НИЦ «КУРЧАТОВСКИЙ ИНСТИТУТ» НИИСИ

Том 15 N 4
ISSN 3033-6422

Труды НИИСИ

SRISA PROCEEDINGS

Москва, 2025

Федеральное государственное автономное учреждение «Федеральный научный центр
Научно-исследовательский институт системных исследований
Национального исследовательского центра «Курчатовский институт»
(НИЦ «Курчатовский институт» — НИИСИ)

ТРУДЫ НИИСИ

SRISA PROCEEDINGS

ТОМ 15 № 4

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ:

ТЕОРЕТИЧЕСКИЕ И ПРИКЛАДНЫЕ АСПЕКТЫ

МОСКВА
2025

Учредитель и издатель

Федеральное государственное автономное учреждение «Федеральный научный центр Научно-исследовательский институт системных исследований Национального исследовательского центра «Курчатовский институт» (НИЦ «Курчатовский институт» — НИИСИ)

«Труды НИИСИ» — это рецензируемый научный журнал, в котором публикуются научные статьи по следующим специальностям и отраслям наук:

- 1.2.1. «Искусственный интеллект и машинное обучение» (физико-математические науки);
- 2.3.1. «Системный анализ, управление и обработка информации, статистика» (физико-математические и технические науки);
- 2.3.2. «Вычислительные системы и их элементы» (технические науки);
- 2.3.3. «Автоматизация и управление технологическими процессами и производствами» (технические науки);
- 2.3.5. «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей» (физико-математические и технические науки).

Направления исследований, по которым журнал публикует оригинальные статьи определены, но не ограничены следующим перечнем: системный анализ; математика; математическое и компьютерное моделирование; задачи автоматизации и управления; обработка сигналов; компьютерное зрение и обработка изображений; распознавание образов; статистика; технологии искусственного интеллекта; информационные технологии; информационная безопасность; вычислительные системы и их элементы; микро- и наноэлектроника; высокопроизводительные вычисления; вопросы численного анализа; нейроморфные и мягкие вычисления; оптико-нейронные технологии; история науки, техники и персоналий. Журнал предназначен для научных сотрудников, инженеров и аспирантов, работающих в указанных направлениях исследований.

Миссия журнала — развитие перечисленных научных направлений в России и за рубежом, включая широкое освещение результатов исследований и обеспечение высококвалифицированных кадров печатными площадями, обеспечение высокого качества исследований путем развития механизма профессионального и общественного обсуждения научных результатов и воспитания молодого поколения ученых-исследователей.

Политика журнала ориентирована на пропаганду передовых научно-технических идей и решений в рамках развития важнейших наукоемких технологий и участия в реализации приоритетов научно-технологического развития Российской Федерации. До 2025 года журнал издавался в печатном виде с названием «Труды НИИСИ РАН» (ISSN 2225-7349). В настоящее время сетевому рецензируемому научному журналу «Труды НИИСИ» присвоен ISSN 3033-6422.

Founder and Publisher

Scientific Research Institute for System Analysis of the National Research Center "Kurchatov Institute" (NRC "Kurchatov Institute" - SRISA)

SRISA Proceedings is a peer-reviewed journal that covers the following key areas of research:

- Artificial intelligence and machine learning
- System analysis, control, information processing, statistics
- Computing systems and their components
- Automation and control in manufacturing
- Mathematical and software support for computing systems, complexes and computer networks .

We publish original articles on topics including, but not limited to: system analysis; mathematics; computer simulation; automation and control; signal processing; computer vision and image processing; pattern recognition; statistics; artificial intelligence; information technologies; cybersecurity; computing

systems and their components; micro- and nanoelectronics; high-performance computing; numerical anal- ysis; neuromorphic and soft computing; optoneural technologies; and the history of science, technology, and researchers. Our readers include researchers, engineers, and doctoral students.

Our mission is to advance these research areas in Russia and worldwide by publishing significant results and offering leading professionals a platform to share their work. We are committed to maintaining high research standards through professional and public review while fostering the next generation of re- searchers.

The journal's policy is to promote advanced research and innovative solutions, foster the develop- ment of high-tech fields, and contribute to key national priorities in science and technology. Until 2025, the journal was published in print as *Trudy NIISI RAN* (Proceedings of SRISA, Russian Academy of Sciences), ISSN 2225-7349. Currently, *SRISA Proceedings* online peer-reviewed journal has been assigned ISSN 3033-6422.

Sergei I. Aryashev, PhD (Engineering), Deputy Director for Microelectronics and Computer Systems, NRC "Kurchatov Institute" - SRISA, Moscow
Alexander N. Godunov, PhD (Phys&Math), Senior Researcher, Head of the Department of System Programming, NRC "Kurchatov Institute" - SRISA, Moscow
Andrei I. Griuntal, PhD (Phys&Math), Head of the Mathematics Department, NRC "Kurchatov Institute" - SRISA, Moscow
Yakov M. Karandashev, PhD (Phys&Math), Leading Researcher, Center for Optic-Neural Technologies, NRC "Kurchatov Institute" - SRISA, Moscow
Anatoly G. Kushnirenko, PhD (Phys&Math), Assoc. Prof., Head of IT for Education Department, NRC "Kurchatov Institute" - SRISA, Moscow
Alexander N. Muranov, PhD (Engineering), Assoc. Prof., Head of the Manufacturing Automation and Control Laboratory, NRC "Kurchatov Institute" - SRISA, Moscow
Konstantin A. Petrov, PhD (Engineering), Senior Researcher, High Performance Microprocessor Architectures Department, NRC "Kurchatov Institute" - SRISA, Moscow
Ilya V. Semenov, PhD (Phys&Math), Leading Researcher, Department of Computational Mathematics, NRC "Kurchatov Institute" - SRISA, Moscow
Andrey I. Tsimbalov, PhD (Engineering), Deputy Director for Microtechnology, NRC "Kurchatov Institute" - SRISA, Moscow

Тематика номера:
Вычислительные системы, их элементы и программное и математическое обеспечение

The topic of the issue:
Computing Systems, Hardware Components and Software

СОДЕРЖАНИЕ

CONTENT

Разработка и внедрение комплекса средств нейросетевой генерации учебного контента, контроля вовлеченности и образовательной аналитики для платформы Мирера

А. И. Белавин[1], Н. В. Гриднев[2], А. С. Караваева[3], К. А. Мащенко[4], Э. А. Орлов[5], Е. Д. Тарасюк[6]

[1]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, aleksei.belavin@math.msu.ru;

[2]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, nikita.gridnev@math.msu.ru;

[3]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, aleksandrakaravaevaa@yandex.ru;

[4]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, МГУ им. М. В. Ломоносова, Москва, Россия, kirill.mashchenko@niisi.ru;

[5]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, eric.orlov@gmail.com;

[6]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, ekaterina.tarasuk@math.msu.ru;

Аннотация. Статья посвящена внедрению в образовательную платформу Мирера системы нейросетевой генерации учебных материалов и тестов на основе ИИ, новых типов заданий для мониторинга просмотра теоретических материалов, а также аналитических инструментов для оценки учебной активности студентов. Рассматриваются архитектура нейросетевых генераторов заданий, механизм распределения баллов за просмотр материала и модуль образовательной аналитики.

Ключевые слова: цифровая образовательная платформа, ЦОП Мирера, нейросетевые модули, ИИ-агенты, аналитика обучения, машинное обучение, искусственный интеллект, LLM, автоматизация образовательных процессов.

1. Введение

Создание качественных учебных материалов и обеспечение эффективного контроля знаний представляют собой ключевые вызовы отечественной цифровой образовательной платформы Мирера [1]. С ростом объема теоретического контента возникла необходимость в автоматизации процессов генерации заданий, мониторинга процесса их изучения и анализа эффективности усвоения. Традиционные подходы требуют значительных временны́х затрат преподавателей на разработку тестов и контроль самостоятельной работы студентов Уменьшение этих затрат особенно актуально при организации массовых курсов по программированию.

В статье рассматривается комплексное решение, включающее три взаимосвязанных нововведения: модуль нейросетевой генерации описаний заданий и тестов, систему отслеживания времени просмотра теоретических материалов, а также аналитические инструменты для оценки учебной активности.

Нейросетевая генерация контента позволяет создавать структурированные описания заданий и разнообразные тестовые форматы (вопросы с выбором числовых и текстовых ответов) на основе промпт-инженерии, а нововведенный тип заданий решает проблему формального прохождения теории, фиксируя реальное время изучения и применяя формулу пропорционального оценивания. Собранный цифровой след студентов, формируемый в процессе обучения, впоследствии подвергается детальному анализу посредством модуля образовательной аналитики. Целью данного анализа является генерация структурированного отчета для преподавателя, содержащего агрегированные и индивидуализированные данные об академических результатах и динамике освоения учебного материала в рамках изучаемого курса.

2. Нейросетевая генерация тестовых заданий и описаний

2.1 Архитектура нейросетевой генерации

В рамках цифровой образовательной

платформы Мирера реализован специализированный модуль, предназначенный для автоматизированной нейросетевой генерации учебных материалов и заданий. Данный инструмент, основанный на применении технологий искусственного интеллекта, призван оптимизировать трудозатраты преподавателей на подготовку дидактического контента и обеспечить соблюдение единых стандартов его качества.

Модуль функционирует на основе языковых моделей, специально адаптированных для решения образовательных задач. Его архитектура включает две ключевые подсистемы: генератор формулировок заданий и генератор тестовых заданий. Обе подсистемы используют методы промпт-инженерии, что позволяет детально управлять процессом генерации и гарантировать соответствие выходного контента установленным критериям платформы [2]. Каждая подсистема решает строго определённую задачу: первая – отвечает за создание однозначных и полных условий заданий, вторая – за формирование вариативных проверочных материалов.

2.2 Генерация описаний заданий

Система предоставляет два методологических подхода к формированию описаний, выбор которых определяется конкретными целями учебного курса.

Первый подход ориентирован на создание детализированного технического описания. Система производит анализ исходного условия, эталонного решения и набора тестов для выявления всех существенных параметров: алгоритмических ограничений, граничных случаев, точных спецификаций форматов входных и выходных данных. Это позволяет трансформировать краткую постановку в исчерпывающее техническое задание. К примеру, для задачи поиска максимального элемента в массиве система может дополнительно определить ограничения на размер данных, диапазон допустимых значений элементов, правила обработки особых случаев (например, пустого массива) [3]. Такой подход минимизирует неоднозначность интерпретаций и позволяет обучающимся сосредоточиться на реализации решения, а не на восстановлении неявно заданных условий.

Второй подход нацелен на генерацию готового к публикации описания с профессиональным структурным и визуальным оформлением. Система формирует оформленный текст в формате HTML с интеграцией LaTeX для отображения математических формул. Итоговый материал включает все стандартные разделы: содержательную постановку, формальную спецификацию, описание входных и выходных данных, примеры работы с подробными комментариями, а также технические ограничения. Данный метод позволяет оперативно получать качественно оформленные материалы без необходимости ручного форматирования, что особенно актуально при разработке новых учебных модулей.

2.3 Генерация тестовых заданий

Система поддерживает создание оценочных материалов в нескольких стандартных форматах, что позволяет осуществлять проверку различных видов познавательной деятельности (рис. 1). Модуль генерации тестовых заданий выгружает из платформы все прикрепленные к занятию материалы, после чего создает на их основе набор тестовых заданий различных типов и выгружает обратно сразу в платформу, без необходимости последующего ручного переноса.

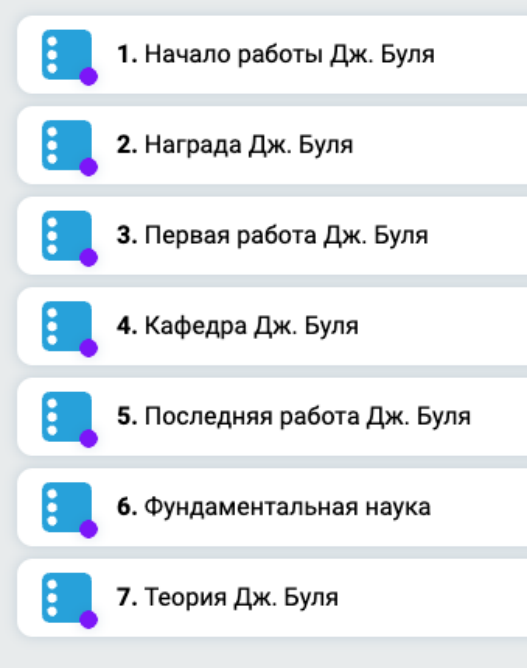

Рис. 1. Задания, сгенерированные нейросетью по теме «Биография и достижения математика Джорджа Буля»

Основные генерируемые форматы включают:

1. Задания с выбором ответа (один или несколько правильных вариантов) (рис. 2). Система формирует постановку вопроса, определяет верный ответ и конструирует так называемые *дистракторы* (правдоподобные ошибочные варианты). Дистракторы строятся на основе типичных ошибок и концептуальных заблуждений, что повышает диагностическую

ценность задания [4].

2. Задания с числовым ответом. Генерируются задания, требующие проведения вычислений (математика, физика, статистика). Система может учитывать допустимую абсолютную или относительную погрешность при верификации ответа, что необходимо для заданий с приближёнными результатами. Например, при оценке вычисленного значения определённого интеграла может быть установлен допустимый диапазон отклонения ±0.01 от эталонного значения.

3. Задания с коротким текстовым ответом (рис. 3). Используются для проверки знания определений, терминов и других коротких и формализуемых ответов.

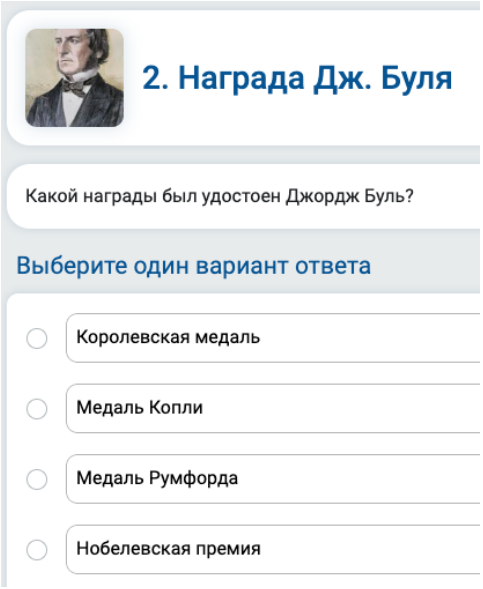

Рис. 2. Пример задания с выбором одного правильного ответа, сгенерированного нейросетью

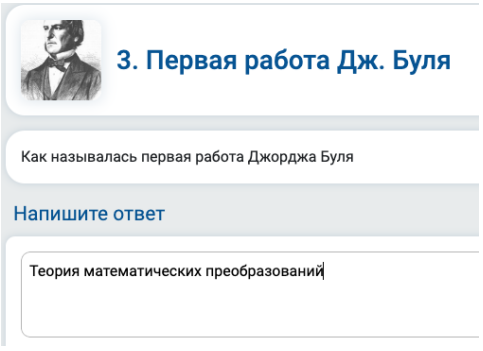

Рис. 3. Пример задания с коротким текстовым ответом, сгенерированного нейросетью

Для каждого формата используются специализированные промпты, которые направляют языковую модель на создание контента строго определённого типа. Например, при генерации заданий с числовым ответом промпт явно указывает, что результатом должно быть вычисляемое значение, и задаёт параметры его проверки, включая точность. При создании вопросов с множественным выбором модель получает инструкции по формированию содержательных дистракторов, которые должны отражать распространённые ошибки в рассуждениях, а не быть заведомо абсурдными.

2.4 Практическая значимость внедрения инструментов генерации

Внедрение автоматизированных средств генерации учебных и оценочных материалов имеет конкретные положительные последствия для образовательного процесса.

Для преподавательского состава ключевое преимущество заключается в существенном сокращении временны́х затрат. Процесс разработки преподавателем качественного задания, включающего точные условия, анализ граничных случаев и подготовку различных форматов проверки, сокращается до нескольких минут.

Для обучающихся польза проявляется в повышении ясности, полноты и структурированности предоставляемых материалов. Детализированные и однозначные описания заданий снижают когнитивную нагрузку, связанную с интерпретацией требований, и позволяют точнее сфокусироваться на содержательной стороне решения. Разнообразие форматов тестовых заданий обеспечивает более объективную и многогранную проверку знаний – от репродуктивного воспроизведения информации до её творческого применения в нестандартных ситуациях [5].

Для образовательного процесса в целом данные инструменты способствуют систематическому накоплению качественного дидактического контента и поддержанию единых стандартов его разработки в масштабах платформы. Благодаря автоматической генерации заданий у преподавателя появляется возможность их массового создания, что значительно повышает количество практических заданий, выполняемых студентами, что в свою очередь повышает эффективность обучения и освоения материалов.

3. Просмотр материалов: мониторинг и оценка

3.1. Просмотр материалов

В последние годы в образовательных курсах широко используются обучающие видео, аудио и

текстовые материалы, которые становятся важным инструментом подачи студентам теоретического материала. В этих условиях важной задачей для преподавателей является повышение вовлечённости студентов в изучение и контроль степени усвоения теоретического материала, поданного в подобной форме. Ранее в ЦОП Мирера возможность прикрепления подобных ресурсов была доступна на страницах контестов – то есть среди материалов, привязанных к занятию «в целом». Однако в целях дозированного предоставления студентам теоретических материалов, с последующей выдачей после очередного блока теории практических заданий, требуется в дополнение к традиционным практическим заданиям включать в контесты новый тип заданий – задания на освоение теоретического материала. А значит преподавателям требуется инструмент, позволяющий разбивать темы на логически завершенные блоки, включающие как теоретические, так и практические составляющие.

Таким инструментом является новый тип заданий – «Просмотр материалов». В этом типе заданий студенту предъявляются только материалы, предоставленные преподавателем для изучения. Подобный формат способствует концентрации внимания на изучаемом содержании, поскольку исключает дополнительные элементы, отвлекающие от усвоения теории. Одновременно он решает проблему формирования отдельных блоков формата «теория + практика», позволяя размещать материалы без привязки к общей теории контеста. Это обеспечивает большую гибкость при конструировании курса и помогает акцентировать внимание студентов на ключевых темах.

В рамках данного подхода реализован метод фиксации времени, затраченного обучающимся на изучение теоретических материалов. Его работа определяется настройкой *«Учитывать время просмотра»*, которая по умолчанию отключена. При активации этой настройки преподаватель задаёт ожидаемое время просмотра материалов (стандартное значение – 30 минут). Подсчёт затраченного времени начинается с момента первого входа обучающегося на страницу соответствующего задания: в этот момент создаётся попытка, внутри которой сохраняется параметр, отражающий накопленное время просмотра. Пока пользователь остаётся на странице, данный параметр автоматически увеличивается через фиксированные интервалы. Полученное значение в дальнейшем используется как для формирования аналитики об активности студентов, так и для начисления баллов за выполнение соответствующего задания.

3.2. Механизм оценивания

Система оценивания в новом типе заданий функционирует в двух режимах, определяемых состоянием настройки *«Учитывать время просмотра»*. При активированной настройке итоговый балл студента рассчитывается по формуле:

$$Score = \min\left\{\begin{array}{c}\text{taskScore},\\ \dfrac{\text{spentTime}}{\text{plannedTime}} \times \text{taskScore}\end{array}\right\},$$

где *taskScore* – максимальный балл за задание, *spentTime* – фактическое время, потраченное обучающимся на странице задания, *plannedTime* – нормативное время, заданное в настройках. В случае, если учёт времени просмотра отключён, пользователю достаточно открыть страницу задания для получения полного балла.

Введение нового типа заданий главным образом направлено на повышение вовлеченности студентов во время изучения теории. Студентов поощряют баллами за потраченное время, что придаёт больший смысл их усилиям [6]. Такая система стимулирует осознанное взаимодействие с теоретическими материалами, минимизируя поверхностный просмотр. В результате, согласно опубликованным данным [7], возрастает качество усвоения знаний, а также укрепляется связь между теорией и практикой внутри курса.

4. Аналитические инструменты и их применение

4.1 Модуль образовательной аналитики

Важным нововведением цифровой образовательной платформы Мирера является модуль образовательной аналитики, предназначенный для поддержки управленческих решений преподавателя и диагностики учебного процесса. В современном понимании образовательная аналитика рассматривается как совокупность методов сбора, анализа и интерпретации данных об обучающихся и образовательных средах с целью улучшения обучения и повышения эффективности образовательных решений [8]. В условиях массового обучения и разнообразия форматов проведения занятий традиционные способы анализа результатов обучения оказываются недостаточно информативными, что обусловило необходимость внедрения визуально-интерпретируемых инструментов анализа

данных, ориентированных на работу с различными образовательными сущностями.

Аналитика по курсам в платформе Мирера представлена в виде двух взаимосвязанных блоков: рейтинговой аналитики и графической визуализации. Такое разделение соответствует современным подходам в области учебной аналитики, предполагающим сочетание количественных показателей и визуальных средств анализа для поддержки интерпретации данных [9]. Это позволяет осуществлять сравнительный анализ показателей обучающихся внутри группы и одновременно получать обобщённое представление о результатах обучения на уровне группы или нескольких групп. Оба блока используют единый набор показателей, что обеспечивает согласованность интерпретации данных при переходе между различными уровнями агрегации.

