Representations. // Siberian Advances in Mathematics. 2023. V.33. №1. P.66-85.

35. *Goncharov S., Nechesov A., Sviridenko D.* Programming Methodology in Turing-complete languages. // International Symposium on Knowledge, Ontology, and Theory (KNOTH 24). 2024. P. 345-352.

36. *Ершов Ю.Л.* Определимость и вычислимость. - Новосибирск: Букинист. 2006.

37. *Гончаров С.С., Свириденко Д.И.* Σ–программы и их семантика. // Логические методы в программировании. Вычислительные системы. 1987.Вып. 7. №120. С. 24-51

38. *Gumirov V., Matyukov P., Palchunov D.* Semantic Domain Specific Languages, IEEE. 2018

39. *Palchunov D., Yakhyaeva G., Yasinskaya O.* Software system for the diagnosis of the spine diseases using case-based reasoning. //The Siberian Scientific Medical Journal. 2016. Vol. 36. No. 1. P. 97-104.

40. *Naydanov C, Palchunov D., Sazonova P.* Development of automated methods for the critical condition risk prevention, based on the analysis of the knowledge obtained from patient medical records. // Proceedings of the International Conference on Biomedical Engineering and Computational Technologies (SIBIRCON / SibMedInfo – 2015). 2015. P. 33-38.

41. *Palchunov D., Yakhyaeva G., Dolgusheva E.* Conceptual Methods for Identifying Needs of Mobile Network Subscribers. // Proceedings of the Thirteenth International Conference on Concept Lattices and Their Applications. 2016. P. 147-160.

42. *Naidanov C. A., Palchunov D. E., Sazonova P. A*. Model-theoretic methods of integration of knowledge extracted from medical documents // Vestnik NSU Series: Information Technologies. 2015. Vol. 13. № 3. P. 29–41.

43. *Palchunov D.E., Yakhyaeva G.E.* Fuzzy algebraic systems // Vestnik NGU. Seriya: Matematica, mexanika, informatika. 2010. Vol. 10, №. 3. P. 75-92.

44. *Palchunov D.E., Yakhyaeva G.E.* Fuzzy logics and fuzzy model theory // Algebra and Logic. 2015. Vol. 54. № 1. P. 74-80.

45. *Альтшуллер Г. С., Шапиро Р. Б.* О психологии изобретательского творчества Архивная копия от 18 июля 2019 на Wayback Machine // Вопросы психологии. 1956, № 6. С.37-49.

46. *Соболев П.А.* Как научиться изобретать. Ужгород: Карпати, 1973.

47. *Половинкин А.И.* Межотраслевой фонд эвристических приёмов преобразования объекта // Основы инженерного творчества: Учеб. пособие для студентов втузов. – М: Машиностроение. 1988.

Reference

1. *Goncharov S.S., Sviridenko D.I.* (1986) Theoretical aspects of Σ-programming// Lecture Notes in Computer Science. № 215. P. 169–179.

2. *Ershov Yu. L, S. S. Goncharov S.S., Sviridenko D.I.* (1986) Semantic programming, Information processing. // Proc. IFIP 10 th World Comput. Congress. Dublin. 1986., Vol. 10. P. 1113–1120.

3. *Goncharov S.S., Sviridenko D.I.* (1986) Matematicheskie osnovy semanticheskogo programirovania [Mathematical basis for semantical programing]. In: Doclady AS USSR [Report of AS USSR], № 289:6. P.1324-1328 (In russ).

4. *Ershov Yu. L., Goncharov S.S., Sviridenko D.I.* (1987) Semantic foundations of programming. // Lecture Notes in Computer Science. Vol. 278. P. 116–122.

5. *Goncharov S.S., Sviridenko D.I.* (1989) Σ-programming. // Transl., II.Ser., Am. Math. Soc. Vol. 142. P. 101–121.

6. *Vityaev E.E., Goncharov S.S., Sviridenko D.I.* (2019) O zadachnom podhode v iscusstvennom intelecte [About task-approach to artificial intellegence] In: Sibirskiy filosofskiy zurnal [Siberian philosophical journal] Vol. 17, № 4. P. 5-25. (In russ).

7. *Vityaev E.E., Goncharov S.S., Sviridenko D.I.* (2020) O zadachnom podhode v iscusstvennom intelecte I cognitivnih naukah [About task-approach to artificial intelligence and cognitive science] In: Sibirskiy filosofskiy zurnal [Siberian philosophical T. 18, № 2. C. 5-29. (In russ).

8. *Vityaev E.E., Goncharov S.S., Sviridenko D.I.* (2023) Task-driven approach to artificial intelligence. // Cognitive Systems Research. Vol. 81. P. 50-56.

9. *Vityaev, E.E., Goncharov, S.S., Gumirov, V.S., Mantsivoda, A.V, Nechesov, A.V., Sviridenko, D.I.* (2023) Task approach: on the way to trusting artificial intelligence // World Congress: Systems Theory, Algebraic Biology, Artificial Intelligence: Mathematical Foundations and Applications. Selected works. P. 179-243.