4.2 Рейтинговый блок

В рейтинговом блоке аналитики данные по каждому студенту визуализируются в виде столбчатых диаграмм, отсортированных от большего значения показателя к меньшему (рис. 4).

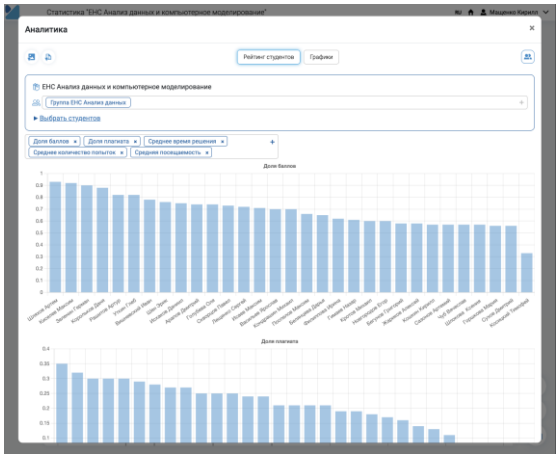

Рис. 4. Рейтинговый блок

Это позволяет преподавателю оперативно оценивать степень неоднородность группы и выявлять студентов с экстремальными значениями показателей. В качестве аналитических параметров используются доля набранных баллов, доля плагиата, время решения заданий, среднее количество попыток при выполнении заданий, а также средняя посещаемость. Показатель доли плагиата рассчитывается как отношение количества заданий, в которых был зафиксирован плагиат (превышение установленного порога сходства), к количеству заданий, проверяемых на плагиат. Данный принцип расчёта применяется как для контестов, так и при агрегации показателей на уровне темы и курса, что обеспечивает сопоставимость результатов между различными сущностями.

4.3 Графический блок

Графический блок аналитики ориентирован на анализ усреднённых показателей и индивидуальных результатов в разрезе выбранной сущности – курса, темы или контеста. Использование визуальных представлений данных позволяет повысить интерпретируемость аналитических результатов и снизить нагрузку на преподавателя при анализе больших массивов образовательных данных [10]. Преподаватель может отображать как средние значения показателей по группе, так и результаты отдельных студентов, что расширяет диагностические возможности анализа. Кроме того, реализована возможность одновременного отображения аналитики по нескольким группам одного курса на едином графике, что позволяет проводить сравнительный анализ образовательных результатов и оценивать влияние методических решений при сопоставимых условиях обучения (рис. 5).

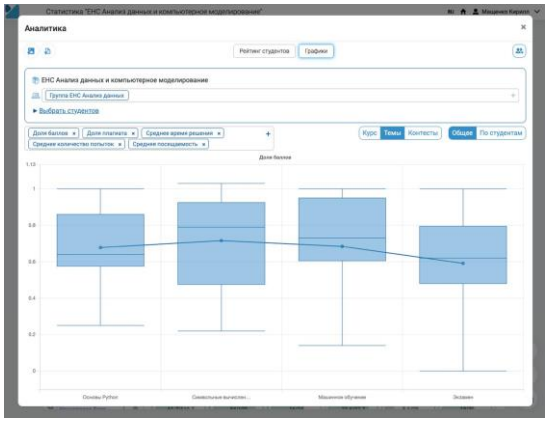

Рис. 5. Графический блок

Дополнительно в аналитическом модуле предусмотрена возможность сравнения данных различных сущностей между собой, а также гибкая фильтрация обучающихся. Для каждой выбранной сущности и группы преподаватель может задать подгруппу студентов, данные которых будут учитываться при построении графиков. Это позволяет, например, анализировать отдельные категории обучающихся, выявлять проблемные подгруппы или оценивать эффективность адресных педагогических вмешательств.

Таким образом, реализованный в ЦОП Мирера модуль образовательной аналитики представляет собой визуальный аналитический

инструмент, обеспечивающий преподавателю доступ к интерпретируемым данным на разных уровнях детализации. Его использование способствует повышению обоснованности управленческих решений и позволяет своевременно выявлять проблемы в учебном процессе, что соответствует современным требованиям к цифровым образовательным платформам и подходам в области образовательной аналитики.

5. Заключение

Разработанный комплекс нововведений для платформы Мирера представляет собой решение, объединяющее возможности искусственного интеллекта для генерации учебных материалов, точный мониторинг учебной активности через систему оценки просмотра теории и мощные аналитические инструменты.

Модуль нейросетевой генерации обеспечивает автоматизированное создание высококачественных описаний заданий и тестов различных форматов, существенно снижая временные затраты преподавателей и повышая единообразие учебного контента.

Внедрение нового типа заданий с контролем времени просмотра теоретических материалов и формулой пропорционального начисления баллов решает проблему формального прохождения теории, обеспечивая оценку реальной вовлеченности студентов.

Аналитическая подсистема предоставляет преподавателям детализированные данные об успеваемости учащихся, способствуя принятию обоснованных решений по совершенствованию курса.

Работа выполнена в рамках темы государственного задания НИЦ «Курчатовский институт» - НИИСИ по теме № FNEF-2024-0001 (1023032100070-3-1.2.1).

Development and implementation of a suite of tools for neural network-based educational content generation, engagement monitoring and learning analytics for the Mirera platform

A. I. Belavin, N. V. Gridnev, A. S. Karavaeva, K. A. Mashchenko, E. A. Orlov, E. D. Tarasuk

Abstract. The article is devoted to the implementation in the Mirera digital educational platform of a neural network–based system for generating learning materials and tests, new types of tasks for monitoring the viewing of theoretical materials, and analytical tools for assessing students' learning activity. The paper considers the architecture of neural task generators, the mechanism for allocating scores for content viewing, and the educational analytics module.

Keywords: digital educational platform, DEP Mirera, neural network modules, AI-agents, learning analytics, machine learning, artificial intelligence, LLM, automation of educational processes

Литература

1. Платформа Мирера [Электронный ресурс] URL: https://mirera.ru (дата обращения:19.12.2025)

2. J. Wei, et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. "Advances in Neural Information Processing Systems", V. 35 (2022).

3. K. Miriyala, M. T. Harandi. Automatic Derivation of Formal Software Specifications from Informal Descriptions. IEEE Transactions on Software Engineering, V. 17 (1991), №10, 1126–1142.

4. T. M. Haladyna, S. M. Downing, M. C. Rodriguez. A review of multiple-choice item-writing guidelines for classroom assessment. "Applied Measurement in Education", V. 15(2002), №3, 309–334.

5. J. Sweller. Cognitive Load During Problem Solving. "Cognitive Science", V. 12(1988) №2, 257–285.

6. M. M. Duisenova, A. N. Zhorabekova. Effects of Rewards on Motivation and Student Achievement in Digital Game-Based Learning in Teaching English as a Foreign Language for Primary School Pupils in Kazakhstan. "Arab World English Journal (AWEJ)", V. 15 (2024) № 3, 125–141. URL: https://awej.org/wp-content/uploads/2024/09/8.pdf (дата обращения: 19.12.2025).

7. M. Romero, E. Barberà. Quality of Learners' Time and Learning Performance Beyond Quantitative Time-on-Task. “International Review of Research in Open and Distance Learning”, V. 12 (2011), № 5, 125–137. URL: https://files.eric.ed.gov/fulltext/EJ963927.pdf (дата обращения: 19.12.2025).

8. Society for Learning Analytics Research. What Is Learning Analytics. URL: https://www.solaresearch.org/about/what-is-learning-analytics/ (дата обращения: 19.12.2025).

9. A. Muslim, M. A. Chatti, M. Guesmi. Open Learning Analytics: A Systematic Literature Review and Future Perspectives. arXiv (2023), URL: https://arxiv.org/abs/2303.12395 (дата обращения: 19.12.2025).

10. M. A. Chatti, A. Muslim, M. Guliani, M. Guesmi. The LAVA Model: Learning Analytics Meets Visual Analytics. arXiv (2023), URL: https://arxiv.org/abs/2303.12392 (дата обращения: 19.12.2025).

Поддержка принятия экспертных решений: ретроспектива и перспективы развития

А. М. Хавторин

НИЦ «Курчатовский институт» — НИИСИ, Москва, Россия, Ahavtorin@mail.ru

Аннотация. Проведен обзор исследований и методик в части формализации экспертного знания и поддержки принятия экспертных решений. Проанализирован подход к решению задачи, разработанный в 70-80-е годы группой под руководством акад. И.М. Гельфанда, в том числе в НИИСИ. Проанализированы сложности и проблемы применения методики. Описаны перспективы переработки и применения методики поддержки принятия экспертных решений в сфере здравоохранения и в других областях экономики.

Ключевые слова: искусственный интеллект, формализация знания, экспертное знание, медицина, медицинская информатика, диагностические игры

1. Введение

Вопросы анализа механизмов принятия решений экспертами в различных предметных областях активно исследовались, начиная с 60-х годов XX века. Именно тогда были разработаны первые модели и алгоритмы, моделирующие процесс мышления человека. Один из наиболее значимых результатов - алгоритм "Кора" М. Бонгарда, моделирующий деятельность человеческого мозга при распознавании образов (60-е гг.) [5].

Слабым местом этих моделей было то, что они требовали высокого уровня формализации предметной области, и проработанной структуры алгоритмов принятия решений. Это значительно ограничивало (хотя и не исключало) использование экспертных систем в слабо формализуемых областях, таких как медицина.

В 70-80е гг под руководством акад. И.М. Гельфанда была проведена работа по анализу структуры и механизмов экспертного знания. Результатом работы стали т.н. «Экспертные игры Гельфанда» - методика формализации экспертного знания. Диагностические игры были апробированы в нескольких областях медицины, главным образом для прогнозирования развития тяжелых состояний пациентов, и показали высокую результативность. Дальнейшему развитию и внедрению методики помешали её громоздкость и недоступность соразмерных вычислительных ресурсов.

В наши дни, в контексте развития технологий искусственного интеллекта и доступности практически безграничных вычислительных ресурсов, диагностические игры Гельфанда могут получить новое развитие. Ограничения систем искусственного интеллекта, с которыми сталкиваются специалисты, могут быть системно обойдены с помощью механизмов анализа и формализации экспертного знания.

На текущем этапе, наиболее перспективной отраслью применения диагностических игр по-прежнему видится медицина.

2. Предпосылки исследований экспертного знания

Начиная с 60-х годов, развитие информационных технологий и кибернетики породило у человечества несколько волн ожиданий, надежд и разочарований. Одно из них - словосочетание «искусственный интеллект» (ИИ) и вера в его грядущее всемогущество. Если отбросить фантастические сценарии, то в обществе сохраняется вера в то, что ИИ:

- Начнёт думать, как человек;
- Сможет принимать технические и управленческие решения за человека;
- Заберет на себя как рутину, так и управление сложными процессами;
- В итоге, коренным образом изменит профессиональный ландшафт цивилизации.

За этими ожиданиями практически был забыт термин «технологии и системы поддержки принятия решений», который наиболее точно описывает место и роль ИИ. Кроме того, всегда остро стоял вопрос корректного обучения ИИ, формирования той базы, на основе которой делаются выводы. В наши дни, с бурным развитием генеративных систем, этот аспект стал особенно актуален.

Далее в статье предлагается опираться на следующие постулаты:

1. ИИ-системы рассматриваются в контексте поддержка принятия

решений.
2. Применение ИИ, как и любой другой технологии, оказывается эффективным только при правильном выборе области применения.
3. Ключевой вопрос результативности применения ИИ – возможность и уровень формализации предметной области.

2.1. Проблемы формализации знания

Одни из первых попыток применения ИИ (экспертных систем) относились к медицине. За основу была принята гипотеза о том, что процесс диагностики заболеваний может быть легко алгоритмизирован в виде дерева вариантов, с минимальной долей неопределенности на выходе. В ряде случаев подобный подход действительно демонстрировал более высокую точность постановки диагноза (или прогнозирования течения заболевания), чем у врача. Однако, алгоритмы оказались в очень высокой степени контекстно-зависимыми.

Результат мог меняться в зависимости от используемых средств диагностики и обследования, подходов конкретного врача и тд. Очевидно, что в основе проблемы лежала неверная попытка формализации как описания объекта исследования (пациента), так и алгоритмов принятия решений. Фактически, не ставилась задача выработки адекватного языка описания возникающих задач, понятного и однозначно трактуемого всеми участниками исследования.

Разработчики экспертных систем, в основном математики и инженеры высокого класса, подходили к медицинским задачам так же, как к физическим или инженерным: путем построения математической модели и поиска оптимального решения. Таким образом, формальное описание задачи и её объекта изначально формировалось в отрыве от постановщиков задачи (врачей), на непонятном им (мета)языке.

Как отмечалось выше, этот подход точечно продемонстрировал хорошие результаты, но системно был ошибочным.

В дальнейшем вопросы формализации были детально проработаны научным коллективом под руководством акад. И.М. Гельфанда.

2.2. Постановка задачи и адекватный язык

Ключевые моменты корректной формализации – постановка задачи и выбор (формирование) адекватного языка описания задачи и объекта исследования.

В вопросы постановки задачи в этой статье углубляться не будем, отметим только, что постановки (формулировки) задачи в одинаковых ситуациях могут радикально различаться в зависимости от подхода исследователя, контекста ситуации, задач высших уровней и тд. Собственно, это то, что в системном анализе называется «точка зрения».

Обозначим только основные критерии правильно поставленной задачи:

1. Должен быть четко и однозначно определён перечень объектов исследования (в приложении к медицине – контингент пациентов).
2. Для каждого объекта (пациента) должен даваться однозначный ответ на поставленный вопрос. Отсутствие ответа в некотором количестве случаев является допустимым.
3. Полученное решение должно поддаваться проверке и быть воспроизводимым.

Адекватный язык описания задачи и решения – ключевой момент формализации. Тема адекватного языка применительно к описанию сложных систем была поднята в [7]. Применительно к задачам медицины, язык можно считать адекватным, если:

1. На нём достаточно просто формулируется как описание задачи(проблемы), так и решения.
2. Язык однозначно понимается постановщиками задачи (врачами) и специалистами – не врачами.
3. Язык позволяет при необходимости легко изменять описание задачи и решения.

В целом, мы придерживаемся гипотезы, что врач организует информацию о больном с помощью небольшого количества показателей. По аналогии со статистическими моделями, можно сказать, что 10% из доступных врачу показателей дают 90% веса.

Действительно, при большом объеме информации, получаемой врачом при обычном осмотре и расспросе больного, не говоря уже о специальных исследованиях, врачи успешно и быстро справляются с обилием информации.

Пример адекватного языка - способ передачи информации от одного врача другому. В беседах друг с другом врачи могут в нескольких фразах сказать все существенное о пациенте. Другими словами, они обладают емким и лаконичным профессиональным языком, позволяющим просто описать ситуацию.

Тема разработки адекватного языка формализации будет детально рассмотрена в готовящейся диссертации. Стоит отдельно отметить, что особое внимание будет уделено

универсализации подхода к языку, позволяющему применять методику в различных областях.

3. Диагностические игры как инструмент формализации знания

В 80-е годы коллективом специалистов-математиков и врачей под руководством акад. И.М. Гельфанда была разработана методика формализации экспертного знания, получившая название «диагностические игры Гельфанда». Первые и наиболее результативные применения методики были в медицине (см. ниже). Также диагностические игры применялись для прогнозирования исходов снежных лавин и оценки продуктивности нефтеносных пластов. [1]

В большинстве случаев методика показала высокие результаты, но широкому применению помешала её громоздкость, в 80-90е годы практически непреодолимая.

3.1. Суть методики

В основе диагностических игр лежат следующие гипотезы, получившие в дальнейшем эмпирическое подтверждение:

1. Знания эксперта носят скрытый характер и проявляются в виде интуитивных решений.
2. Эксперт принимает решение, опираясь на небольшое количество значимых признаков (обычно не более 7).
3. Эксперт не в состоянии выделить и описать эти признаки.
4. Попытка выстроить алгоритм принятия решения по принципу «пусть эксперт детально расскажет, как думает» неэффективна.

Диагностические игры – методика, основанная на моделировании ситуаций, встречающихся в ходе лечебного процесса. Этот подход, с одной стороны, обеспечивает близость деятельности врача в ходе игры к реальной лечебно-диагностической практике, с другой - делает доступным формальный анализ процесса принятия решений.

В ходе диагностической игры ведущий (математик) дает врачу порции информации о пациенте. Из массивов ответов врача можно итерационно вычленить признаки, которым эксперт отводит ключевую роль. Для иллюстрации процесса игры, ниже приведен фрагмент расшифрованной записи. [1][2]

П р и м е р 1.

МАТЕМАТИК (М). В Ваше отделение только что поступил больной инфарктом миокарда (ИМ).

Комментарий. Заранее было оговорено, что речь пойдет о больных с крупноочаговым ИМ, поступивших в клинику не позднее 48 ч от начала острого приступа и не требующих немедленных реанимационных мероприятий. В противном случае вопросы о назначении лечения, прогнозе и т.д. приобретают совсем другой смысл.

ВРАЧ (В). Сколько времени прошло от начала острого приступа?

М. 3 часа.

В. Купированы ли боли?

М. Боли продолжаются.

В. Есть ли сердечная недостаточность (СИ)?

М. Отсутствует.

В. Скажите мне частоту сердечных сокращений (пульс) и артериальное давление.

М. Зачем Вам нужна эта информация, если Вы знаете, что сердечной недостаточности нет?

В. Во-первых, для контроля отсутствия СН, во-вторых, лечение при брадикардии и тахикардии различается, оно различается также при гипертензии и гипотензии.

Комментарии. 1. Недоверие к оценке СН лечащим врачом связано с большой ее субъективностью, особенно в стертых случаях. 2. Дополнительные вопросы, задаваемые математиком, позволяют получать информацию о взаимосвязях между признаками в естественной ситуации, когда врач анализирует состояние конкретного больного. В этом случае он отвлекается от шаблонов и может отметить связи, которые пропустит или сочтет несущественными, отвечая на общий вопрос.

М. Пульс 80—40, давление 120/80- 100/70.

В. Частота дыхания?

М. При поступлении 18, потом возрастает до 24.

В. Есть ли цианоз?

М. Резкий.

В. Имеется диссонанс в сообщаемых данных. Видимо, при поступлении у больного не было СН, а вскоре после этого развились явная СН (резкий цианоз, частота дыхания 24) и нарушения ритма, вследствие чего пульс упал до 40.

Комментарий. Конечно, в этом месте мы столкнулись с несовершенством вопросника, в нем не было полностью отслежено быстрое изменение состояния больного.

М. Вы правы. При поступлении отмечаются частые одиночные и групповые экстрасистолы. Затем на их фоне возникает полная атриовентрикулярная блокада (А-У блокада). Какова Ваша оценка тяжести состояния больного?

В. Больной очень тяжелый.

М. Ваш ближайший прогноз исхода ИМ?

В. Каков возраст больного?

Комментарий. Вопрос о возрасте возник в связи с прогнозом. В следующем примере с относительно благоприятным прогнозом врач этот вопрос не задаст вообще.

М. 47 лет.

В. Прогноз плохой, но не полностью безнадежный из-за относительной молодости больного.

Комментарий. В действительности больной выжил

В ходе разработки методики, были в полном объеме и успешно проведены диагностические игры по следующим направлениям [1][2][6]:

- Прогнозирование исхода мозговых кровоизлияний при разных методах лечения;
- Прогноз исхода инфаркта миокарда;
- Прогнозирование осложнений при разных типах клинического течения инфаркта миокарда;
- Прогнозирование сроков сохранения синусового ритма сердца после устранения мерцательной аритмии;
- Прогнозирование заживления дуоденальных язв;
- Прогнозирование рецидива кровотечения язвы желудка и двенадцатиперстной кишки;
- Дифференциальный диагноз гнойных менингитов различной этиологии у детей первого года жизни.

3.2. Организация диагностической игры

Диагностические игры, как и любой достаточно сложный инструмент, дают эффект, только будучи примененными «по месту» и корректно.

Вопрос выбора точки приложения авторами методики явным образом не рассматривался, поскольку работа изначально велась в тесном сотрудничестве с врачами, и отбирались интуитивно подходящие задачи (*ирония: экспертное знание о том, как выбрать область применения методики, не было формализовано*). В ходе проработки темы в наши дни были выработаны следующие критерии выбора точки приложения методики:

1. Задача/проблема должна возникать регулярно;
2. Должен быть накоплен опыт ее решения;
3. Решение должно требовать привлечения эксперта;
4. Должен существовать дефицит экспертного ресурса.

Таким образом, можно выделить следующие этапы проведения диагностической игры:

1. Определение точки приложения методики;
2. Постановка задачи и предварительная структуризация данных;
3. Составление многоцелевого опросника;
4. Разработка адекватного языка описания объекта (пациента);
5. Разработка и проверка решающих правил.

4. Перспективы применения диагностических игр

Как было отмечено выше, диагностические игры не получили дальнейшего развития главным образом из-за громоздкости самой методики. Современный уровень доступности вычислительных мощностей, и в особенности возможности систем искусственного интеллекта, позволяют оптимистично оценивать возможность «реанимации» и дальнейшего широкого использования методики, причем не ограничиваясь медициной.

Почему сегодня, при высоком уровне развития ИИ-технологий, а особенно генеративных систем, диагностические игры могут найти своё применение? Дело в том, что реальная эффективность систем поддержки принятия решений зависит от наличия, объемов и качества материалов для обучения. Так, ИИ успешно применяется в медицине там, где есть поток обучающих материалов, и качество выводов не хуже, чем у врача. Яркий пример – анализ рентгеновских снимков. С помощью ИИ успешно решаются относительно простые, поточные задачи. Нетиповые задачи, редкие случаи – всё то, что требует экспертного вмешательства, остается в зоне человеческих решений: ИИ просто не на чем обучить.

Аналогично обстоят дела с генеративными системами. Результаты их работы априорно носят вероятностный характер, и зависят в первую очередь от корпуса текстов, на которых обучалась система. Генеративные ИИ, обученные на специализированных текстах (например, оцифрованных историях болезни) существуют, но они не общедоступны.

Таким образом, существующие ИИ-решения не покрывают ту область, где требуется работа эксперта.

С помощью диагностических игр планируется в выбранных областях снизить нагрузку на экспертов (см. выше критерии выбора точки приложения). Непосредственно для проведения игры, формирования сценариев,

моделирования конкретных случаев планируется широко использовать локальные генеративные системы, с дообучением их профильной информацией.

Методику планируется развивать, на первом этапе применяя её в медицине. Именно в этой области её можно достаточно быстро довести до состояния практической готовности, особенно с учетом того, что бОльшая часть специалистов из коллектива И.М. Гельфанда еще живы и открыты к сотрудничеству. Планируется тесное взаимодействие с НМИЦ нейрохирургии им. Н.Н. Бурденко. Предварительно, точки применения методики следующие:

1. Показания к хирургическому/радиологическому лечению/наблюдению при доброкачественных опухолях основания черепа:
   - менингиомы;
   - шванномы;
   - нейрофибромы;
   - остеомы;
   - множественные доброкачественные новообразования.
2. Показания к хирургическому лечению при разных видах нейрохирургической патологии
   - нейроваскулярный конфликт;
   - аномалия Киари;
   - гидроцефалия.

В ходе предварительного анализа, будут выделены 3-4 наиболее характерных задачи из перечисленных выше.

В целом, планируется сделать методику более универсальной, не ограничиваясь медициной. При всей актуальности медицинских задач, схожая ситуация с экспертизой, применением экспертного знания наблюдается во многих областях науки и промышленности. С учетом опыта применения диагностических игр за пределами медицины, можно предположить, что в первую очередь методика найдет применение в геологии (разведка полезных ископаемых, прогноз продуктивности месторождений), эксплуатации сложных инженерных систем, сельском хозяйстве.

Также развитие методики видится исключительно перспективным в свете реализации положений национального проекта «Экономика данных и цифровая экономика» и поддержки развития технологий искусственного интеллекта.

5. Заключение

В настоящей статье описывается подход к вопросам формализации экспертного знания и анализа механизмов принятия решений экспертами, разработанный под руководством акад. И.М. Гельфанда в 70-80 гг. Проанализированы возможности развития и перспективы применения методики в различных областях в наше время.

Публикация выполнена в рамках подготовки кандидатской диссертации по специальности 2.3.1 Системный анализ, управление и обработка информации, статистика.

Support for Expert Decision-Making: Background and Prospects

A. M. Khavtorin

Abstract. We reviewed research and methodologies related to the formalization of expert knowledge and the support of expert decision-making. We examined an approach developed in the 1970s and 1980s by a research group led by I. M. Gelfand, a member of the Academy of Sciences, with the participation of researchers from SRISA. We analyzed the difficulties and limitations associated with applying this methodology. We also discussed prospects for its further development and application to support expert decision-making in healthcare and other sectors of the economy.

Keywords: artificial intelligence, knowledge formalization, expert knowledge, medicine, medical informatics, diagnostic simulation

Литература

1. И. М. Гельфанд, Б.И. Розенфельд, М.А. Шифрин. Очерки о совместной работе математиков и врачей. М.: Едиториал УРССб 2005.

2. И. М. Гельфанд, Б. И Розенфельд, М.А. Шифрин. Психологический журнал, «Диагностические

игры» в задачах медицинской диагностики и прогнозирования», 1985.

3. И. М. Якубович, «Игры математиков и врачей», https://zdrav.fom.ru/post/igry-matematikov-i-vrachey.

4. Ю.Б. Котов, «Методы формализации профессионального знания врача в задачах медицинской диагностики»: диссертация доктора физико-математических наук: 05.13.18. - Москва, 2002.

5. Бонгард М.М. Проблема узнавания М.: Физматгиз, 1967.

6. Гельфанд И.М., Губерман Ш.А., Сыркин А.Л., Головня Л.Д., Извекова М.Л., Алексеевская М.А. Прогнозирование исхода инфаркта миокарда с помощью программы «Кора-3» // Кардиология. – 1977. – Т.17, № 6. – С.19-23

7. Ю. М. Васильёв, И. М. Гельфанд, Ш. А. Губерман, М. Л. Шик. Взаимодействие в биологических системах. М.: Природа, 1969, №06,07.

Support for Expert Decision-Making: Background and Prospects

A. M. Khavtorin

Scientific Research Institute for System Analysis of the National Research Centre Kurchatov Institute, Russian Federation; Ahavtorin@mail.ru

Abstract. We reviewed research and methodologies related to the formalization of expert knowledge and the support of expert decision-making. We examined an approach developed in the 1970s and 1980s by a research group led by I. M. Gelfand, a member of the Academy of Sciences, with the participation of researchers from SRISA. We analyzed the difficulties and limitations associated with applying this methodology. We also discussed prospects for its further development and application to support expert decision-making in healthcare and other sectors of the economy.

Keywords: artificial intelligence, knowledge formalization, expert knowledge, medicine, medical informatics, diagnostic simulation

1. Introduction

Expert decision-making in various areas has been intensively researched since the 1960s. Back then, the first models and procedures simulating human thinking were developed. One of the most significant results is the combinatorial recognition algorithm (CORAL) proposed by M. Bongard in the 1960s. It simulates the human brain's pattern recognition process [5].

The weak point of these models was that they needed a highly formalized definition of the subject area and well-developed decision-making procedures. This significantly limited (though did not exclude) the application of expert systems in poorly formalized fields such as medicine.

In the 1970s and 1980s, I. Gelfand and his team analyzed the structure and mechanisms of expert knowledge. It resulted in the so-called ‘expert games’ method developed by Gelfand for formalizing expert knowledge. Diagnostic simulations have been tested in several areas of medicine, mostly for predicting the development of serious conditions in patients, and have proven to be highly effective. Further development and commercialization were hampered by its cumbersome nature and the unavailability of sufficient computing power.

Nowadays, with the rise of artificial intelligence and virtually unlimited computing resources, Gelfand's diagnostic simulation may get a new boost. The current AI limitations can be systematically circumvented using expert knowledge analysis and formalization.

Medicine is still the most promising field for diagnostic simulations.

2. Background

Since the 1960s, the progress of information technologies and cybernetics has given rise to several waves of human expectations, hopes, and disappointments. One of them is the so-called "artificial intelligence" (AI), allegedly omnipotent. Besides sci-fi scenarios, the society continues to believe that AI will:

– begin to think like a human being
– be able to make engineering and managerial decisions, substituting for humans
– take on both routine tasks and the management of complex processes
– radically change every job profile.

These expectations have overshadowed the concept of decision-making support technology, which most accurately describes the real place and role of AI. Another important issue is training AI with correct information to infer conclusions. This aspect is particularly acute for rapidly growing generative transformers.

In this study, we will use the following assumptions:

1. AI systems are considered decision-making support tools
2. The application of AI, like any other technology, is only effective in appropriately selected areas
3. The key aspect of AI efficiency is the possibility and level of formalization of the subject area.

2.1. Knowledge Formalization

Some of the first AI tools (expert systems) were in medicine. The hypothesis was that medical diagnostics could be easily represented as a tree of options, with a minimum degree of uncertainty.

In some cases, this approach indeed demonstrated greater diagnostic accuracy (or predicting the disease progression) compared to human physicians. However, such systems turned out to be highly context-dependent.

The result varies depending on the diagnostic and examination methods used, the approach of the specific doctor, etc. The root cause is an incorrect attempt at formalization of the patient definition and the decision-making procedures. The researchers failed to develop an appropriate definition language that would be understandable and unambiguously interpreted by all stakeholders.

Developers of expert systems, mainly highly skilled mathematicians and engineers, approached medicine in the same way as engineering: create a simulation model and find the optimal solution. As a result, a formalized definition of the problem and its object was initially developed independently of the problem originators (doctors), and in a (meta) language incomprehensible to them.

As noted above, this approach demonstrated good results in specific cases, but was fundamentally flawed.

Subsequently, a team led by I. Gelfand, I. investigated knowledge formalization in greater detail.

2.2. Problem Definitions and the Need for Appropriate Language

Key points of a correct formalization are the problem definition and the selection (creation) of an adequate problem/research object definition language.

Problem definition is beyond the scope of this study. We only note that definitions of identical problems can be radically different depending on the researcher's approach, current context, higher-level objectives, etc. Actually, this is called a "point of view" in systems analysis.

Let us outline only the key requirements for a correctly defined problem:

1. A clear, unambiguous list of research objects (a cohort of patients in medicine)
2. For each research object (patient), the system should give a clear answer to the question asked. In some cases, it is acceptable not to give any answer at all.
3. The solution must be verifiable and reproducible.

An appropriate problem/solution definition language is a key aspect of formalization. Yu. Vasilyev et al. [7] discussed such a language for the definition of complex systems. In the field of medicine, a language is appropriate if:

1. Problems and solutions are sufficiently easy to define in the language
2. The language is unambiguously understood by both the problem originators (doctors) and non-doctors.
3. The language allows the user to easily edit the problem/solution definition if necessary.

In general, we adhere to the hypothesis that doctors represent a patient profile using a small number of metrics. As in statistical models, we can claim that 10% of the metrics available to the doctor account for 90% of the information.

Indeed, given the large amount of information obtained during routine examinations and interviews with patients, not to mention special tests, doctors can successfully and quickly cope with the abundance of such information.

An example of an appropriate language is the one used to convey information between doctors. Doctors can express all the essential information about a patient in just a few sentences. In other words, they have a concise, succinct professional language suitable for easy definitions.

The development of an appropriate formalization language will be covered in detail in the forthcoming doctoral dissertation. It is worth noting that special attention will be paid to language generalization, making it applicable to various fields.

3. Diagnostic Simulation as a Knowledge Formalization Tool

In the 1980s, a team of mathematicians and doctors led by I. Gelfand developed an approach to expert knowledge formalization called "Gelfand's diagnostic simulation." The first and most effective applications of the approach were in medicine (see below). Diagnostic simulations were also used to predict the outcomes of snow avalanches and assess oil reservoir productivity [1].

In most cases, the approach gave excellent results, but its cumbersome nature was insurmountable in the 1980s and 1990s, preventing its widespread use.

3.1. Approach Essentials

The diagnostic simulation was based on the following hypotheses (subsequently confirmed empirically):

1. An expert’s knowledge is implicit. It manifests itself as intuitive decisions.
2. The expert makes a decision from a small number of significant features (usually no more than 7).
3. The expert is unable to identify and describe these features.
4. Any attempt to build a decision-making procedure on "let the expert explain in detail how they think" is ineffective.

The diagnostic simulation reproduces real-life medical situations. On the one hand, with this approach, the simulation is close to real medical

diagnostic and treatment practice; on the other hand, it makes formal analysis of the decision-making process accessible.

During a diagnostic simulation, the facilitator (mathematician) provides the doctor with bits of information about the patient. From the doctor's responses, we can iteratively extract the features that the expert prioritizes. To illustrate the process, below is an excerpt from a transcription of such a simulation [1][2].

Example 1.

MATHEMATICIAN (M). A patient with myocardial infarction (MI) has just been admitted to your department.

Comment. It was agreed in advance that the patients in the simulation had transmural MIs, were admitted no later than 48 hours after the heart attack, and did not require transfer to the ICU. The purpose was to narrow down the questions about treatment, prognosis, etc.

DOCTOR (D). How much time has passed since the onset of the heart attack?

M. 3 hours.

D. Is the pain relieved?

M. The pain is persisting.

D. Is there cardiac insufficiency?

M. No.

D. Tell me the heart rate and blood pressure.

M. Why do you need this information if you already know there is no cardiac insufficiency?

D. First, to verify that there is no cardiac insufficiency; second, the treatment for bradycardia and tachycardia is different, as does treatment for hypertension and hypotension.

Comments. 1. Distrust of the cardiac insufficiency assessment by the attending physician is associated with its high subjectivity, especially in mild cases. 2. Additional questions asked by the mathematician are intended to gather information about the relationships between the features in a real-life situation, when a doctor analyzes the condition of a specific patient. In this case, the doctor deviates from the routine and may note relationships that they would miss or consider insignificant when answering a general question.

M. Heart rate: 80–40; blood pressure 120/80–100/70.

D. What is the respiratory rate?

M. Upon admission, 18; then increased to 24.

D. Any cyanosis?

M. Severe.

D. There is a discrepancy. Apparently, the patient did not have cardiac insufficiency upon admission, but shortly thereafter developed it (severe cyanosis, respiratory rate: 24) and arrhythmia, causing his heart rate to drop to 40.

Comment. At this point, we encountered some imperfection of the questionnaire: it did not fully track the rapid changes in the patient's condition.

M. You are right. On admission, frequent isolated and couplet premature ventricular contractions (PVCs) were observed. Then it was followed by complete heart block. What is your assessment of the severity of the patient's condition?

D. Extremely severe.

M. What is your estimation of the immediate outcome?

D. How old is the patient?

Comment. The age was needed to make a prognosis. In the following example, where the patient's prognosis was relatively favorable, the doctor did not ask this question at all.

M. 47 years old.

D. The prognosis is poor but not entirely hopeless due to the patient's relatively young age.

Comment. The real patient survived.

We completed diagnostic simulations in the following areas [1, 2, 6]:

- Prediction of the outcome of cerebral hemorrhages after different treatments
- MI prognosis
- Prediction of complications after different types of myocardial infarctions
- Prediction of the duration of sinus rhythm following atrial fibrillation ablation/elimination
- Prediction of the healing of duodenal ulcers
- Prediction of the recurrence of stomach and duodenal ulcer bleeding
- Differential diagnosis of purulent meningitis of various etiologies in infants up to one year old.

3.2. Diagnostic Simulation Procedure

Diagnostic simulations, like any sufficiently complex tool, are only effective when applied appropriately.

The authors of the approach did not explicitly address the selection of its applications, as the research was initially conducted in close collaboration with physicians, who intuitively chose suitable cases (*ironically, the expert knowledge guiding the choice of applications was not formalized*). Today, we investigated this field further and proposed the following criteria for selecting the applications:

1. The problem is recurrent.
2. The history of successful cases is available
3. The solution requires the involvement of an expert
4. There is a shortage of experts.

A diagnostic simulation is divided into stages as follows:

1. Selection of the specific area

2. Problem definition, preliminary data structuring
3. Compilation of a multi-purpose questionnaire
4. Creation of an appropriate language for the research object (patient) definition
5. Development and verification of the decision-making rules.

4. Prospects for Diagnostic Simulations

As noted above, diagnostic simulations were not further developed, largely due to their intrinsically cumbersome nature. With currently available computing power and artificial intelligence, there is reason for optimism about the revival and widespread adoption of this approach, extending beyond medicine.

Why, given the maturity of AI technologies, especially generative transformers, can diagnostic simulations find their applications today? The real effectiveness of decision-making support systems depends on the availability, size, and quality of the training dataset. For instance, AI is successfully applied in medicine if there is a steady flow of training content. In this case, the quality of diagnostics is no worse than that of a human doctor. A striking example is the analysis of X-ray images. AI successfully solves relatively simple, routine tasks. Unconventional tasks and rare cases require human expert intervention: there is just no training dataset for the AI system.

It is similar for generative transformers. Their outputs are probabilistic by design and depend primarily on the training datasets. There are generative AI systems trained on specialized texts (e.g., digitized medical records) exists, but they are not publicly available.

Current AI solutions do not cover expert knowledge.

Diagnostic simulations can reduce expert workload in some areas (see above about the selection of applications). Local generative transformers augmented with specialized information can be widely used in such simulations to generate scenarios and create specific cases.

The approach will be further developed with a focus on its medical applications. In this area, it can mature fast, since most of the Gelfand team members are still alive and ready to collaborate. We will partner with Burdenko's Research Institute of Neurosurgery. The preliminary list of applications is:

1. Indications for surgical/radiological treatment/observation for benign tumors of the skull base:
   - meningiomas
   - schwannomas
   - neurofibromas
   - osteomas
   - multiple benign neoplasms.
2. Indications for various types of neurosurgical treatment
   - neurovascular conflicts
   - Chiari malformations
   - hydrocephalus.

The preliminary analysis will identify the 3-4 most prominent applications from the above list.

The approach will be made more universal, not limited to medicine. A situation with expert knowledge similar to that in medicine also exists in many other areas of science and technology. Based on experience with diagnostic simulations outside medicine, the approach can be used in geology (exploration, reservoir productivity assessment), the operation of complex engineering systems, and agriculture.

The approach is also well aligned with the Data-Driven and Digital Economy national initiative and artificial intelligence technologies.

5. Conclusion

This study presents an approach to the formalization of expert knowledge and the analysis of expert decision-making developed under the guidance of I. Gelfand in the 1970s and 1980s. We analyzed the options for further development and various applications of this approach.

This paper forms part of a forthcoming doctoral dissertation in systems analysis, information management and processing, and statistics.

References

1. I.M. Gelfand, B.I. Rozenfeld, M.A. Shifrin Essays on the Collaboration between Mathematicians and Physicians. Moscow: Editorial URSS 2005 (in Russ.)

2. I.M. Gelfand, B.I. Rozenfeld, M.A. Shifrin "Diagnostic Simulations" for Medical Diagnostics and Prognosis. Journal of Psychology, 1985 (in Russ.)

3. M. Yakubovich Games Played by Mathematicians and Physicians, https://zdrav.fom.ru/post/igry-matematikov-i-vrachey (in Russ.)

4. Yu.B. Kotov Formalization of a Physician's Professional Knowledge for Medical Diagnostics: Doctorate dissertation. Moscow, 2002 (in Russ.).

5. Bongard M.M. The problem of Recognition Moscow.: Fizmatgiz, 1967 (in Russ.).

6. Gelfand I.M., Guberman Sh.A., Syrkin A.L., Golovnia L.D., Izvekova M.L., Alekseevskaia M.A. Kora-3 Software for Predicting the Outcome of Myocardial Infarctions // Cardiology. 1977. Vol. 17, No. 6, pp.19-23

7. Yu. M. Vasilyev, I. M. Gelfand, Sh. A. Guberman, M. L. Shik. Interaction in Biological Systems. Moscow: Priroda, 1969, No. 06, 07 (in Russ.)

Цифровая трансформация раннего инженерного образования: опыт массовой подготовки педагогических кадров в Республике Татарстан

Р. Ф. Шайхелисламов[1], Л. Ф. Осипова[2], М. Ф. Аблаев[3]

[1]Высшая школа педагогического мастерства ФГАОУ ВО «Казанский (Приволжский) федеральный университет», Казань, Российская Федерация, Rais.Shajhelislamov@kpfu.ru;

[2]Центр непрерывного повышения профессионального мастерства педагогических работников Высшей школы педагогического мастерства ФГАОУ ВО «Казанский (Приволжский) федеральный университет», Казань, Российская Федерация, LFOsipova@kpfu.ru;

[3]Высшая школа педагогического мастерства ФГАОУ ВО «Казанский (Приволжский) федеральный университет», Казань, Российская Федерация, Marat.Ablayev@kpfu.ru.

Аннотация. В статье представлен опыт Республики Татарстан по реализации масштабного проекта непрерывного повышения квалификации педагогических работников в области преподавания основ алгоритмики, программирования и инженерии детям дошкольного и младшего школьного возраста. Проект, реализованный в 2025 году Центром непрерывного повышения профессионального мастерства педагогических работников КФУ, охватил 2061 педагога из всех муниципальных образований республики. В качестве ключевого образовательного инструмента использовалась российская предметно-цифровая образовательная среда «ПиктоМир» разработки НИЦ «Курчатовский институт»-НИИСИ, адаптированная под задачи формирования инженерного мышления с ранних лет. Описаны организационная модель, содержательные аспекты программ для воспитателей дошкольных образовательных организаций и учителей начальных классов, а также результаты, подтверждающие эффективность выбранного подхода для построения преемственной вертикали «детский сад – начальная школа».