10. *Goncharov S.S, Sviridenko D.I., Vityaev E.E.* (2020) Task Approach to Artificial Intelligence. // Proceedings of the Workshop on Applied Mathematics and Fundamental Computer Science 2020, Omsk, Russia, April 23-30. CEUR Workshop Proceeding. Vol. 2642.

11. *Goncharov S.S., Vityaev E.E., Sviridenko D.I.* (2020) Problem-solving approach in artificial intelligence // Applied Mathematics and Fundamental Informatics. T. 7. No. 2. P. 4-9.

12. *Sviridenko D.I., Vityaev E.E.* (2020) A task-based approach to artificial intelligence and its theoretical and technological base. // 18 National conference on artificial intelligence with an international participation (CII-2020), (October 10-16, 2020, Moscow, Russia). Conference proceedings. M: MFTI. P. 36-44.

13. *Kolmogorov A. N.* (1933) Grundbegriffe der Wahrscheinlichkeitrechnung. In: Ergebnisse der Mathematik. Berlin.

14. *Church A.* (1936) An Unsolvable Problem of Elementary Number Theory // American Journal of Mathematics. Vol. 58, №. 58. P. 345–363

15. *Altshuler G.S.* (2010) Nayti ideu [To find an idea] Moscow. (In russ)

16. *Ershov Yu.L., Samochvalov K.F.* (1984) O novom podhode k filosoofii matematiki [About new approach to philosophy of mathemtics] In: Strukturniy analiz symvolnih posledovatelnostey [Structural analysis of symbolic sequences] Novosibirsk. Vol. 101. P. 141 - 148. (In russ)

17. *Ershov Yu.L., Samochvalov K.F* (2007) Sovremennaya filosofia matematiki: nedomogania I lechenie [Recent philosophy of mathematics: ailments and treatment.] Novosibirsk: Parallel. (In russ)

18. *Goncharov S.S.* (2017) Conditional terms in semantic programming. // Siberian Mathematical Journal. Vol. 58. № 5. P. 794-800.

19. *Goncharov S., Sviridenko D.* (2018) Recursivnie teoremy v semanticheskom programirovanii [Recurcive theoremsin semantic programing] In: Sibirskimatematicheskiy zurnal [Siberian mathematical journal] Vol.. 59. № 6. P. 1279–1290. (In russ)

20. *Goncharov S., Sviridenko D.* (2018) Logicheskiy yazik opisania polynominalnoy vichislimosty [Logical language for describing polynomial computability] In: Doklady AS USSR [Report of AS USSR] Vol. 485. № 1. P. 11–14. (In russ)

21. *Goncharov S., Sviridenko D.* (20019) Semantic Modeling and Hybrid Models // 2019 InternationalMulti-Conference on Engineering, Computer and Information Sciences (SIBIRCON), Novosibirsk, Russia. P. 987-990.

22. *Goncharov S., Ospichev S., Ponomaryov D., Sviridenko D*. (2020) The expressiveness of looping terms in the Semantic Programming. Siberian Electronic Mathematical News. P. 380-394.

23. *Vityaev E.* (2006) The logic of prediction. In: Proceedings of the 9th Asian Logic Conference. World Scientific Publishers. P.263-276.

24. *Vityaev E, Smerdov S.* (2011) On the Problem of Prediction // KONT/KPP. 2007, LNAI 6581. P. 280-296.

25. *Demin A.V., Vityaev E.E.* (2014) Learning in a virtual model of the C. elegans nematode for locomotion and chemotaxis // Biologically Inspired Cognitive Architectures. Vol.7. P.9–14.

26. *Demin A.V., Vityaev E.E.* (2018) Adaptive control of multiped robot // Procedia Computer Science 145C. P. 629-634.

27. *Demin A.D., Vityaev E.E.* (2018) Adaptive Control of Modular Robots // Biologically Inspired Cognitive Architectures (BICA) for Young Scientists, Advances in Intelligent Systems and Computing 636. P. 204-212.

28. *Vityaev E.E., Perlovsky L.I., Kovalerchuk B.Y., Speransky S.O.* (2013) Probabilistic dynamic logic of cognition. // Biologically Inspired Cognitive Architectures. Special issue: Papers from the Fourth Annual Meeting of the BICA Society (BICA 2013). Vol.6. P. 159-168.

29. *Vityaev, E., Odintsov, S*. (2019) How to predict consistently? Trends in Mathematics and Computational Intelligence. // Studies in Computational Intelligence, 796. P. 35-41.