Ключевые слова: алгоритмическое мышление, инженерное образование, цифровая образовательная среда «ПиктоМир», повышение квалификации педагогов, дошкольное образование, начальная школа, Республика Татарстан, непрерывность образования

1. Введение: вызов времени и стратегический ответ

Формирование основ цифровой грамотности и инженерного мышления с самого раннего возраста – это требование современного мира. Изучение алгоритмики и программирования закономерно «спустилось» из университетов в школы, из школ в детские сады, став новой грамотностью. Это отмечается и в работах ведущих российских специалистов [1, 2]. Республика Татарстан, являясь одним из лидеров в сфере цифровизации и образования, ответила на этот вызов комплексным проектом массовой подготовки педагогических кадров, способных реализовывать пропедевтические курсы по алгоритмике и инженерии.

В отличие от известного опыта г. Сургута, где фокус был сделан на непосредственное обучение дошкольников с помощью среды «ПиктоМир» [3], татарстанский проект сосредоточился на системной переподготовке уже работающих педагогов, создавая кадровый фундамент для повсеместного внедрения современных образовательных практик.

2. Методологическая основа и образовательный инструмент

Теоретической и практической основой проекта стала российская разработка – бестекстовая цифровая образовательная среда (ЦОС) «ПиктоМир» [4]. Разработка среды была начата в 2010 году в ФГУ ФНЦ НИИСИ РАН, а сегодня продолжается в НИЦ «Курчатовский институт»-НИИСИ. Среда доказала свою эффективность в ходе массового внедрения [5],

полностью соответствует отечественным методикам работы с детьми 5–10 лет. В среде используются:

- Бесписьменный интерфейс: ориентация на работу с пиктограммами позволяет дошкольникам и младшим школьникам преодолеть барьер неумения читать и писать.
- Геймификация: учебный процесс построен как прохождение уровней компьютерной игры, что обеспечивает высокую мотивацию.
- Дидактическая защита от ошибок: жёсткие шаблоны программ и различная форма пиктограмм делают синтаксические ошибки невозможными, позволяя ребёнку сосредоточиться на логике решения поставленных задач.
- Преемственность: среда «ПиктоМир» и её гибридная версия «ПиктоМир-К» позволяют плавно переходить от пиктограммного к текстовому программированию.

Именно на этой, проверенной массовой практикой, основе были разработаны две адресные программы повышения квалификации.

3. Организационная модель проекта: гибкость и массовость

В рамках проекта реализовались лучшие наработки программ повышения квалификации, четкая организационная структура и гибкое использование форматов обучения, что позволило обеспечить максимальный охват. Для воспитателей ДОО (1074 человека): Смешанный формат «2+3»: 2 дня дистанционного изучения теоретических основ на платформе КФУ и 3 дня очного интенсивного практикума в Казани. Очный формат включал работу с материальными носителями (кубики, карточки), освоение интерфейса «ПиктоМира» и методику проведения занятий. Для учителей начальных классов (987 человек): Дистанционный формат с элементами онлайн-сопровождения. Педагоги самостоятельно изучали материалы на платформе КФУ, а ключевые практические модули (робототехника, 3D-моделирование, углубленная работа с ЦОС) осваивали на вебинарах с экспертами из НИЦ «Курчатовский институт» – НИИСИ (г. Москва), преподавателями КФУ и практикующим учителем-инноватором из Казани.

Такая модель позволила эффективно обучить педагогов из самых отдалённых районов республики, минимизировав их отрыв от работы.

4. География и масштаб

Программа внедряется в рамках всей республики, с уклоном на Казанскую агломерацию в дошкольном сегменте и более равномерное распределение по районам республики среди учителей начальных классов.

- Воспитатели ДОО (1074 чел.): Обучение прошли педагоги из 42 муниципалитетов. (Рис.1). Наибольшая активность была зафиксирована в г. Казани (774 чел., 72 % от общего числа), что позволило создать мощный методический кластер в столице республики. При этом значительное число воспитателей обучилось в крупных сельских районах (Лаишевский – 86, Пестречинский – 69).

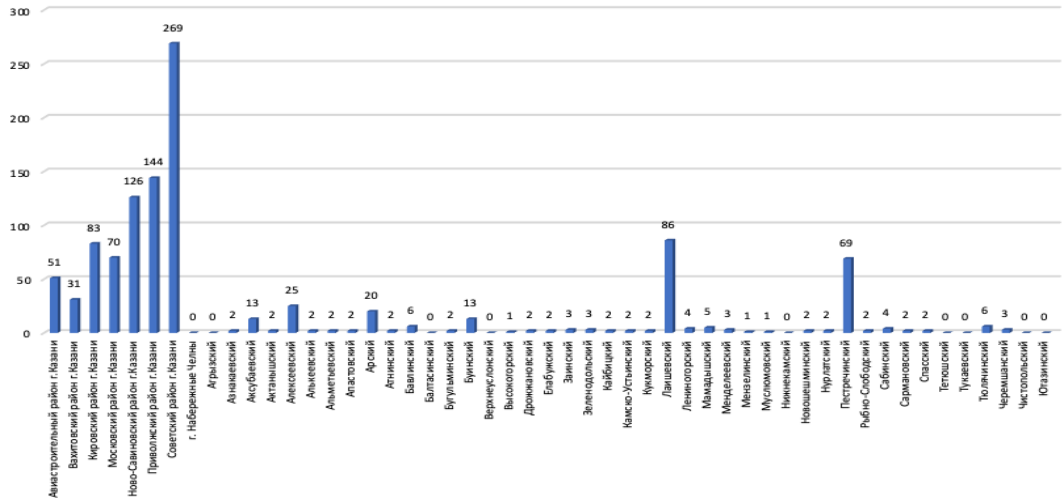

Рис. 1. Распределение обученных воспитателей ДОО по муниципальным образованиям Республики Татарстан

- Учителя начальных классов (987 чел.): Проект охватил все 51 муниципальное образование РТ, включая г. Набережные Челны и крупные промышленные центры, где по дошкольному направлению активность была ниже (Рис. 2). Лидерами стали Альметьевский (60 чел.), Нижнекамский (58 чел.) и Бугульминский (42 чел.) районы. Удельный вес Казани составил 12 %, что свидетельствует о целенаправленном распространении компетенций в регионы.

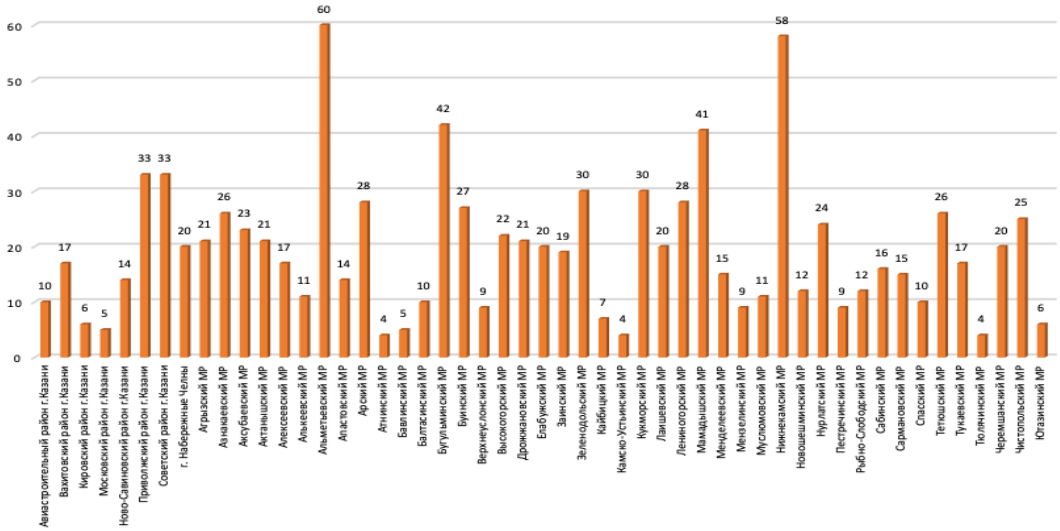

Рис. 2. Распределение обученных учителей начальных классов по муниципальным образованиям Республики Татарстан

Суммарно в проекте приняли участие 2061 педагогов, что создаёт критическую массу специалистов, готовых к внедрению нового содержания.

5. Содержательное ядро: две программы – одна цель

Программы обучения, разработанные для двух категорий педагогов, были выстроены в единую логическую вертикаль с учётом возрастных особенностей обучающихся.

Для воспитателей ДОО акцент делался на формировании представлений об алгоритмах и компьютерных программах через предметные игры и манипуляции с кубиками; освоении пиктограммного интерфейса «ПиктоМира» и методике проведения занятий в игровой форме; понимании психолого-педагогических особенностей детей 5–7 лет и способах поддержки детской инициативы.

Для учителей начальных классов программа была расширена и углублена: углублённое изучение основ алгоритмизации и программирования; работа с более сложной, гибридной средой «ПиктоМир-К», обеспечивающей переход к текстовому программированию; модули по прикладной инженерии: основы робототехники, 3D-моделирования и 3D-печати; методика интеграции алгоритмики в учебные предметы (математика, окружающий мир) и проектная деятельность.

Итогом обучения для всех педагогов стала разработка собственного учебного проекта (сценария занятия или урока), что гарантировало практико-ориентированность результатов.

6. Заключение: построение непрерывной образовательной траектории

Опыт Республики Татарстан 2025 года демонстрирует успешную модель региональной образовательной политики, нацеленной на опережающую подготовку кадров для цифровой трансформации начальных ступеней обучения.

Ключевые достижения проекта:

1. Массовость и системность: подготовлено более 2 тысяч педагогов, готовых внедрять полученный опыт на практике.

2. Преемственность: Создана содержательная и методическая связь между программами для ДОО и начальной школы, что закладывает основу для непрерывного развития алгоритмического и инженерного мышления у ребёнка с 5 до 11 лет.

3. Адаптивность: Гибкая организация обучения (очно-дистанционная) позволила охватить всю территорию республики с учетом специфики каждой категории педагогов.

4. Опора на отечественную технологию, сформулированную в статье академика В.Б. Бетелина с соавторами [6]: широкая апробация и методическая адаптация среды «ПиктоМир» подтвердили её эффективность как стержневого инструмента российской цифровой педагогики для раннего возраста.

Таким образом, Республика Татарстан на практике реализует стратегию, согласно которой формирование основ алгоритмического мышления надо проводить системно, последовательно и массово, начиная с дошкольной ступени. Проект направлен на формирование будущих инженерно-технических кадров, обладающих критическим мышлением, логикой и творческим подходом к решению задач уже с первых лет обучения.

Digital Transformation of Early Engineering Education: Experience of Large-Scale Teacher Training in the Republic of Tatarstan

R. F. Shajhelislamov, L. F. Osipova, M. F. Ablayev

Abstract. This article presents the experience of implementing a large-scale teacher training project aimed at developing algorithmic and engineering thinking in preschool and primary school children.

Keywords: early engineering education, algorithmic thinking, PiktoMir, teacher training

Литература

1. Кушниренко, А. Г. Годовой цикл занятий «Алгоритмика для дошкольников» в подготовительных группах дошкольных образовательных учреждений / А. Г. Кушниренко, А. Г. Леонов, И. Н. Грибанова, М В. Райко // Вестник кибернетики. – 2018. – № 2(30). – С. 138–144.

2. Леонов, А. Г. Подходы к преподаванию основ программирования от вуза до детского сада / А. Г. Леонов. – Москва : Наука, 2024. – 139 с.

3. Бесшапошников, Н. О. Цифровая образовательная среда «ПиктоМир»: опыт разработки и массового внедрения годового курса программирования для дошкольников / Н. О. Бесшапошников, А. Г. Кушниренко, А. Г. Леонов, М. В. Райко, О. В. Собакинских // Информатика и образование. 2020. – № 10.

4. Кушниренко, А. Г. Предметно-цифровая среда «ПиктоМир» как инструмент формирования алгоритмического мышления и психоэмоционального развития детей / А. Г. Кушниренко, А. Г. Леонов, М. В. Райко, У. М. Солопова // Педагогические исследования. – Астрахань, 2025. – № 3 (23). – С. 206–232.

5. Леонов, А. Г. Результаты освоения годовой программы «Алгоритмика для дошколят» подготовительными группами муниципального ДОУ / А. Г. Леонов, М. В. Райко, О. В. Собакинских, Н. В. Собянина. – 2020.

6. Бетелин, В. Б. Основные понятия программирования в изложении для дошкольников / В. Б. Бетелин, А. Г. Кушниренко, А. Г. Леонов // Информатика и её применения. – 2020. – Т. 14, вып. 3. – С. 55–61.

Оптимизация ввода-вывода с помощью кеширующих блочных устройств в среде GNU/Linux

А. Г. Прилипко[1], С. Г. Романюк[2], Д. В. Самборский[3]

[1]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, aleksey.prilipko@gmail.com;

[2]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, sgrom@niisi.ras.ru;

[3]НИЦ «Курчатовский институт» - НИИСИ, Москва, Россия, samborsky_d@fastmail.com

Аннотация. Современные устройства хранения данных имеют широкий диапазон основных характеристик: объёма, пропускной способности, и скорости записи. В силу физических ограничений в одном устройстве невозможно достичь максимальных значений всех этих характеристик. Тем не менее, совместное использование разных типов накопителей часто позволяет оптимизировать производительность операционной системы для решения прикладных задач. Ядро ОС GNU/Linux позволяет создавать композитные устройства блочного ввода-вывода, такие как программные массивы накопителей данных (RAID-устройства) и кеширующие программные устройства. В данной статье выполнен анализ системных средств и ядра ОС GNU/Linux с целью поиска стратегии повышения производительности аппаратно-программной конфигурации подсистемы ввода-вывода.

Ключевые слова: ввод-вывод данных, Linux, RAID, dm-cache, dm-writecache

1. Введение

Выбор конфигурации системы хранение данных для вычислительных узлов серверного кластера в последние годы осложняется разнообразием устройств и дороговизной наиболее ёмких и производительных моделей SSD-накопителей. Вместе с тем, морально устаревшие механические HDD-диски также продолжают использоваться, когда требуются хранение и обработка больших массивов информации.

Операционная система GNU/Linux имеет сложную подсистему виртуальных блочных устройств, которая обеспечивает создание и работу программных RAID-массивов, кеширующих устройств, средств контроля целостности данных, и других виртуальных устройств, расширяющих функции системы Linux для работы с накопителями данных. Данная работа рассматривает задачу повышения скорости работы прикладных задач с устройствами хранения данных системными средствами ОС GNU/Linux. В статье рассмотрены три конфигурации использования разнородных накопителей данных и выполнен сравнительный анализ этих конфигураций.

2. Асимметричное RAID1-устройство

В современных версиях ядра Linux есть опции для тонкой настройки алгоритма репликации данных в RAID1-массивах [1]. При наличии условно быстрого устройства (с технологией ячеек памяти Z–NAND, XL– FLASH, или 3D XPoint) и условно медленного SSD- или HDD-устройства можно попробовать создать такое RAID1-устройство, в котором запись на медленное устройство откладывается и не замедляет работу прикладной программы. Это достигается с помощью опции write-mostly, указывающей на медленное устройство, и опции write-behind, задающей объем данных, запись которых на это устройство допускается задержать. Увеличение объема отложенных для записи данных вызывает эффект сглаживания периодов интенсивной нагрузки на подсистему ввода-вывода со стороны прикладных программ.

Опция write-mostly обозначает медленное устройство и подсказывает, что чтения данных лучше выполнять с другого, более быстрого устройства. Следует отметить, что действие опции write-behind также распространяется и на операции принудительного завершения операций записи (системные вызовы sync и datasync).

Другими словами, опция write-behind нарушает контракт, согласно которому работает

программный RAID1-массив, поскольку при её использовании в течение некоторого периода времени теряется гарантия наличия двух экземпляров записанных данных. При выходе из строя быстрого диска данные будут потеряны, что может привести к некорректному состоянию прикладных программ. В случае внезапного выключения сервера сразу после перезагрузки на этом RAID1-диске будет запущена процедура восстановления данных. Данная процедура использует битовую карту неполных блоков и для каждого такого блока сделает запись в правильном направлении — запишет данные на медленный диск, поскольку опция write-mostly заставляет алгоритм драйвера RAID1-массива избегать чтение данных с медленного диска. По этой же причине прикладные программы прочитают корректный блок данных, даже если эта операция чтения опередит процесс восстановления массива и прочитает блок, у которого еще различаются копии данных.

Таким образом, если имеется два быстрых и два медленных устройства, то есть два способа построения RAID -массивов: (1) - симметричные RAID1-массивы из быстрых и медленных дисков, и (2) - два асимметричных RAID1-массива. Во втором способе будет в 2 раза больше объем данных, доступ к которым будет осуществляться преимущественно через быстрый накопитель.

Суммируя, можно перечислить следующие достоинства и недостатки асимметричных RAID1-массивов:

- *Достоинства:* Простота конфигурации. Высокая скорость чтения; Высокая скорость записи, если объем записанных данных в течении короткого промежутка времени не превышает значения write-behind.
- *Недостатки:* Риск потери данных в случае выхода из строя быстрого устройства; Неполное использование объема «медленного» диска; Повышение нагрузки на быстрое устройство операциями чтения данных.

3. Универсальное кеширующее устройство (dm-cache)

Для кеширования чтения и записи данных на медленных накопителях в ОС GNU/Linux существует драйвер dm-cache [2]. Этот драйвер позволяет использовать быстрый накопитель в качестве кеш-устройства для ускорения операций ввода-вывода на медленном устройстве, базовом устройстве. Для обеспечения надежности хранения данных в качестве базового и кеш-устройства используются RAID-массивы.

Кеширование данных драйвером dm-cache может выполняться в двух основных режимах: writeback и writethrough. По умолчанию используется режим отложенной записи (writeback), в котором операция записи считается завершенной, когда данные записаны либо на обоих устройствах, либо только на кеш-устройстве. В последнем случае блок данных кеш-устройства помечается как несинхронизированный и его запись на базовое устройство откладывается на неопределенное время. Операции принудительной записи данных (системные вызовы sync и datasync) не влияют на алгоритм кеширования и не требуют ожидания записи на базовое устройство, поэтому прикладные программы, требующие синхронного ввода-вывода, также видят повышение скорости записи. В режиме сквозной записи (writethrough) данные записываются сразу на оба устройства, и только после этого операция считается выполненной. Такой режим ускоряет операции чтения недавно прочтённых или записанных данных.

В обоих режимах операции чтения блоков данных с базового устройства вызывают запись этих блоков в кеш-устройство для их возможного повторного использования. Если основное назначение кеш-устройства состоит в ускорении записи, то кеширования прочтённых данных следует избегать. До версии ядра Linux v.4.6 для этого было достаточно использовать настройки read-promote-adjustment и write-promote-adjustment, которые определяли баланс кеширования чтения и записи. Однако, начиная с версии v4.6 единственной политикой кеширования в драйвере dm-cache стала новая политика кеширования smq (stochastic multi-queue), в которой используется стохастический алгоритм, не имеющий никаких пользовательских настроек [4].

Таким образом, универсальное устройство кеширования дает наибольший эффект, когда кеширование чтения тоже необходимо – либо, когда быстрый накопитель имеет достаточно большой объём.

4. Устройство кеширования записи (dm-writecache)

Для кеширования только операций записи данных в ядро ОС Linux начиная с версии

v.4.18 был добавлен драйвер dm-writecache [3]. Основными мотивациями написания этого драйвера были сложность настройки универсального драйвера dm-cache и тот факт, что с удешевлением оперативной памяти задача кеширования чтения данных стала хорошо решаться общесистемным файловым кешем (page cache, [5]). Драйвер dm-writecache также способен использовать накопители в формате модулей памяти DIMM, называемых NVDIMM (non-volatile DIMM) или Pmem (persistent memory), которые в силу интерфейса шины DIMM имеют наименьшую задержку при передачи данных.

При создании данного кеширующего устройства допустимо указать параметры алгоритма работы кеша [3], которые влияют на интенсивность синхронизации данных кеш-устройства с базовым устройством. Параметры по умолчанию обеспечивают режим приоритетного обслуживания операций записи: алгоритм старается как можно быстрее сбросить кешированные данные на базовый диск, чтобы кеш был готов к будущим всплескам операций записи. Если после пиковой нагрузки операциями записи наблюдается замедление операций чтения, то следует уменьшить значения параметров writeback_jobs и pause_writeback.

Когда необходимо одновременно решать задачи хранения данных большого объёма и обеспечения высоких скоростей синхронной записи, то оказывается оправдано применение трехуровневого устройства: кеш записи dm-writecache подключен к кеш-диску dm-cache, работающему в режиме writethrough и подключенному к медленным устройствам, таким как RAID10-массив состоящих их механических дисковых накопителей (HDD). Например, сервер виртуализации требует хранения большого объема данных для виртуальных ОС, и, хотя большая часть этих данных является «холодными» и редко модифицируется, внутри виртуальных ОС встречаются приложения, дающие высокую нагрузку записи. В таком случае трудно разделить «горячие» и «холодные» данные по разным системам хранения, и требуется универсальное скоростное и объёмное хранилище. Если старт или возобновление работы некоторых прикладных систем замедляется чтением данных с самого медленного накопителя, то предварительный или периодический «прогрев» общесистемного файлового кеша и кеша чтения могут выполнить утилиты vmtouch [7] и vmprobe [8]. Ядро Linux также имеет системный вызов mincore, который сообщает список находящихся в файловом кеше блоков указанного файла. Применение этого системного вызова позволяет построить карту часто используемых данных и удерживать эти данные в кеше чтения внешним инструментом, без модификации прикладных систем или виртуальных ОС [6].

5. Использование RAM-диска

Предварительную оценку максимального теоретического ускорения операций ввода-вывода от применения кеширующего накопителя данных можно сделать с помощью виртуального блочного устройства, хранящего данные в оперативной памяти (RAM-диск). Такое устройство теряет данные при внезапном выключении питания и его недопустимо применять в рабочей конфигурации, но оно имеет заведомо более высокую скорость чтения и записи данных и поэтому при тестировании покажет максимально достижимый прирост производительности. Блочное устройство RAM-диска нужного размера создается с помощью загрузки модуля ядра brd [9] с опцией rd_size, задающей размер устройства в килобайтах. Далее созданное устройство /dev/ram0 готово к использованию и с точки зрения операционной системы неотличимо от обычного устройства хранения данных.

Такое тестирование производительности драйверов dm-cache и dm-writecache не совсем корректно, так как объем памяти, доступной для создания RAM-диска, обычно меньше размера даже наиболее быстрых SSD-накопителей. Однако, для задач с короткими периодами интенсивной записи данных необходимый объем кеша записи может оказаться меньше размера RAM-диска, и тогда оценка прироста производительности будет довольно точной.

6. Измерения производительности различных конфигураций дисковых подсистем

Для оценки роста производительности операций ввода-вывода при использовании быстрых SSD-накопителей в качестве кеша записи было выполнено тестирование утилитой FIO версии 3.33 [10]. В тесте использовались три накопителя данных: PCIe-плата Optane SSD 900P 280GB, использующая технологию 3D XPoint для ячеек памяти, NVMe-накопитель средней ценовой категории Apacer AS2280P4U

2TB, и механический HDD-накопитель Western Digital WD43PURZ-74BWPY0 4TB.

Запуск теста выполнялся командой

fio --bs=4k --ioengine=libaio --iodepth=32 --direct=1 --rw=randwrite --numjobs=1 ...,

которая инициирует операции записи коротких блоков данных однопотоковой нагрузкой. Этот режим соответствует наиболее требовательным прикладным программам, в которых узким местом становится ожидание подтверждения последней операции записи. Тест применялся к каждому из накопителей, а после этого к нескольким комбинациям кеш-устройства и базового устройства (асимметричный RAID1-массив не тестировался). Кеш-устройство создавалось драйвером dm-writecache в режиме writeback с параметрами по умолчанию. Длительность теста была выбрана равной 5 мин, что приблизительно соответствует среднему времени всплесков активности прикладных задач.

Результаты данного теста (Таблица 1) показывают, что кеширование записи позволяет почти полностью сохранить скорость кеш-устройства, и при этом использовать весь объем базового накопителя данных.

Таблица 1. Тесты записи блоков размером 4Kb на различные устройства

Режим тестирования	Объем устройства, Gb	IOPS	Медианная задержка, µs
Optane	261	180000	174
Apacer	1908	80000	388
HDD WD	3725	360	90000
Optane + Apacer	1908	131000	237
Apacer + HDD WD	3725	79000	388
Optane + Apacer + HDD WD	3725	123000	260

7. Заключение

Опыт использования накопительных устройств разного типа в серверах с операционной системой GNU/Linux показал, что многоуровневое кеширование позволяет увеличить производительность подсистемы хранения данных без применения наиболее дорогих накопителей категории high-end.

Работа выполнена в рамках темы государственного задания НИЦ «Курчатовский институт» - НИИСИ по теме № FNEF-2024-0001 (1023032100070-3-1.2.1).

Optimizing Input/Output using caching block devices in GNU/Linux environment

A. G. Prilipko, S. G. Romanyuk, D. V. Samborskiy

Abstract. Modern data storage devices vary widely in their key characteristics: capacity, throughput, and write latency. Due to physical limitations, no single device can maximize all of these characteristics simultaneously. However, combining different types of storage often makes it possible to optimize operating system performance for specific application workloads. The GNU/Linux kernel enables the creation of composite block I/O devices, including software RAID arrays and caching devices. This article presents an analysis of system tools and GNU/Linux kernel capabilities, with the goal of developing an algorithm for determining the optimal hardware-software configuration of the I/O subsystem.

Keywords: input-output, Linux, RAID, dm-cache, dm-writecache

Литература

1. Сайт документации ядра Linux, раздел «Device Mapper: RAID». https://www.kernel.org/doc/Documentation/device-mapper/dm-raid.txt (дата обращения 19.12.2025)

2. Сайт документации ядра Linux, раздел «Device Mapper: Cache». https://www.kernel.org/doc/Documentation/device-mapper/cache.txt (дата обращения 19.12.2025)

3. Сайт документации ядра Linux, раздел «Device Mapper: Write cache». https://www.kernel.org/doc/Documentation/device-mapper/writecache.txt (дата обращения 19.12.2025)

4. Сайт документации ядра Linux, раздел «Device Mapper: Cache policies». https://www.kernel.org/doc/Documentation/admin-guide/device-mapper/cache-policies.txt (дата обращения 19.12.2025)

5. Сайт документации ядра Linux, раздел «Memory Management». https://www.kernel.org/doc/html/latest/admin-guide/mm/index.html (дата обращения 19.12.2025)

6. А.Б. Бетелин, Г.А. Прилипко, А.Г. Прилипко, С.Г. Романюк, Д.В. Самборский. Динамический анализ и оптимизация ввода-вывода в среде виртуализации GNU Linux/QEMU/KVM. «Труды НИИСИ РАН», т.14 (2024), №1, 25-32

7. Сайт утилиты vmtouch. https://hoytech.com/vmtouch (дата обращения 19.12.2025)

8. Сайт утилиты vmprobe. https://vmprobe.com/intro (дата обращения 19.12.2025)

9. Сайт документации ядра Linux, раздел «RAM block device driver». https://www.kernel.org/doc/html/latest/admin-guide/blockdev/ramdisk.html (дата обращения 19.12.2025)

10. Сайт утилиты Flexible I/O tester (FIO). https://fio.readthedocs.io/en/latest/index.html (дата обращения 19.12.2025)

Архитектура системы для подготовки и аттестации персонала, участвующего в разработке и эксплуатации АСУ ТП

С. Е. Базаева[1]

[1] НИЦ «Курчатовский институт» — НИИСИ, Москва, Российская Федерация, bazaeva@niisi.msk.ru

Аннотация. Рассматриваются принципы построения и архитектура системы для обучения / аттестации персонала, обслуживающего АСУ ТП. Предложена архитектура тренажера на основе полунатурного стенда для подготовки и аттестации операторов и инженеров-технологов, участвующих в эксплуатации АСУ ТП.

Ключевые слова: критическая инфраструктура, АСУ ТП, полунатурный стенд, тренажер, инженерная психология

1. Введение

В данной работе рассматриваются принципы построения и архитектура системы для подготовки и аттестации персонала, чья деятельность связана с разработкой и эксплуатацией автоматизированных систем управления технологическими процессами (АСУ ТП).

Процесс создания и эксплуатации АСУ ТП широко освещен в различных публикациях [1-5]. Участниками этого процесса являются коллективы сотрудников различной численности, имеющих различное образование и квалификацию, однако все они включают инженеров-технологов, операторов и программистов.

Инженеры-технологи отвечают за разработку и модификацию (по необходимости) прикладных программ, исполняемых на программируемых логических контроллерах (ПЛК). Эти прикладные программы взаимодействуют с физическим оборудованием объекта управления и реализуют таким образом технологические процессы. Инженеры-технологи не обязаны владеть навыками программирования на высокоуровневых языках и применять эти языки в своей работе. Для программирования на ПЛК имеются специальные среды разработки и языки, в том числе, позволяющие использовать графические элементы, обозначающие различные элементы алгоритмов [6, 7].

Операторы контролируют функционирование АСУ ТП: вводят значения параметров, запускают систему, следят за диагностическими сообщениями и индикаторами, в случае нештатных ситуаций выполняют предписанные действия.

Программисты отвечают за разработку и сопровождение программ, поддерживающих передачу информации между ПЛК и ЭВМ более высоких уровней, а также за программное обеспечение (ПО), исполняемое на ЭВМ более высокого уровня (различные протоколы передачи данных, базы данных, программы накопления и обработки статистики, средства контроля несанкционированного доступа и пр.).

Следует отметить, что АСУ ТП для каждого масштабного проекта потенциально может обладать своими особенностями, даже если объекты управления являются однотипными (например, тепловые электростанции, гидроэлектростанции, шахты и т.п.). И если можно себе представить единую инструкцию по наладке и эксплуатации АСУ «умного» дома для всех домов одной определенной серии, то составить единую инструкцию для АСУ атомных электростанций (даже в случае использования однотипных реакторов) очевидно невозможно.

Поэтому традиционно для подготовки и аттестации персонала, обслуживающего АСУ масштабных проектов, одновременно с созданием АСУ ТП разрабатывается учебно-аттестационная система-тренажер.

Необходимость создания таких систем обусловлена следующими причинами:

- автоматизированные системы используются для управления технологическими процессами на объектах критической инфраструктуры, где необходимо обеспечить высокий уровень кибербезопасности [1], а персонал по сути является одним из компонентов АСУ;
- персонал должен обладать достаточной квалификацией для выполнения своих обязанностей как при штатном функционировании объекта управления, так и в случае аварийных ситуаций различного

вида (выход из строя оборудования, несанкционированное вмешательство, неумышленная ошибка персонала и т.д.);
- обучение и аттестация персонала должны осуществляться в условиях, максимально приближенных к реальной производственной обстановке с учетом особенностей конкретного объекта управления.

Система подготовки/аттестации персонала для реального действующего современного объекта управления предполагает наличие у работников среднего профессионального или высшего образования.

Однако параллельно в профильных образовательных учреждениях существуют системы подготовки и аттестации, предназначенные для студентов, направление обучения которых так или иначе связано с темой АСУ ТП. Принципы разработки обучающих систем для образовательных учреждений имеют свои особенности по сравнению реальными проектами и в данной работе не рассматриваются.

Производственная система подготовки / аттестации персонала в первую очередь предназначена для инженеров-технологов и операторов АСУ ТП, поскольку обучение и аттестация программистов происходит в системе высшего и среднего профессионального образования. Полученные программистами навыки разработки и интеграции информационных систем являются в определенном смысле универсальными. Например, на крупном промышленном предприятии придется следить за взаимодействием технологических процессов с системой контроля складских запасов, системой управления логистикой, бухгалтерией и т.д. При этом предполагается, что качество ПО АСУ должным образом контролировалось в процессе разработки, отладки и тестирования [8, 9].

В отличие от программиста, инженер-технолог, обученный в системе образования программированию на ПЛК и даже имеющий опыт работы, может столкнуться с трудностями при освоении ПО новой АСУ ТП. В частности, необходимо будет освоить систему разработки и отладки, с помощью которой создаются прикладные программы для ПЛК на конкретном объекте.

Наиболее важна система подготовки / аттестации персонала для обучения операторов АСУ ТП, поскольку, как было указано ранее, каждая АСУ для более-менее сложного объекта управления обладает своими особенностями и получить навык работы с ней можно только с помощью специально разработанной учебной системы.

Таким образом, при выборе архитектуры для системы подготовки/аттестации персонала АСУ ТП будем опираться на следующие принципы:
- система обеспечивает обучение и аттестацию операторов и инженеров-технологов для эксплуатируемой АСУ ТП;
- в состав системы включен штатный пульт управления для оператора;
- в состав системы включена штатная программно-аппаратная среда для разработки, загрузки и модификации инженером-технологом встраиваемого прикладного ПО;
- в состав системы включено рабочее место инструктора для загрузки задач и контроля действий обучаемого / аттестуемого персонала;
- система функционирует в реальном масштабе времени, соответствующем реальным условиям работы АСУ ТП;
- общие принципы разработки аппаратно-программных систем (модульность, возможность настройки, использование стандартов разработки и т.п.).

Далее предлагается рассмотреть реализацию архитектуры системы подготовки/аттестации персонала в виде тренажера на основе полунатурного стенда, имитирующего работу реального объекта под управлением АСУ. Данный подход отвечает перечисленным принципам, поскольку концепция тренажера подразумевает многократное повторение действий под контролем инструктора с использованием специально разработанной методики, а полунатурный стенд позволяет воспроизвести для персонала условия работы, близкие к реальным.

2. Архитектура тренажера на основе полунатурного стенда для имитации работы АСУ ТП

Тренажеры для обучения, тренировки и аттестации инженеров-технологов и операторов технологических процессов, функционирующих под управлением АСУ, широко применяются в различных отраслях промышленного производства, в авиации и космонавтике, в судовождении, в энергетике, в нефтегазовой отрасли и т.д. [10, 11]. Рассмотрим возможности и характеристики тренажеров на основе полунатурных стендов, моделирующих функционирование АСУ ТП, отличающие их от других средств обучения и аттестации операторов АСУ ТП.

2.1. Ключевые характеристики технологических тренажеров

Понятие «тренажер для обучения персонала» нередко трактуют чрезмерно широко, называя так любой источник полезных сведений об управлении технологическим процессом, который не является обычным бумажным учебником или инструкцией, тем более, если этот источник доступен на компьютере.

«Настоящие» же тренажеры предназначены для передачи знаний об организации практической деятельности в отличие от декларативных знаний о технологическом процессе, содержащихся в учебниках, опросниках, системах тестирования, учебных фильмах и пр. Результатом занятий обучаемого персонала на тренажерах является формирование, закрепление и совершенствование навыков управления объектом.

Перечислим принципиальные отличия технологических тренажеров от других средств обучения и обязательные составные части тренажеров, обусловленные этими отличиями [11]:

- тренажер содержит модель объекта, замещающую реальный объект, который невозможно, опасно или дорого использовать для обучения (модель объекта реализована в виде полунатурного стенда);
- на тренажере взаимодействие обучаемого с моделью объекта осуществляется посредством специальной среды, называемой информационной моделью (или интерфейсом обучаемого). С помощью данного интерфейса обучаемый воздействует на модель объекта и получает информацию о ее состоянии;
- на тренажере модель объекта, снабженная интерфейсом обучаемого, представляет собой лишь имитатор реального объекта с возможностью манипулирования им. Имитатор становится тренажером при наличии модели обучения, включающей правила, методы работы, тренировочные упражнения и прочие методические атрибуты.

Разработка каждого компонента тренажера, безусловно, должна быть выполнена качественно, то есть, на очень высоком уровне. Для модели объекта, например, критерием качества может служить степень ее подобия реальному технологическому процессу. Однако критерий качества тренажера в целом не сводится к качеству отдельных компонентов.

Тренажер призван обеспечить перенос навыков тренинга в реальную деятельность, следовательно, качество тренажера определяется уровнем соответствия деятельности обучаемого в тренинге и на практике.

2.2 Этапы развития технологии компьютерного тренинга

Разработка первых тренажеров в 60е и 70е г.г. двадцатого века выполнялась на аналоговой технике и цифровых ЭВМ с архитектурой IBM-360, 370, 390, для размещения которых требовались большие площади и мощная система охлаждения. Возможности графических интерфейсов тренажеров были более чем скромными, а производительности используемых ЭВМ было недостаточно для разработки качественных моделей.

С появлением персональных компьютеров, недорогих и более производительных, в учебных центрах предприятий и организаций появились тренажерные системы с массовым доступом. Однако для включения в их состав высокоточных моделей вычислительной мощности одного-двух ПК по-прежнему было недостаточно, поэтому предпринимались попытки создания распределенных систем. В частности, так называемые «фермы» (системы, включающие десятки ПК) применялись для разработки цифровых реалистичных изображений и анимации.

Персональные компьютеры также начали проникать на пульты управления электростанций и других производственных объектов, заменяя традиционные щиты (см. рис. 1 и 2).

Рис. 1. Щит управления на ТЭЦ-27, 90-е годы 20-го века, г. Москва [12]

Ранее создание тренажеров, включающих щит управления в качестве натурного элемента, сопровождалось проблемами из-за необходимости разработки линий связи и протоколов передачи данных между аппаратурой ввода-вывода данных щита

и цифровыми ЭВМ нового поколения.

Рис. 2. Щит управления на ТЭЦ-27, 2012 год, г. Москва [13]

К концу двадцатого века появились мощные персональные компьютеры, сети, качественная компьютерная графика. Одновременно происходило развитие систем управления техническими комплексами, возникли SCADA-системы. Стандартизация аппаратуры и программного обеспечения облегчала тиражирование разработок.

Дальнейшее развитие информационных технологий на основе роста вычислительных мощностей ЭВМ и расширения функциональных возможностей медиатехнологий оказало революционное воздействие как на системы управления технологическими процессами, так и на тренажеры, используемые для обучения персонала новым методам работы (см. рис. 2). Современный «настоящий» щит управления ТЭЦ-27 легко может быть встроен в архитектуру тренажера в полном объеме.

Параллельно с созданием тренажеров, внешне похожих на своих предшественников, развиваются новые направления – тренажеры с трехмерным изображением объекта управления и VR-тренажеры [14].

Тренинг с использованием трехмерной картинки напоминает компьютерную игру – на экране перемещается персонаж, выполняющий учебное задание под управлением обучаемого лица. Реалистичность изображения зависит от вычислительной мощности используемых ЭВМ и поставленной задачи.

VR-тренажеры демонстрируют обучаемому реалистичную качественную картинку, но пока испытывают проблемы при имитации тактильных ощущений [15, 16], особенно в случае моделирования мелкой моторики, поскольку качественное имитационное оборудование имеет слишком высокую цену.

2.3. Архитектура типового тренажера для персонала технического комплекса

Рассмотрим технический комплекс, конфигурация которого изображена на рис. 3.

Рис. 3. Конфигурация технического комплекса

Аппаратная часть технического комплекса включает:

– ПЛК;
– ЭВМ в составе АРМ оператора. Данное рабочее место используют оператор АСУ ТП и инженер-технолог;
– инструментальную ЭВМ (ИЭВМ), используемую инженером-технологом для разработки и поддержки прикладного ПО ПЛК.

В качестве ЭВМ могут быть использованы компьютеры с архитектурой x86-64 (как указано на рис. 3) или MIPS.

АРМ оператора и ПЛК соединены сетью PROFIBUS. ПЛК располагает аппаратными интерфейсами для получения сигналов от «полевых» датчиков, передачи данных исполнительным устройствам и связи с прочими смежными внешними системами через промышленные сети.

ПЛК является базовым аппаратным элементом любой системы автоматизированного управления технологическим процессом. Логика функционирования АСУ ТП, независимо от области применения, состоит в том, чтобы с помощью «полевых» датчиков, подключенных к производственному оборудованию и/или анализирующих состояние окружающей среды, передавать на более высокие уровни системы информацию о технологическом процессе и в случае необходимости оказывать управляющее воздействие на оборудование с помощью исполнительных механизмов.

Поступающие с «полевого» уровня данные обрабатываются на ПЛК прикладными программами в соответствии с задачами, поставленными перед АСУ ТП, а затем

передаются на диспетчерский уровень.

Данные, выводимые на исполнительные механизмы, формируются прикладными программами, исполняемыми на ПЛК, на основе информации, полученной от «полевых» датчиков и, возможно, от внешних систем и программ диспетчерского уровня.

Таким образом, комбинацию ПЛК + интерфейсы подключенных полевых устройств и исполнительных механизмов, показанную на рис. 3, можно считать типовой структурной единицей АСУ ТП.

В состав аппаратуры тренажера включим следующие натурные элементы объекта-оригинала:

- ЭВМ, входящую в состав АРМ оператора АСУ (архитектура x86-64);
- ПЛК (интерфейсы для полевых устройств не используются, поскольку устройства заменены моделями) и сетевой интерфейс для взаимодействия с АРМ оператора АСУ и моделями полевых устройств, исполняемыми на моделирующей ЭВМ.

Опираясь на описания типового компьютерного тренажерного комплекса для обучения операторов и инженеров технологических процессов [17, 18], построим свою конфигурацию тренажера (рис. 4).

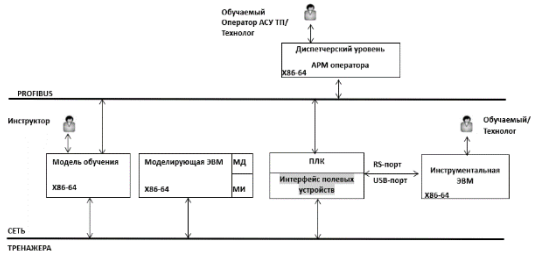

Рис. 4. Конфигурация аппаратуры тренажерного комплекса

Предлагаемая конфигурация тренажерного программно-аппаратного комплекса соответствует определению полунатурного стенда, поскольку включает натурные компоненты объекта-оригинала и модели, заменяющие отсутствующие компоненты.