30. *Goncharov, S. Nechesov, A.* (2012) Polynomial Analogue of Gandy's Fixed Point Theorem. // Mathematics. 2021. 9. 2102. Access: https://www.mdpi.com/2227-7390/9/17/2102

31. *Goncharov S., Nechesov A.* (2022) Solution of the Problem P = L. // Mathematics. 10. 113. Access: https://www.mdpi.com/2227-7390/10/1/113/notes

32. *Sviridenko D*. (2023) Zadachniypodhod k postroeniu cifrovogo dvoynika organizacii [A task-based approach to building a digital twin of an organization] In: Teoria system, algebraicheskaya biologia, iskusstvenniy intellect: matematicheskie systemy I prilozenia [Systems theory, algebraic biology, artificial intelligence: mathematical foundations and applications] Moskow: Science. P.770-773. (In russ)

33. *Nechesov A.* (2024) Functional variant of Polynomial Analogue of Gandy's Fixed Point Theorem.// arXiv. doi: 10.48550/arXiv.2407.02515

34. *Nechesov A.* (2023) Semantic Programming and Polynomially Computable Representations. // Siberian Advances in Mathematics. Vol.33. №1. P.66-85.

35. *Goncharov S., Nechesov A., Sviridenko D.* (2024) Programming Methodology in Turing-complete languages. // International Symposium on Knowledge, Ontology, and Theory (KNOTH 24). P. 345-352.

36. *Ершов Ю.Л.* Определимость и вычислимость. - Новосибирск: Букинист. 2006.

37. *Goncharov S., Sviridenko D.* (1987) Σ-programi I ih semantika [Σ-programs and their semantics] In: Logicheskie metody v programirovanii [Logical methods in programing] Vol. 7. №120. P. 24-51. (In russ)

38. *Gumirov V., Matyukov P., Palchunov D.* (2018) Semantic Domain Specific Languages, IEEE.

39. *Palchunov D., Yakhyaeva G., Yasinskaya O.* (2016) Software system for the diagnosis of the spine diseases using case-based reasoning. //The Siberian Scientific Medical Journal. Vol. 36. No. 1. P. 97-104.

40. *Naydanov C, Palchunov D., Sazonova P.* (2015) Development of automated methods for the critical condition risk prevention, based on the analysis of the knowledge obtained from patient medical records. // Proceedings of the International Conference on Biomedical Engineering and Computational Technologies (SIBIRCON / SibMedInfo – 2015). P. 33-38.

41. *Palchunov D., Yakhyaeva G., Dolgusheva E.* (2016) Conceptual Methods for Identifying Needs of Mobile Network Subscribers. // Proceedings of the Thirteenth International Conference on Concept Lattices and Their Applications. P. 147-160.

42. *Naidanov C. A., Palchunov D. E., Sazonova P. A*. (2015) Model-theoretic methods of integration of knowledge extracted from medical documents // Vestnik NSU Series: Information Technologies. Vol. 13. № 3. P. 29–41.

43. *Palchunov D.E., Yakhyaeva G.E.* (2010) Fuzzy algebraic systems // Vestnik NSU: Matematica, mexanika, informatika. 2010. Vol. 10, №. 3. P. 75-92.

44. *Palchunov D.E., Yakhyaeva G.E.* (2015) Fuzzy logics and fuzzy model theory // Algebra and Logic. Vol. 54. № 1. P. 74-80.

45. *Altshuler G.S. Shapiro R.B.* (1956) O psihologii izobretatelskogo tvorchestva [On the psychology of inventive creativity] In: voprosy psihologii [Questions of psychology]. № 6. P.37-49. (In russ)

46. *Sobolev P.A.* (1973) Kak nauchitsa izobretat [How to learn to invent]. Uzhhorod: Karpati. (In russ)

47. *Polovinkin A.I.* (1988) Mezotroslevoy fond evristicheskih priemov preobrazovania obekta [Interdisciplinary fund of heuristic techniques for transforming an object] In: Osnovy inzenernogo tvorchestva: uchebnoe posobie dlia studentov vtuzov [Fundamentals of engineering creativity: Textbook for students of higher education institutions.] Moscow: Mechanical engineering.

Информация об авторах

Гумиров Виталий Шамилович – директор Eyeline СНГ, 630090, Новосибирск, ул. Мусы Джалиля, 3/1), независимый исследователь.

Гумиров Андрей Витальевич – магистрант ММФ НГУ (630090, Новосибирск, ул. Пирогова, 1).

Свириденко Дмитрий Иванович – доктор физ.-мат наук, Институт философии и права СО РАН (630090, Новосибирск, ул. Николаева, 8), Центр искусственного интеллекта Новосибирского национального исследовательского государственного университета (630090, Новосибирск, ул. Пирогова, 1).

dsviridenko47@gmail.com

Information about authors

Gumirov Vitaly Shamilovich – Director of Eyeline CIS, 3/1 Musa Jalil Street, Novosibirsk, 630090), independent researcher.

Gumirov Andrey Vitalyevich – Master's student, Faculty of Mathematics and Mathematics, Novosibirsk State University (1 Pirogova Street, Novosibirsk, 630090).

Sviridenko Dmitry Ivanovich – Doctor of Physical and Mathematical Sciences, Institute of Philosophy and Law, Siberian Branch of the Russian Academy of Sciences (8 Nikolaeva Street, Novosibirsk, 630090), Center for Artificial Intelligence, Novosibirsk National Research State University (1 Pirogova Street, Novosibirsk, 630090).

dsviridenko47@gmail.com

Дата поступления 02.08.2025
Принята к публикации 23.12.2025