ПЛК в составе конфигурации выступает в качестве натурного компонента системы управления объекта-оригинала.

На стенде (рис. 4) входные данные для алгоритмов управления выдает и выходные данные от алгоритмов управления принимает модель объекта-оригинала, исполняемая на моделирующей ЭВМ. Штатная схема подключения полевых устройств к ПЛК и алгоритмы нижнего уровня, осуществляющие прием и передачу данных в реальных условиях, здесь не используются (отмечены на рис. 4 серым цветом).

Данные от ПЛК на интерфейс обучаемого или экзаменуемого (АРМ оператора) в рассматриваемой конфигурации поступают обычным порядком, таким образом, при обеспечении функционирования программ тренажерного комплекса в реальном масштабе времени и наличии качественной модели объекта-оригинала оператор не должен чувствовать разницы между работой на стенде и в условиях реальной эксплуатации технической системы.

Модель объекта-оригинала в сочетании с алгоритмами системы управления, исполняемыми на ПЛК, составляют модель технической системы. Иногда алгоритмы системы управления переносят на моделирующую ЭВМ и создают таким образом единую модель технической системы.

При проектировании тренажера необходимо решить, какие штатные компоненты объекта-оригинала и системы управления требуется включить в состав стенда для получения желаемых результатов по завершении обучения персонала.

Например, на пилотажных стендах в точности воспроизводится лишь внутренний антураж кабины (кресла пилотов, приборная доска, панели индикаторов, тумблеры, педали, ручка управления или штурвал и пр.) и задействуется аппаратура, непосредственно обеспечивающая функционирование приборов, органов управления и прочих элементов интерьера. Макет кабины монтируется на неподвижную платформу или подвижную платформу с шестью степенями свободы, перемещение которой в пространстве обеспечивают с помощью гидроцилиндров.

Как показывает практика, установка дополнительного натурного оборудования не повышает качество модели и продуктивность тренировок. Напротив, на пилотажном стенде большое внимание уделяется программно-аппаратной имитации реалистичной закабинной обстановки с помощью средств визуализации и ощущений, испытываемых вестибулярным аппаратом пилота, в результате перемещения подвижной платформы с установленным макетом кабины. Технические средства, используемые для этих целей, не входят в состав реального летательного аппарата.

Тренажерный комплекс функционирует следующим образом. Инструктор выполняет подготовительные действия: загрузку программного обеспечения, настройку на программном и аппаратном уровне, готовит задания для обучаемого. Затем обучаемый запускает тренажер.

Модель объекта-оригинала передает в систему управления значения датчиков, вычисленные на основании начальных условий. Далее работают управляющие алгоритмы, а затем сформированные ими значения передаются на вход моделям исполнительных механизмов. Параллельно осуществляется вывод параметров, отображаемых в интерфейсе обучаемого, на АРМ оператора. Перечисленные действия выполняются циклически.

Инструктор в рамках модели обучения имеет возможность вносить возмущения в работу комплекса, имитируя сбои, отказы, аварии с помощью доступного ему интерфейса. Также инструктор может наблюдать за действиями обучаемого – для этого используется нештатное подключение к АРМ оператора по сети объекта-оригинала.

Разумеется, все наблюдаемые параметры сохраняются в базе данных по ходу эксперимента и затем анализируются. На основании этой информации делаются выводы об успешности процесса обучения.

3. Проблемы, возникающие при создании компьютерных тренажеров

Несмотря на кажущуюся простоту идеи компьютерного тренажера, следует тщательно планировать его состав и архитектуру, чтобы найти оптимальное соотношение между степенью детализации модели технической системы, задачей, которую решает тренажер, и качеством обучения.

Проблемы, возникающие при создании тренажеров, следует разделить на две группы:

- сложности построения «каркаса» той среды, в которой функционируют модель технической системы и модель обучения (интерфейс инструктора), то есть, вопросы подключения аппаратуры, разработка протоколов передачи данных между компонентами тренажера, обеспечение совместимости и взаимодействия ПО, разработанного с помощью различных систем, и т.п.;
- разработка качественной модели технической системы и методических материалов в составе модели обучения, включая автоматизированный оперативный анализ результатов в процессе обучения и при аттестации обученного персонала.

При решении проблем первого типа, очевидно, помогут лучшие практики разработки сложных программно-аппаратных систем, начиная с этапа тщательного планирования архитектуры тренажера.

Проблемы второго типа гораздо сложнее, особенно в случае моделирования масштабных производств критической инфраструктуры (нефтеперерабатывающие предприятия, газо- и нефтедобывающие комплексы, атомные электростанции), поскольку в настоящее время обычным требованием стало моделирование не только хорошо известных режимов работы, но и таких режимов и событий, которые еще не случались – например, действия персонала в условиях тяжелой аварии с плавлением активной зоны реактора [19].

С другой стороны, включение в модель всей без исключения технологической цепочки не увеличит ценность модели при использовании в составе тренажера. Производственные линии, не функционирующие в нормальном режиме, а также при пуске и останове, можно не моделировать. Вспомогательные системы, резервные схемы и параллельно работающее оборудование допустимо моделировать упрощенно [21]. Однако в модели должны быть представлены все технологические нарушения в работе оборудования и в системе управления, а также система противоаварийной защиты.

Разработка модели обучения может стать «гибридной» проблемой, например, если необходимо научить оператора правильно действовать в непривычных условиях при дефиците времени на реакцию. В этом случае для разработки модели обучения потребуется участие профессионального психолога. Затруднения вызывает также процесс составления упражнений и алгоритмов оценки качества их выполнения.

4. Тенденции развития технологии создания и применения компьютерных тренажеров

Технологии создания и применения компьютерных тренажеров для технических систем развиваются в трех направлениях [18, 22].

Во-первых, в настоящее время при конструировании тренажеров наблюдается стремление интегрировать наиболее удачные разработки разных производителей ПО, а не начинать каждый проект «с нуля».

Данному подходу способствует продолжающаяся общая тенденция к унификации интерфейсов и повышению совместимости разработок, как на аппаратном, так и на программном уровне. Таким образом, общие черты приобретают не только моделирующие комплексы, но

и разрабатываемые системы управления, что облегчает создание тренажеров из готовых компонентов.

Примером является семейство тренажеров «ТРОПА» [20, 23] компании НПФ «Круг» для персонала, обслуживающего систему управления технологическим оборудованием. Заказчик получает набор компонентов, из которых собирает собственный тренажер для обучения и аттестации работников различных отраслей промышленности, включая теплоэнергетику, нефтяную, газовую, нефтехимическую и пр. (перечень отраслей ограничен). Объектами управления могут быть котлы, паровые и газовые турбины, нефтеперерабатывающие установки, нефте- и газохранилища и т.д.

Заказчик имеет возможность изменять прикладное программное обеспечение тренажёра (алгоритмы математических моделей агрегатов, алгоритмы системы управления, базу данных системы, графический интерфейс). Все изменения осуществляются в среде разработки компьютерного тренажерного комплекса «ТРОПА» с помощью визуального интерфейса, не требующего квалификации программиста.

Во-вторых, тренажерные модели смещаются в сторону «полных» имитационных моделей. Происходит переход к инжиниринговым моделям, то есть, высокоточным моделям, используемым для технологического инжиниринга процесса производства. И если технологические вопросы, касающиеся непосредственно процесса производства, формально не требуют использования тренажеров, то отрабатывать и оптимизировать эргономику операторского интерфейса, методику обучения и аттестации персонала удобно именно с помощью тренажера.

Термин «инжиниринг» (engineering) здесь означает инженерно-консультационную деятельность, содержанием которой является решение инженерных задач, связанных с созданием или совершенствованием продукции, систем и/или процессов [24].

В настоящее время применение сложных высокоточных моделей ограничено как недостатком вычислительных ресурсов, так и уровнем самих моделей, которые могут не обладать в полной степени адекватностью для некоторых технологических областей.

Однако по мере увеличения мощности вычислительных ресурсов и появления теоретических достижений в области динамического моделирования можно будет освободиться от упрощенных тренажерных моделей.

В-третьих, происходят изменения в модели обучения. Успехи инженерной психологии в области создания эффективных схем тренинга персонала обещают новые возможности на уровне АРМ инструктора в части автоматизированных обучающих систем и систем тестирования для персонала. Уже сейчас доступны тренажеры с использованием 3D-изображений и VR-технологий, которые коренным образом меняют подход к тренингу по сравнению с эпохой использования подлинных щитов управления на электростанциях в составе тренажеров. Более того, на уровне организации ПАО «Россети» разработан корпоративный стандарт «Тренажерные комплексы на основе технологий виртуальной и дополненной реальности (VR/AR)» [25].

Рост вычислительных мощностей уже сейчас позволяет приступить к синхронизации тренажера и технологического процесса, позволяющей тренировать операторов в реальных режимах работы, а также воспроизводить при тренинге реально произошедшие нарушения в работе оборудования и отрабатывать действия по устранению нежелательных последствий.

Дальнейший прогресс в области стандартизации и расширения совместимости оборудования разных производителей, используемого в качестве компонентов АСУ ТП, позволит в будущем снизить многообразие моделируемых компонентов объекта-оригинала и сделать тренажеры более масштабными, не увеличивая радикально уровень затрат. Возможно, современные тренажеры со временем трансформируются в виртуальные аналоги целых предприятий.

5. Заключение

Система обучения/аттестации персонала является обязательным компонентом проектов автоматизации масштабных производственных комплексов.

Предложенный в данной работе подход к построению архитектуры рассматриваемой системы на основе полунатурного стенда позволяет создавать технологические тренажеры, предназначенные для передачи и/или проверки знаний об организации практической деятельности, а также для формирования, закрепления и совершенствования навыков управления реальным объектом.

Объектом дальнейших исследований должны быть архитектуры систем обучения и аттестации, предназначенных для использования в составе образовательных программ, связанных с тематикой АСУ ТП,

в учреждениях профессионального среднего и высшего образования.

Публикация выполнена в рамках государственного задания по проведению фундаментальных исследований по теме «Вопросы разработки и применения инструментальной платформы для создания автоматизированных систем управления на отечественных программно-аппаратных средствах».

Architecture of the system for training and certification of personnel involved in the development and operation of automated process control systems

S. E. Bazaeva

Abstract. The principles of design and architecture of a system for training and certification of personnel servicing automated process control systems (APCS) are discussed. A simulator architecture based on HIL simulation is proposed for training and certification of operators and process engineers involved in the operation of APCS.

Keywords: critical infrastructure, automated process control systems, HIL simulation stand, simulator, engineering psychology

Литература

1. А.И. Грюнталь, С.Е. Базаева. Вопросы обеспечения кибербезопасности при разработке и использовании АСУ ТП. // Труды научно-исследовательского института системных исследований Российской академии наук, Т. 11, № 4. С. 56-67, Москва, 2021 г.

2. ГОСТ Р 71531-2024 Системы киберфизические. Термины и определения. М.: Российский институт стандартизации, 2024. 8 с.

3. М.С. Аристов, А.И. Грюнталь, Я.А. Зотов, Я.А. Шаповалов, Д.В. Яриков. Архитектура типовой системы автоматизации технологических процессов на базе отечественных СВТ и ПО. // Труды научно-исследовательского института системных исследований Российской академии наук, Т. 13, № 4. С. 64-67, Москва, 2023 г.

4. В.В. Девятков. Методология и технология имитационных исследований сложных систем: современное состояние и перспективы развития: монография. М.: Вузовский учебник: ИНФРА-М, 2013 г. 448 с.

5. Программно-аппаратный (программно-технический) комплекс КРУГ-2000. Источник: https://www.krug2000.ru/products/ptk.html (Дата обращения: 11.10.2025).

6. Документация по среде разработки «Beremiz». Источник: https://beremiz-ru.readthedocs.io/ru/master/ (Дата обращения: 11.10.2025).

7. ГОСТ Р МЭК 61131-3-2016 «Контроллеры программируемые. Часть 3: Языки программирования». Источник: https://docs.cntd.ru/document/1200135008 (Дата обращения: 11.10.2025).

8. Я.А. Зотов. Вопросы разработки системы автоматизированного тестирования программного обеспечения технических комплексов АСУ ТП // Программная инженерия. 2025. Том 16, № 3, С. 122-133.

9. Я.А. Зотов. Разработка инструментальной платформы для создания стендов полунатурного моделирования. // Распределенные компьютерные и телекоммуникационные сети: управление, вычисление, связь: материалы XXVIII Междунар. научн. конфер, 22–26 сент. 2025 г., Москва / под общ. ред. В.М. Вишневского, К.Е. Самуйлова; Ин-т проблем упр. им. В.А. Трапезникова. С. 63-68.

10. Применение компьютерных тренажеров в процессе комплексного обучения персонала буровых установок в соответствии с требованиями IWCF и IADC // Бурение и нефть. 2023. № 7-8. С. 38-43.

11. В.М. Дозорцев, Д.В. Кнеллер. Технологические компьютерные тренажеры: все, что вы всегда

хотели знать // Промышленные АСУ и контроллеры. 2004. № 12. С. 1-13.

12. Музей истории Мосэнерго. Северная ТЭЦ (ТЭЦ-27). Источник: https://www.mosenergo-museum.ru/History_of_Mosenergo/Historical_Review/21985/ (Дата обращения: 11.05.2025).

13. Т.И. Башкаев. Единый центральный щит управления ТЭЦ 27 в Москве. // АрхРевю, 12.05.2012. Источник: https://www.archrevue.ru/profilearchitectblogmessage/9158.html (Дата обращения: 11.05.2025)

14. VR + точная тактильная отдача. Имитационный тренажер буровой установки ZBO S15. Источник: https://habr.com/ru/articles/663664/ (Дата обращения: 11.05.2025).

15. VR-тренажёры и иммерсивное обучение. Источник: https://habr.com/ru/articles/769160/ (Дата обращения: 11.05.2025).

16. В.П. Овчинников, М.Д. Гаммер, А.В. Епихин, Ю.А. Гильманов. Тренажер-симулятор процесса бурения нового поколения. // Бурение и нефть. 2025. № 3. С. 42-48.

17. В.М. Дозорцев, Д.В. Кнеллер. Типовой компьютерный тренажерный комплекс для обучения операторов ТП // Автоматизация в промышленности. 2003. № 2. С. 9-14.

18. В.М. Дозорцев. Компьютерный тренинг операторов технологических процессов нефтяного комплекса: состояние и тенденции развития. Источник: https://www.researchgate.net/publication/283343494_komputernyj_trening_operatorov_tehnologiceskih_processov_neftanogo_kompleksa_sostoanie_i_tendencii_razvitia. (Дата обращения: 11.05.2025).

19. Как создают и совершенствуют тренажеры для оперативного персонала АЭС. Интервью зам. директора ВНИИАЭС НТП В. Чернакова. Источник: https://strana-rosatom.ru/2023/07/20/kak-sozdajut-i-sovershenstvujut-trenazh/ (Дата обращения: 11.05.2025).

20. Тренажер для персонала, обслуживающего АСУ ТП технологических установок. Источник: https://www.krug2000.ru/decisions/trenajery-podgotovki-operatorov-technologov/1786.html (Дата обращения: 11.05.2025).

21. В.М. Дозорцев. Имитационные модели технологических процессов в компьютерных тренажерах для обучения операторов // Имитационное моделирование. Теория и практика: Сборник докладов третьей всероссийской научно-практической конференции ИММОД-2007. Том 2. СПб.: ФГУП ЦНИИТС. 2007. – с. 58-61.

22. В.М. Дозорцев. Компьютерные тренажеры для обучения операторов технологических процессов: история, состояние, перспективы // В кн. Труды Института проблем управления РАН, Т.1. – М., 1998. – 95 с.

23. Программно-аппаратный (программно-технический) комплекс КРУГ-2000. Источник: https://www.krug2000.ru/products/ptk.html (Дата обращения: 11.05.2025).

24. ГОСТ Р 57306-2016 Инжиниринг. Терминология и основные понятия в области инжиниринга. Стандартинформ (2018 г.) 15 с.

25. СТО 34.01-22-001-2022 Тренажерные комплексы на основе технологий виртуальной и дополненной реальности (VR/AR). Типовые технические требования. ПАО «Россети» (2022г.) 62 с. Источник: https://www.rosseti.ru/upload/iblock/94f/cdfarq7dyfwbwar82ougf9rk41n10266.pdf (Дата обращения: 11.05.2025).

Анализ изменений стандарта MISRA-C 2023

К. А. Костюхин[1], Г. Л. Левченкова[2]

[1]НИЦ "Курчатовский институт" - НИИСИ, Москва, kost@niisi.ras.ru;

[2]НИЦ "Курчатовский институт" - НИИСИ, Москва, galka@niisi.ras.ru

Аннотация. Работа содержит краткий обзор основных изменений стандарта MISRA C 2023 по сравнению с его предыдущей редакцией 2012 года. рассматриваются принципиальные изменения в структуре стандарта, новые требования и рекомендации, а также практические последствия для разработчиков встроенных и критичных к безопасности систем.

Ключевые слова: язык Си, стандарт, MISRA C 2023

1. Введение

Стандарт MISRA C (Motor Industry Software Reliability Association) [1] возник как ответ на потребности автомобильной индустрии в предписывающих руководящих принципах для безопасного, переносимого и проверяемого использования языка C. С течением времени он превратился в де-факто эталон для множества отраслей, где надежность программного обеспечения критична: авиация, медицина, железнодорожный транспорт, промышленные контроллеры и т.п.

Версия MISRA C 2012 [2] установила базу для строгой дисциплины программирования на языке C с четкой классификацией правил, директив и механизмов доказательства соответствия. Однако развитие языка C (включая стандарты C11, C17 и C23), рост числа многопоточных/параллельных программ, а также накопившийся опыт применения MISRA C в промышленности требовали пересмотра и обновления набора правил. В результате появился MISRA C 2023 [3], который, при соблюдении преемственности, вносит ряд существенных изменений: обновленную структуру правил, новые или переработанные требования, уточнения в терминологии и механизмах доказательства соответствия, а также усиление внимания к современным аспектам, таким как порядок вычислений, многопоточность и взаимодействие с компиляторными расширениями.

Цель данной статьи - выполнить систематическую декомпозицию отличий между MISRA C 2012 (AMD1+AMD2) и MISRA C 2023, предоставить объяснение существенных новых или измененных правил и проиллюстрировать их примерами.

2. Обзор изменений в структуре стандарта

2.1. Переработка организационной структуры и классификации правил

Ключевое изменение. MISRA C 2023 переосмысливает и уточняет структуру правил: происходит более строгая сегрегация требований по уровням обязательности, четкая фиксация зависимостей между правилами и директивами, а также улучшенное разделение между строго нормативными требованиями и поясняющими рекомендациями. Это отражено в переработанных заголовках правил и их аннотациях.

Почему это важно. В прошлой версии (MISRA C 2012) некоторые правила и директивы могли трактоваться неоднозначно при реализации процессов соответствия. Новая структура снимает неоднозначности, облегчая реализацию и аудит: инструменты статического анализа теперь могут точнее отображать причину нарушения, уровень строгости и возможную стратегию соответствия.

Пример (иллюстрация структурной перестройки).

Исходный контекст: правило X (в MISRA C 2012) могло быть классифицировано как рекомендация, однако его нарушение на практике приводило к критическим ошибкам. MISRA C 2023 перенесло это положение в разряд обязательных или ввело сопутствующую директиву для уточнения области применения.

Код:

/* Пример: использование динамического приведения типов для работы с бинарными пакетами */

```
void process (void *data) {
   uint32_t *p = (uint32_t *)data;
/* предположение: data выровнено и указывает на 32-битный блок */
   uint32_t v = *p;
   /* ... */
}
```

Что иллюстрирует пример. Данный фрагмент иллюстрирует ситуацию, где код полагается на предположения о выравнивании и представлении данных, что может быть небезопасно на разных архитектурах. В MISRA C 2023 такие конструкции получили более жесткую классификацию и дополнились директивой по обязательной верификации условий (alignment/representation) при использовании подобных конструкций.

Действие для приведения к стандарту. Явная проверка выравнивания, использование memcpy для копирования вместо безопасного приведения, или структурированное чтение байтов с учетом их порядка. Все эти меры иллюстрируют практическую адаптацию.

2.2. Расширение охвата языка C и явный учет современных языковых конструкций

Ключевое изменение. MISRA C 2023 официально учитывает развитие языка C после C11 (включая особенности, включенные в C17/C23) и уточняет совместимость правил с конструкциями современных компиляторов, в том числе с механизмами атомарных операций, ограничениями на порядок вычислений и новыми типовыми возможностями. Это зафиксировано в ряде новых заголовков и пояснений.

Почему это важно. Современные проекты чаще используют элементы стандарта C, появившиеся после C99, включая атомарные типы <stdatomic.h>, условную компиляцию для разных стандартов и новые правила оптимизации компиляторов. Для поддержания релевантности MISRA C должен явным образом описывать поведение правил для этих конструкций.

Пример (атомарные операции и порядок выполнения).

Код:

```
#include <stdatomic.h>

atomic_int flag = 0;
int shared = 0;

void writer (void) {
   shared = 42;              /* (1) */
   atomic_store_explicit (&flag, 1, memory_order_release); /* (2) */
}

void reader (void) {
   if (atomic_load_explicit (&flag, memory_order_acquire) == 1) { /* (3) */
      /* гарантируется видимость записи shared */
      int v = shared;       /* (4) */
   }
}
```

Что иллюстрирует пример. Использование атомарных операций с семантикой release/acquire для синхронизации записи и чтения некоторых переменных. MISRA C 2023 более явно рассматривает такие поведенческие шаблоны и дает рекомендации по корректному использованию атомарных операций, чтобы избежать неопределенного поведения и несогласованных состояний в многопоточной среде.

Действие для приведения к стандарту. Правила стандарта требуют однозначности - если код использует атомарные операции, эти операции должны иметь явно заданный порядок доступа к памяти, и их применение должно сопровождаться комментариями/доказательствами, почему выбранный порядок корректен в контексте проекта.

2.3. Уточнения терминологии и механизма доказательства соответствия (compliance)

Ключевое изменение. MISRA C 2023 уточняет термины: что имеется в виду под «нарушением», «директивой», «рекомендацией», «правилом обязательного характера», а также отношения между «доказательством соответствия» и «допустимыми отклонениями (deviations)». Эти уточнения помогают аудиторам и инженерам точно интерпретировать выявленные нарушениях и корректно оформлять отклонения.

Почему это важно. Четкое понимание классификации нарушений – это основа для корректных процедур аудита, тестирования и документирования решений об отклонениях. Без этого разработчики могут неправильно относить предупреждения компилятора/анализатора к категории «рекомендация» вместо «обязательное требование», и наоборот.

Пример (доказательство соответствия).

Сценарий: Проект использует низкоуровневую операцию ввода-вывода,

которая по дизайну нарушает правило X (например, прямой доступ к регистрам через приведенный указатель). В MISRA C 2012 это могло быть признано допустимым при оформлении отклонения. MISRA C 2023 требует более формального доказательства: документального анализа, тестов и статического/динамического обоснования безопасности.

Что иллюстрирует пример. Необходимость формальной документации при отклонениях - не только запись «отклонение разрешено», но и конкретные ссылки на тесты, статический/динамический анализ и условия эксплуатации. Это повышает надежность процедур соответствия.

3. Новые и существенно измененные требования MISRA C 2023

Ниже приведен разбор ключевых новых требований и переработок. Для удобства они сгруппированы по тематике: управление неопределенным поведением, типовая безопасность и преобразования, обработка указателей и выравнивание, порядок вычислений и побочные эффекты, многопоточность и параллелизм, взаимодействие с компиляторными расширениями, проверяемость и инструментальная поддержка, процесс отклонений.

3.1. Управление неопределенным поведением (Undefined Behavior, UB)

Изменение. MISRA C 2023 усиливает требования по избеганию ситуаций, приводящих к неопределенному поведению в языке C. В тексте правил уделено повышенное внимание таким ситуациям, как: переполнение знаковых целых типов, некорректный доступ по выровненному указателю, чтение неинициализированной памяти, использование освобожденной памяти и некорректные преобразования типов.

Почему это важно. UB может приводить к непредсказуемым последствиям: от ошибочных результатов до аварийной остановки или эксплуатации уязвимостей. Современные компиляторы активно используют UB для оптимизаций, следовательно, код, полагающийся на UB, может вести себя по-разному в разных версиях компилятора или с разными параметрами оптимизации.

Пример 1 (переполнение знакового типа).

Код:

```
#include <stdint.h>

int32_t sum (int32_t a, int32_t b) {
  return a + b; /* потенциальное переполнение */
}
```

Что происходит. При сложении двух знаковых 32-битных чисел может произойти переполнение, которое в стандарте C является неопределенным поведением. Компилятор может предполагать, что такого переполнения не бывает, и оптимизировать код исходя из этого предположения, что приведет к неожиданным результатам в случае переполнения.

MISRA C 2023. Указывает на необходимость либо использовать беззнаковую арифметику с проверкой переполнения, либо производить проверки до операции, либо использовать безопасные функции/макросы, которые явно документируют поведение при переполнении.

Как исправлять. Один из подходов - приведение к 64-битному типу с проверкой:

```
int32_t sum_safe (int32_t a, int32_t b) {
  int64_t tmp = (int64_t)a + (int64_t)b;
  if (tmp > INT32_MAX) {
    /* обработка ошибки */
  }
  if (tmp < INT32_MIN) {
    /* обработка ошибки */
  }
  return (int32_t)tmp;
}
```

Пример 2 (чтение неинициализированной памяти).

Код:

```
int f (void) {
  int x; /* неинициализирован */
  return x + 1;
}
```

Что происходит. Чтение x до инициализации, классический пример неопределенного поведения. MISRA C 2023 ужесточает требования по обнаружению подобных операций и рекомендует явно инициализировать переменные или использовать статический анализ для доказательства инициализации.

Практика. Использование инструментов статического анализа, добавление правил кодирования, обязательные обзоры и тесты покрывают такие проблемы.

3.2. Порядок вычислений и побочные эффекты

Изменение. MISRA C 2023 сильнее фокусируется на проблемах, связанных с неопределенным порядком вычислений выражений, в результате чего пересматриваются и уточняются правила, запрещающие конструкции, зависящие от порядка вычислений подвыражений (например, модификация объекта более одного раза между последовательными точками наблюдения).

Почему это важно. Распространенной категорией ошибок является использование выражений вроде i = i++ + 1; или a[i++] = i; где результат зависит от порядка вычислений. Компиляторы и стандарты языка допускают вариативное поведение, следовательно, такие конструкции приводят к UB.

Пример (выражение с побочными эффектами).

Код:

```
int i = 0;
int a[2];
a[i] = ++i;
```

Что происходит. Поведение не определено: одновременно читается и модифицируется i без промежуточной последовательной точки наблюдения. MISRA C 2023 требует запрета таких конструкций и рекомендует разделять операции:

```
int temp = ++i;
a[i] = temp;
```

Особое внимание следует уделять выражениям с функциями и порядком вычислений их аргументов. MISRA C 2023 вводит пояснения о том, что порядок вычисления аргументов функций является зависимым от реализации, поэтому код должен не допускать зависимостей между аргументами, где один аргумент модифицирует объект, читаемый в другом аргументе.

3.3. Указатели, выравнивание и приведенные типы

Изменение. В MISRA C 2023 усилена позиция по наглядности и безопасности операций с указателями: переработаны требования к приведению указателей и обеспечению корректного выравнивания.

Почему это важно. Неправильное выравнивание может привести к аппаратным исключениям на некоторых архитектурах, а неявные преобразования типов указателей могут скрывать ошибки. Это особенно критично в низкоуровневом коде (драйверы, работа с регистровыми картами).

Пример (приведение void* к типу с более строгим выравниванием).

Код:

```
void *buffer = get_unaligned_buffer ();
uint32_t *p = (uint32_t *)buffer;
uint32_t v = *p; /* потенциальное нарушение выравнивания */
```

Что происходит. Если buffer не выровнен по 4-байтовой границе, то чтение *p может вызывать аппаратную ошибку. MISRA C 2023 требует либо проверки выравнивания перед такими операциями, либо использование memcpy для безопасного копирования.

Исправление:

```
uint32_t v;
memcpy (&v, buffer, sizeof v);
```

Пример (strict aliasing).

Код:

```
float f = 1.0f;
int *p = (int *)&f;
int i = *p; /* может нарушать strict aliasing */
```

Что происходит. Чтение float через int* нарушает правило strict aliasing и может вызвать неопределенное поведение при оптимизациях. MISRA C 2023 более явным образом рекомендует избегать подобных приведений и использовать, например, memcpy.

3.4. Типовая безопасность и преобразования

Изменение. Стандарт усиливает требования к контролю за неявными преобразованиями, особенно между знаковыми и беззнаковыми типами, при преобразовании целочисленных типов различной ширины, а также при использовании литералов и макросов, которые могут приводить к неожиданным расширениям типа.

Почему это важно. Неправильные преобразования приводят к логическим ошибкам и уязвимостям, в частности когда отрицательные значения трактуются как большие положительные после преобразования в беззнаковый тип.

Пример (смешивание знаковых и беззнаковых типов).

Код:

```
int32_t a = -1;
uint32_t b = 1;
if (a < b) { /* сравнение int32_t и uint32_t */
   /* ... */
}
```

Что происходит. В выражении a < b значение a будет преобразовано к uint32_t, что дает большое положительное число, и поэтому условие, вероятно, окажется ложным, что не соответствует интуитивному ожиданию. MISRA

C 2023 заставляет избегать таких неочевидных сравнений и требует явного приведения типов или использования временных переменных.

Исправление:

```
if ((int32_t)a < (int32_t)b) { /* явное приведение и проверка */
    /* ... */
}
```

3.5. Многопоточность, атомарные операции и взаимодействие с памятью

Изменение. В MISRA C 2023 появляется более формализованное обсуждение многопоточности, атомарности и порядка доступа к памяти. Это включает рекомендации по использованию <stdatomic.h>, корректной последовательности вызовов для обращения к памяти и необходимости документирования инвариантов при многопоточном доступе к общим объектам.

Почему это важно. Современные встраиваемые системы все чаще полагаются на многопоточность и аппаратный параллелизм. Неправильное использование неблокирующих конструкций и атомарных операций может приводить к трудноуловимым ситуациям гонки и UB.

Пример (неправильное использование атомарности).

Код:

```
#include <stdatomic.h>

int counter = 0; /* неатомарная переменная */

void inc (void) {
  counter++; /* гонка при параллельном доступе */
}
```

Что происходит. Инкремент counter++ не является атомарной операцией, следовательно, при конкурирующем доступе возможны потерянные значения counter. MISRA C 2023 рекомендует использовать атомарные типы:

```
atomic_int counter = ATOMIC_VAR_INIT(0);

void inc (void) {
  atomic_fetch_add_explicit (&counter, 1, memory_order_relaxed);
}
```

Дополнение. Стандарт требует также документировать ожидаемую семантику памяти и показать, почему выбранный порядок доступа к памяти (в примере - memory_order_relaxed) является достаточным для корректности.

3.6. Взаимодействие с расширениями компилятора и платформозависимыми конструкциями

Изменение. MISRA C 2023 дает более формализованные указания по использованию компиляторных расширений, встроенных ассемблерных вставок, специфичных платформенных API и пр. Новые объяснения направлены на то, чтобы разработчик явно документировал причину и контекст использования расширения, а также включал доказательства того, что поведение, выходящее за рамки стандарта C, находится под контролем и безопасно.

Почему это важно. Расширения компилятора часто используются для оптимизации или доступа к аппаратуре; при этом они нарушают переносимость и могут скрывать UB. Четкая документация и ограничения предотвращают непреднамеренные последствия.

Пример (встроенный ассемблер).

Код:

```
int read_hw (void) {
  int val;
  __asm__ volatile ("in %0, 0x60" : "=r"(val));
  return val;
}
```

Что происходит. Встроенный ассемблер выходит за рамки стандарта C. MISRA C 2023 допускает его использование при условии оформления отклонения/пояснения: указать причину, влияние на переносимость, предусмотреть альтернативные реализации и тесты.

Практика. Добавление макроса-обвязки, тестов эмуляции и документирование вызовов обеспечивает соответствие требованиям.

3.7. Процесс оформления отклонений (deviations) и требования к документации

Изменение. MISRA C 2023 уточняет и формализует процесс оформления отклонений: какие шаги должны присутствовать в документе об отклонении, какие доказательства требуются, и какие тесты/проверки должны проводиться.

Почему это важно. Отклонения - это не «дыра» в стандарте, а формализованный процесс управления исключениями. Повышенные требования к доказательствам и тестам делают отклонения менее рискованными и более прозрачными для аудита.

Пример (ориентировочная структура документа отклонения).

Документ должен содержать:

- Идентификатор отклонения и ссылку на правило.
- Обоснование (архитектурное, аппаратное, производительное).
- Варианты минимизации рисков от использования отклонения.
- Список тестов и результаты статического анализа, которые подтверждают безопасность.
- Ответственное лицо и срок пересмотра.

4. Выборочный разбор новых правил и пояснения с примерами

Следующий раздел содержит последовательный разбор наиболее значимых новых или существенно измененных правил, как они отражены в заголовках MISRA C 2023, их практическое значение и развернутые примеры.

Важное замечание. Ниже приведены тематически независимые пункты. Каждый из них ориентирован на отдельную проблему безопасности или надежности кода на языке C и иллюстрирован примером.

4.1. Требования по выявлению операций чтения неинициализированных объектов

MISRA C 2023 усиливает требование по обнаружению ситуаций, где читается неинициализированная память. Инструменты должны обнаруживать такие чтения на этапе статического анализа, а код должен либо явно инициализировать объекты, либо иметь доказательство инициализации до чтения.

Пример (локальная переменная, используемая без инициализации).

Код:

```
int compute(void) {
   int x;          /* неинициализированная переменная */
   if (some_cond ()) {
      x = 10;
   }
   return x;          /* чтение может быть неопределенным */
}
```

Анализ. Если some_cond() иногда ложно, то x останется неинициализированным.

MISRA C 2023 требует либо предоставить инициализацию по умолчанию (int x = 0;), либо изменить код, чтобы гарантировать присваивание перед возвратом.

Рекомендация. Предпочтительно инициализировать переменные при объявлении, особенно для скалярных локальных переменных, либо использовать явные проверяемые пути присваивания.

4.2. Уточнение правил работы с switch и case

MISRA C 2023 требует ясности при использовании switch/case: все допустимые варианты должны быть явно обработаны, а «проваливание» (fallthrough) из одного case в другой допустимо только при явном указании и обосновании. Это уменьшает вероятность ошибок при добавлении новых case-ветвей.

Пример (неявное проваливание).

Код:

```
switch (x) {
case 1:
   do_a();
case 2:  /* неявное проваливание */
   do_b();
   break;
}
```

Что происходит. Если разработчик явно полагался на поведение «проваливания», это может быть трудно заметно при анализе кода. MISRA C 2023 требует явной пометки и обоснования:

```
switch (x) {
case 1:
   do_a();
   /* fallthrough */ /* обоснование: Intentional fallthrough */
case 2:
   do_b();
   break;
}
```

4.3. Новые требования по явной обработке ошибок и возвратных кодов

Повышены требования к проверке возвратных кодов функций, особенно тех, которые могут указывать на ошибку (например, функции ввода-вывода, системные вызовы). MISRA C 2023 предписывает явную обработку таких кодов или документированное обоснование, почему их можно игнорировать.

Пример (игнорирование кода возврата).

Код:

```
int res = write (fd, buf, n);
/* Далее res нигде не используется, игнорирование */
```

Что происходит. Игнорирование результата операции может привести к потере данных. Новая политика требует либо обработки res, либо оформленного отклонения с указанием причин.

4.4. Расширенные требования по анализу времени выполнения (timing) и влиянию на безопасность

MISRA C 2023 акцентирует внимание на коде, где временные характеристики (латентность, дедлайн) критичны для безопасности. Новые аннотации правил требуют, чтобы код, влияющий на временную надежность, был документирован и прошел соответствующие измерения/тесты.

Пример (цикл с потенциально неопределенной длительностью).

Код:

```
while (!hardware_ready ()) {
   /* пустой цикл ожидания */
}
```

Что происходит. Если hardware_ready () никогда не станет истинной, цикл будет бесконечным, что может иметь критические последствия. MISRA C 2023 требует установки явного таймаута и наличия стратегий выхода из цикла:

```
unsigned timeout = 1000u;
while (!hardware_ready () && timeout--) {
   /* ожидание с тайм-аутом */
}
if (timeout == 0u) {
   /* обработка ошибки */
}
```

4.5. Требования по фиксации и аудиту операций со смещениями и индексами массивов

Уточнены правила про доступ к массивам: индексы должны быть проверены, значения границ явно документированы, и использование вычисляемых индексов без проверки должно быть обосновано.

Пример (индексирование с риском выхода за границы).

Код:

```
int get_value (int *arr, size_t idx) {
   return arr[idx]; /* нет проверки границ */
}
```

Что происходит. Без гарантии на размер arr доступ к arr[idx] может выйти за пределы массива. MISRA C 2023 рекомендует либо передавать размер массива вместе с указателем, либо использовать API, где длина известна, либо выполнять проверки.

4.6. Новые положения, касающиеся использования макросов и генерации кода через препроцессор

MISRA C 2023 ужесточает требования к макросам: сложные макросы, содержащие инструкции, влияющие на ход выполнения потока, многострочные выражения или побочные эффекты, должны быть сокращены и заменены inline функциями или тщательно документированы. Макрос, меняющий семантику исходного кода, требует отдельного обоснования и тестов.

Пример (макрос с побочными эффектами).

Код:

```
#define MAX(a,b) ((a) > (b) ? (a) : (b))
int x = MAX(i++, j++);
```

Что происходит. Макрос приводит к двойному выполнению i++ или j++, что является непредвиденным поведением. MISRA C 2023 требует избегать таких макросов или использовать inline функции:

```
static inline int max_int(int a, int b) {
   return (a > b) ? a : b;
}
```

5. Инструменты и практическая интеграция (рекомендации)

MISRA публикует упрощенные тексты правил (headlines) для интеграции с анализаторами кода - это позволяет инструментам сопоставлять предупреждение с правилом MISRA и облегчает отчетность.

Ниже приведены рекомендации по внедрению в проект.

Обновление процесса анализа кода. Включить проверки на соответствие MISRA C 2023 в обязательный процесс анализа кода. Ошибки высокого уровня должны блокировать процесс слияния веток кода или сборки.

Обучение команды. Провести семинары по новым правилам и их примерам; обратить внимание на тонкие места: порядок вычислений, строгую семантику, атомарность.

Стратегия обработки отклонений. Разработать шаблон для документа, описывающего отклонения в соответствии с требованиями MISRA C 2023.

Миграция кода. Постепенно производить рефакторинг проблемных областей: макросы → inline-функции; неинициализированные переменные → явная инициализация; неатомарные глобальные переменные → атомарные типы/мьютексы там, где требуется.

Инструментальная автоматизация. Настроить процедуру интеграции для регулярного запуска статического анализа с отчетностью по тенденциям (новые нарушения, устраненные, продолжающиеся).

6. Заключение

MISRA C 2023 представляет собой значимое эволюционное обновление набора правил, нацеленное на повышение надежности, предсказуемости и проверяемости кода на языке C, особенно в задачах критичных к безопасности. Основные направления изменений: ужесточение требований по управлению неопределенным поведением, уточнение правил работы с указателями и выравниванием, более явное рассмотрение современной семантики языка (атомарность, порядок вычислений), усиление требований к документированию отклонений и улучшение интеграции с инструментами статического анализа. Эти изменения отражают опыт применения MISRA в промышленности и учитывают современные практики разработки ПО.

Практическая польза от миграции на MISRA C 2023 очевидна: снижение числа трудноуловимых дефектов, улучшение переносимости и предсказуемости поведения на разных компиляторах и архитектурах, а также повышение качества процедур аудита и управления отклонениями. Однако внедрение потребует усилий: обновления инструментов, обучения кадров и рефакторинга унаследованного кода. Поэтому организациям рекомендуется планировать поэтапную миграцию с выделенными ресурсами на адаптацию инструментов и обучение.

MISRA C 2023 - это современный и прагматичный шаг вперед в развитии правил надежного программирования на языке C. Он способствует снижению рисков, связанных с неопределенным поведением и контекстно зависимыми ошибками, делая управление широким набором уязвимостей и дефектов более формализованным и предсказуемым.

«Публикация выполнена в рамках государственного задания НИЦ «Курчатовский институт» - НИИСИ» по теме FNEF-2024-0001».

MISRA C 2023 brief overview

K. A. Kostiukhin, G. L. Levchenkova

Abstract. The paper provides a brief overview of the main changes in the MISRA C 2023 standard compared to its previous version in 2012. Fundamental changes in the structure of the standard, new requirements and recommendations, as well as practical implications for developers of embedded and security-critical systems are considered.

Keywords: C language, standard, MISRA C 2023

Литература

1. A. Homann, L. Grabinger, F. Hauser, J. Mottok. An Eye Tracking Study on MISRA C Coding Guidelines. In Proceedings of the 5th European Conference on Software Engineering Education (ECSEE '23). Association for Computing Machinery, 2023, New York, NY, USA, pp. 130–137. https://doi.org/10.1145/3593663.3593671
2. MISRA C C 2012 recommendations, https://gitlab.com/MISRA/MISRA-C/MISRA-C-2012/tools/-/blob/main/misra_c_2012__headlines_for_cppcheck%20-%20AMD1+AMD2.txt
3. MISRA C C 2023 recommendations, https://gitlab.com/MISRA/MISRA-C/MISRA-C-2012/tools/-/blob/main/misra_c_2023__headlines_for_cppcheck.txt

Методы реализации резервирования процессорных модулей для Багет-ПЛК1

Я. А. Зотов[1], Д. В. Яриков[2]

[1]НИЦ «Курчатовский институт» – НИИСИ, Москва, Россия, zotov@niisi.ras.ru

[2]НИЦ «Курчатовский институт» – НИИСИ, Москва, Россия, yarikov@niisi.ras.ru

Аннотация. В статье рассматриваются методы реализации резервирования процессорных модулей для отечественного программируемого логического контроллера Багет-ПЛК1. Предложен алгоритм резервирования с синхронизацией данных между основным и резервным процессорными модулями по сети Ethernet, обеспечивающий бесшовное переключение в случае отказа основного модуля. Описаны особенности программной реализации на основе трансляции прикладных программ с языков МЭК 61131-3 в код на языке Си, а также механизм автоматического определения ролей модулей и восстановления работоспособности системы. Решение предназначено для применения в системах управления объектами критической инфраструктуры.

Ключевые слова: резервирование, процессорные модули, Багет-ПЛК1, синхронизация данных, бесшовное переключение

1. Введение

В НИЦ «Курчатовский институт» НИИСИ разработаны программируемые логические контроллеры семейства Багет-ПЛК1 (далее ПЛК) на основе отечественного микропроцессора 1890ВМ108 [1]. ПЛК предназначены для сбора, преобразования, обработки, хранения информации и выработки команд управления в режиме реального времени в автоматизированных системах управления (АСУ) объектов критической инфраструктуры.

ПЛК содержит один или несколько процессорных модулей и модулей ввода-вывода, обеспечивающие взаимодействие ПЛК с периферийными устройствами по дискретным и аналоговым сигналам, а также по последовательным интерфейсам. Процессорные модули и модули ввода-вывода взаимодействуют между собой посредством аппаратной шины RS-485 с программным интерфейсом Modbus RTU.

ПЛК функционирует под управлением отечественной операционной системы реального времени ОС РВ Багет 2.7 (далее ОСРВ). ОСРВ предназначена для создания программного обеспечения вычислительных систем, комплексов и средств автоматизированного управления, работающих в режиме реального масштаба времени [2].

Описанная в данной работе реализация резервирования процессорных модулей предназначена для выполнения в прикладной программе, созданной с помощью инструментальной программы «Среда разработки и отладки» (СРиО). Программа СРиО предназначена для создания и отладки прикладных программ на языках, специализированных в стандарте МЭК 61131-3-2016 (далее МЭК) для ПЛК [3]. Помимо других функций программа СРиО осуществляет трансляцию программ с языков МЭК в программу на языке Си для ОСРВ, который после компилируется в бинарный исполняемый файл для ПЛК.

2. Резервирование

Резервирование — это метод повышения характеристик надёжности технических устройств или поддержания их на требуемом уровне посредством использования аппаратной избыточности за счет включения запасных (резервных) элементов и связей, дополнительных по сравнению с минимально необходимым для выполнения заданных функций в данных условиях функционирования [4].

Элементы устройства, обеспечивающие его работоспособность, называются основными элементами; резервными элементами называются элементы, предназначенные для обеспечения работоспособности устройства в случае отказа основных элементов.

Резервирование процессорных модулей ПЛК может осуществляться двумя методами: активным и пассивным. При активном резервировании два или более процессорных модуля работают одновременно, обеспечивая постоянный контроль процесса управления. Один модуль является основным, а остальные – резервными, активирующимися в случае отказа

основного. Пассивное резервирование предполагает устройство системы таким образом, что один процессорный модуль выполняет прикладную программу, а резервные находятся в состоянии ожидания. При возникновении неисправности основного процессорного модуля, переключение на резервный процессорный модуль осуществляется вручную или автоматически.

Процедура резервирования включает определение неисправности элемента оборудования, автоматическую замену неисправного оборудования резервным элементом, автоматическую передачу резервному элементу оборудования функций основного элемента [5].

Процедура резервирования должна занимать настолько малое количество времени, чтобы замена основного элемента на резервный не вызывала изменений в функционировании объекта критической инфраструктуры. Такое резервирование называется «бесшовным».

Резервирование процессорных модулей включает в себя выполнение в ПЛК процедуры анализа корректности функционирования процессорного модуля.

Если происходит сбой процессорного модуля, который не допускает дальнейшей эксплуатации, то управление передаётся резервному модулю, который должен выполнять в полном объёме функции основного модуля. Передача управления от основного процессорного модуля к резервному модулю должна происходить так, чтобы алгоритм управления объектом критической инфраструктуры не нарушался. В частности, требуется, чтобы на момент переключения данные, необходимые для функционирования управляющей программы на резервном модуле, совпадали с такими данными на основном модуле. На основном и на резервном процессорном модуле должна выполняться одна и та же прикладная программа

3. Аппаратная конфигурация ПЛК с резервированием

Для обеспечения резервирования в рамках рассматриваемой реализации процессорные модули должны быть соединены напрямую друг с другом посредством локальной сети Ethernet.

Основной и резервный процессорные модули подключены к одному набору модулей ввода-вывода. Для связи с модулями ввода-вывода на аппаратном уровне используется шина RS-485, а на программном уровне протокол Modbus RTU. Этот протокол предполагает только одно активное устройство [6], поэтому резервный процессорный модуль не выполняет опросы модулей ввода-вывода.

Помимо модулей ввода-вывода ПЛК может общаться с периферийными устройствами по сети Ethernet посредством промышленных протоколов связи через первый порт Ethernet.

Аппаратная конфигурация ПЛК с резервированием представлена на Рис. 1.

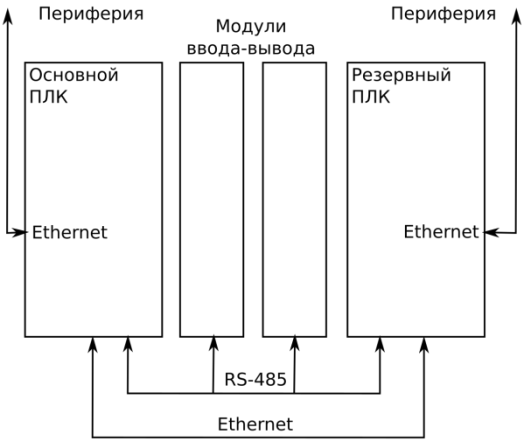

Рис. 1. Аппаратная конфигурация ПЛК

4. Алгоритм функционирования ПЛК с резервированием

Программа, написанная на языке МЭК, представляет собой набор задач, объединённых в один ресурс (или в несколько ресурсов). Задачи циклически выполняются, каждая со своим периодом. Задача имеет входные переменные, выходные переменные и промежуточные переменные, которые можно использовать и как входные и как выходные. Кроме того, ресурс может иметь глобальные переменные, доступные образующим ресурс задачам. Значения глобальных переменных сохраняются между циклами выполнения этих задач [7].

Результат выполнения каждой задачи на очередном цикле полностью определяется набором значений входных переменных задачи и набором значений глобальных переменных ресурса.

Поэтому для дублирования вычислительного процесса на резервном (запасном) процессорном модуле достаточно обеспечить совпадение значений входных и глобальных переменных на основном и на резервном модулях.

Для реализации алгоритма резервирования на резервный модуль передаются актуальные значения всех переменных из исходных текстов, написанных на языках МЭК. Сразу после переключения управления с текущего основного процессорного модуля на резервный, на резервном модуле начинает выполняться

управляющая программа с актуальными текущими значениями всех переменных.

В силу изложенного результат выполнения управляющей программы будет такой же, как и на основном модуле (при условии его штатной работы).

Таким образом, на текущем основном модуле программа выполняет основной цикл управления (опрос данных, выполнение прикладного алгоритма, вывод данных) и последующую передачу значений входных и глобальных переменных на резервный модуль.

Программа, выполняемая на резервном модуле в цикле, определяет работоспособность основного модуля и в случае обнаружения некорректной работы основного модуля, запускает прикладную программу, которая берет на себя функции управления.

Ещё одна функция резервного модуля состоит в том, чтобы перевести бывший основной модуль в состояние резервного, для того чтобы этот модуль гарантированно перестал направлять управляющие воздействия модулям ввода-вывода.

Алгоритм функционирования ПЛК с резервированием основан на постоянной синхронизации данных между основным и резервным процессорными модулями и контроле работоспособности основного процессорного модуля.

Для передачи данных между основным и резервным процессорными модулями используется протокол связи, использующий спецификацию TCP. Установка соединения между процессорными модулями осуществляется по их индивидуальным IP-адресам. На основном и на резервном процессорном модуле выполняется одна и та же прикладная программа, количество и типы переменных прикладной программы определены статично и не меняются во время выполнения программы. Это позволяет заранее определить набор прикладных переменных и последовательно передавать их в определенном порядке. При этом целостность и порядок следования данных обеспечивается на уровне протокола TCP [8].

Алгоритм функционирования прикладной программы ПЛК состоит из трех циклически повторяемых этапов:

- опрос значений модулей ввода
- выполнение алгоритма прикладной программы
- отправка значений в модули вывода.

Для резервирования после трех основных этапов добавляется четвертый этап: синхронизация текущего состояния переменных между основным и резервный процессорными модулями. При этом на основном процессорном модуле происходит отправка значений переменных на резервный процессорный модуль, а при выполнении на резервном процессорном модуле получение значений переменных от основного процессорного модуля.

В ходе выполнения прикладной программы, основной процессорный модуль выполняет все четыре этапа алгоритма, а резервный процессорный модуль выполняет только четвертый этап и находится в состоянии ожидания, для того чтобы приступить к выполнению трех основных этапов при смене ролей.

При старте процессорных модулей происходит автоматическое определение основного и резервного процессорного модуля. В связи с тем, что время инициализации программы ПЛК не является константным (в масштабе микросекунд), это значение используется для инициализации генератора псевдослучайных чисел.

В прикладной программе указаны IP адреса портов Ethernet основного и резервного процессорного модуля. При старте процессорного модуля определение роли (основной/резервный) выполняется следующим образом:

- Процессорный модуль назначает себе случайный IP адрес из заданной в конфигурации подсети (адресом является адрес основного процессорного модуля, у которого младший октет заменен случайным числом, генерация случайного числа происходит на основе времени загрузки)
- происходит поиск остальных устройств в локальной сети второго порта Ethernet
- если обнаружено устройство с адресом основного процессорного модуля, текущий процессорный модуль устанавливает себе сетевые адреса резервного процессорного модуля (на первый и второй порт Ethernet)
- если обнаружено устройство с адресом резервного процессорного модуля, текущий процессорный модуль устанавливает себе сетевые адреса основного устройства
- если устройств с адресами из конфигурации не обнаружено, то процессорный модуль, у которого последний октет адреса второго порта Ethernet больше, становится основным и устанавливает себе сетевые адреса основного процессорного модуля
- если устройств с адресами из конфигурации не обнаружено, то процессорный модуль, у которого последний октет адреса второго порта Ethernet меньше, становится резервным и устанавливает себе сетевые адреса резервного

процессорного модуля

- Процессорный модуль устанавливает подключение по адресу другого процессорного модуля (основного/резервного), запрашивает его роль (основной/резервный) и принимает себе роль отличную от второго процессорного модуля (резервный/основной).

При таком алгоритме резервирования сетевые адреса процессорного модуля не связаны однозначно с его ролью, то есть при смене роли не должна происходить смена сетевых адресов.

Смена роли происходит при обрыве соединения или отсутствии данных на резервном процессорном модуле более трех циклов программы, при этом резервный процессорный модуль сначала пытается повторно установить соединение с основным и в случае неудачи меняет свои роль на основную.

При выходе из строя одного из процессорных модулей аппаратура позволяет произвести его горячую замену. Условием возможности замены является то, что прикладная программа на вновь устанавливаемом процессорном модуле, а также на основном и резервном процессорных модулях одна и та же.

При этом на вновь устанавливаемом процессорном модуле будет автоматически выбрана роль резервного процессорного модуля, а сетевые адреса будут отличаться от активного процессорного модуля.

5. Особенности программной реализации

Специфика программной реализации состоит в том, что управляющая программа, написанная на одной из языков МЭК, подвергается промежуточной трансляции, которая переводит исходную программу на язык программирования Си.

При трансляции программы, написанной на языках МЭК в Си код, происходит преобразование объектов языков МЭК в конструкции языка Си. При этом простые типы преобразуются в простые типы языка Си. В то же время программы и функциональные блоки, написанные на языках МЭК, преобразуются в функции и структуры Си, содержащие все переменные программы или функционального блока [9].

Транслятор не осуществляет неявное создание переменных, сохраняющих свои значения при выполнении прикладной программы, помимо переменных, явно определенных в программе на языках МЭК.

Для осуществления бесшовного резервирования при смене роли резервного процессорного модуля на основную, необходимо синхронизировать значения всех переменных, сохраняющих свое значение между итерациями цикла функционирования прикладной программы ПЛК. Таким образом, необходимо синхронизировать значения всех глобальных в терминологии МЭК переменных, переменных программ и функциональных блоков.

Для синхронизации стандартных типов достаточно скопировать содержимое участков памяти, в которых хранятся значения переменных прикладной программы. Несколько сложнее обстоит дело с пользовательскими программами и функциональными блоками.

Согласно стандарту МЭК 61131-3-2016 переменные пользовательского функционального блока могут иметь тип "внешний", а переменные программы могут как иметь тип внешний, так и соответствовать физическим входам или выходам. В этих случаях такие переменные в полученном Си коде преобразуются в указатели на глобальные переменные или переменные, содержащие значение физического входа или выхода, соответственно. Аналогично преобразуются глобальные переменные, соответствующие физическим входам или выходам. Значения таких переменных-указателей передавать на резервный блок не требуется, и более того во время выполнения программы на разных процессорных модулях данные адреса не совпадают, поэтому копирование значений указателей может привести к ошибке сегментирования в программе при активации резервного процессорного модуля.

Для решения задачи синхронизации переменных пользовательских программ и функциональных блоков при трансляции программы из языков МЭК в Си код дополнительно добавлено формирование вспомогательных элементов:

- массив структур содержащих указатель на переменную, требующую синхронизацию, и размер переменной в памяти;
- массив указателей на переменные-указатели.

Данные массивы позволяют осуществить передачу в резервный процессорный модуль содержимого участков памяти, в которых хранятся значения переменных. Для передачи достаточно в цикле обойти все указатели и отправить фрагмент памяти указанного размера для каждой переменной. При приёме необходимо выполнить следующее:

- скопировать значения локальных указателей переменных во временный массив;

- принять весь набор переменных, полученных от основного модуля, во временный буфер;
- после успешного приема резервным модулем всех переменных, скопировать полученные значения из временного буфера в участки памяти переменных прикладной программы;
- восстановить значения ранее скопированных локальных указателей из временного массива.

Количество и типы переменных не меняются во время выполнения программы, поэтому размер буфера также не изменяется, и этот размер возможно вычислить при старте программы. В случае если не удается получить буфер целиком, заполнение значений переменных не выполняется.

При смене роли резервный процессорный модуль начинает выполнение алгоритма прикладной программы, при этом выполнение программы продолжается из состояния, отличающегося не более чем на один цикл прикладной программы. Под состоянием в данном случае предполагается набор значений всех переменных программы. В связи с тем, что синхронизация значений переменных осуществляется после выполнения основных этапов цикла прикладной программы, не представляется возможным гарантировать синхронизацию значений всех необходимых переменных между двумя процессорными модулями с разницей менее чем в один цикл.

Передача значений между процессорными модулями не является атомарной операцией. В случае отказа процессорного модуля во время выполнения отправки значений переменных, резервный модуль не получит полный набор переменных, размер полученного буфера будет меньше ожидаемого. Поскольку размер буфера фиксирован и не меняется во время выполнения программы, получение меньшего количества данных будет критерием выхода из строя основного процессорного модуля. В таком случае резервному процессорному модулю придется продолжать выполнение программы с набором переменных предыдущего цикла прикладной программы.

6. Заключение

В ходе работы над реализацией процедуры резервирования процессорных модулей для Багет-ПЛК1 был создан алгоритм и разработана соответствующая программная реализация определения состояния основного процессорного модуля и автоматического изменения роли резервного модуля на основную.

Основной процессорный модуль отправляет данные на резервный процессорный модуль каждую итерацию цикла прикладной программы, благодаря чему достигается идентичность состояния процессорных модулей обеспечивается бесшовное резервирование.

Данная реализация процедуры резервирования предназначена для прикладных программ, написанных на языках МЭК, разработанных в среде разработки СРиО.

Публикация выполнена в рамках государственного задания НИЦ «Курчатовский институт» - НИИСИ по теме FNEF-2024-0001.

An approach to implementing a redundancy algorithm

Y. A. Zotov, D. V. Yarikov

Abstract. The article discusses methods for implementing redundancy of processor modules for the programmable logic controller Baget-PLC1. A redundancy algorithm with data synchronization between the main and backup modules via Ethernet is proposed, ensuring seamless switching in case of main module failure. The specifics of software implementation based on the translation of application programs from IEC 61131-3 languages into C code are described, as well as the mechanism for automatic role determination of modules and system recovery. The solution is intended for use in control systems of critical infrastructure facilities.

Keywords: redundancy, processor modules, Baget-PLC1, data synchronization, seamless switching

Литература

1. Сердин, О. В. Многоцелевой программируемый логический контроллер «Багет-ПЛК1»: патент на полезную модель № 211983 Рос. Федерация: G06F 9/00 / О. В. Сердин, М. А. Голяков, А. В. Бакалдин, С. Е. Серяков, М. А. Чушев; патентообладатель Федеральное государственное учреждение «Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук». — № 2021129783 ; заявл. 12.10.2021 ; опубл. 30.06.2022.

2. Годунов А.Н., Солдатов В.А. Операционные системы семейства Багет (сходство, отличия и перспективы) // Программирование, 2014, № 5, с. 68-76

3. ГОСТ ГОСТ Р МЭК 61131-3-2016 «Контроллеры программируемые. Часть 3. Языки программирования»

4. Черкесов. Г.Н. Надежность аппаратно-программных комплексов/ Учебное пособие. – СПб.: Питер, 2005. –479 с.

5. Энциклопедия АСУ ТП. 8. Аппаратное резервирование. 8.1. Основные понятия и определения. URL: https://www.reallab.ru/bookasutp/8-apparatnoe-rezervirovanie/8-1-osnovnie-ponyatiya-i-opredeleniya // Энциклопедия АСУ ТП (дата обращения: 08.12.2025).

6. Просто о Modbus RTU с подробным описанием и примерами. URL: https://ipc2u.ru/articles/prostye-resheniya/modbus-rtu // IPC2U — Промышленные компьютеры, ПЛК, системы связи (дата обращения: 10.12.2025).

7. Энциклопедия АСУ ТП. 9.3. Системы программирования на языках МЭК 61131-3. URL: https://www.reallab.ru/bookasutp/9-programmnoe-obespechenie/9-3-sistemi-programmirovaniya-mek-61131-3/ // Энциклопедия АСУ ТП (дата обращения: 09.12.2025).

8. Рубашенков Антон Михайлович, Бобров Андрей Виорелович Протокол tcp // Наука, техника и образование. 2018. №11 (52). URL: https://cyberleninka.ru/article/n/protokol-tcp (дата обращения: 16.12.2025).

9. Documentation | beremiz.org. URL: https://beremiz.org/doc // beremiz.org (дата обращения: 12.12.2025).

Наименование: сетевой рецензируемый научный журнал «Труды НИИСИ»
ISSN: 3033-6422
Сведения о переименовании: до 2025 года журнал издавался в печатном виде с названием «Труды НИИСИ РАН», ISSN 2225-7349
Журнал основан: 2011 г.
Периодичность: 4 раза в год
Учредитель и издатель: НИЦ «Курчатовский институт» — НИИСИ
Главный редактор: Бетелин Владимир Борисович, д. ф.-м. н., профессор, академик РАН
Адрес учредителя и издателя: 117218, Москва, Нахимовский проспект, д.36, к.1
Адрес редакции: 117218, Москва, Нахимовский проспект, д.36, к.1
Контакты: Тел.: +7 (925) 924-83-46; muranov@niisi.msk.ru
Подписка: Электронная версия журнала находится в свободном доступе на сайте журнала, а также в базах данных открытого доступа

Title: *SRISA Proceedings* online peer-reviewed journal
ISSN: 3033-6422
Former title: until 2025, the journal was published in print under the name *Trudy NIISI RAN* (Proceedings of SRISA, Russian Academy of Sciences), ISSN 2225-7349
Published since: 2011
Publication frequency: Quarterly
Founder and publisher: NRC "Kurchatov Institute" - SRISA
Chief Editor: Vladimir B. Betelin, Doctor of Science (Phys&Math), Prof., member of the Russian Academy of Sciences,
Address of the founder and publisher: 117218, Moscow, Nakhimovsky avenue, 36, bldg. 1
Address of the editorial office: 117218, Moscow, Nakhimovsky avenue, 36, bldg. 1
Contacts: Tel.: +7 (925) 924-83-46; muranov@niisi.msk.ru
Subscription: The electronic version of the journal is freely available on the journal's website, as well as in open access databases

Подписано в печать 23.12.2025 г.
Формат 60x90/8
Печать цифровая. Условных печатных листов – 7,3