КОНЦЕПЦИЯ ПЛАТФОРМЫ ГЕНЕРАЦИИ ПРИКЛАДНЫХ СИСТЕМ УПРАВЛЕНИЯ

Р. Д. Гимранов[1,а], **В. А. Галкин**[2,3,б], **Т. В. Гавриленко**[2,3,в], **Д. А. Моргун**[2,г], **М. Т. Гавриленко**[2,3,д], **М. Е. Амелин**[2,3,е], **С. А. Батуро**[2,3,жс], **А. И. Грюнталь**[4,з]

[1] *ПАО «Сургутнефтегаз», г. Сургут, Российская Федерация*

[2] *Сургутский филиал федерального государственного бюджетного учреждения «Федеральный научный центр Научно-исследовательский институт системных исследований Национального исследовательского центра «Курчатовский институт», г. Сургут, Российская Федерация*

[3] *Сургутский государственный университет, г. Сургут, Российская Федерация*

[4] *Федеральное государственное бюджетное учреждение «Федеральный научный центр Научно-исследовательский институт системных исследований Национального исследовательского центра «Курчатовский институт», г. Москва, Российская Федерация*

[а] *gimranov_rd@mail.ru,* [б] *val-gal@yandex.ru,* [в] *taras_gavrilenko@office.niisi.tech,*

[г] ✍ *morgun_da@office.niisi.tech,* [д] *gavrilenko.michail.t@office.niisi.tech,*

[е] *amelin_me@office.niisi.tech,* [жс] *baturo_sa@office.niisi.tech,* [з] *grntl@niisi.ras.ru*

Аннотация: в статье представлена концепция low-code/no-code платформы автоматизации построения прикладных систем управления технологическими процессами нефтегазовой отрасли на основе онтологического моделирования. Проведен сравнительный анализ методологий (Methontology, On-To-Knowledge, подход Грубера, BORO) и языков описания онтологий (RDF, RDFS, OWL, KIF, CycL, Gellish) с обоснованием выбора OWL и инструментальной среды Protégé. Описана пятиуровневая иерархическая архитектура платформы, охватывающая физические устройства, контроллеры, обработку данных, визуализацию и бизнес-логику. Рассмотрен обобщенный алгоритм функционирования платформы: последовательная трансформация онтологических моделей через BPMN- и DFD-диаграммы к метаописанию проекта и итоговому программному коду. Сформулированы базовые требования к концепции, включая поддержку отечественных низкопроизводительных вычислительных систем.

Ключевые слова: онтология, OWL, Protégé, low-code платформа, АСУ ТП, BPMN, DFD, нефтегазовая отрасль, иерархическая архитектура, управление метаданными, технологический суверенитет.

Благодарности: работа выполнена в рамках НИОКР «Разработка прототипа программной платформы генерации прикладных систем управления» № 1814 от 28.12.2024.

Для цитирования: Гимранов Р. Д., Галкин В. А., Гавриленко Т. В., Моргун Д. А., Гавриленко М. Т., Амелин М. Е., Батуро С. А., Грюнталь А. И. Концепция платформы генерации прикладных систем управления. *Успехи кибернетики.* 2026;7(2):41–51.

Поступила в редакцию: 03.06.2026. В окончательном варианте: 21.06.2026.

THE CONCEPT OF A PLATFORM FOR APPLICATION-CENTRIC CONTROL SYSTEM SYNTHESIS

R. D. Gimranov[1,a], **V. A. Galkin**[2,3,b], **T. V. Gavrilenko**[2,3,c], **D. A. Morgun**[2,d], **M. T. Gavrilenko**[2,3,e], **M. E. Amelin**[2,3,f], **S. A. Baturo**[2,3,g], **A. I. Gryuntal**[4,h]

[1] *"Surgutneftegas" Public Joint Stock Company, Surgut, Russian Federation*

[2] *Surgut Branch of Scientific Research Institute for System Analysis of the National Research Centre "Kurchatov Institute", Surgut, Russian Federation*

[3] *Surgut State University, Surgut, Russian Federation*

[4] *Scientific Research Institute for System Analysis of the National Research Centre "Kurchatov Institute", Moscow, Russian Federation*

[a] *gimranov_rd@mail.ru,* [b] *val-gal@yandex.ru,* [c] *taras_gavrilenko@office.niisi.tech,*

[d] ✍ *morgun_da@office.niisi.tech,* [e] *gavrilenko.michail.t@office.niisi.tech,*

[f] *amelin_me@office.niisi.tech,* [g] *baturo_sa@office.niisi.tech,* [h] *grntl@niisi.ras.ru*

Abstract: we proposed the concept of a low-code/no-code platform for automating the construction of process control systems in the oil and gas industry based on ontological modeling. We carried

out a comparative analysis of ontology engineering methodologies (Methodology, On-To-Knowledge, Gruber's ontology design principles, BORO) and ontology description languages (RDF, RDFS, OWL, KIF, CycL, Gellish) to justify the selection of OWL and the Protégé environment as the implementation tool. We described a five-level hierarchical architecture of the platform, which covers physical devices, controllers, data processing, visualization, and business logic. We proposed a generalized algorithm for platform operation, which ensures the sequential transformation of ontological models through BPMN and DFD diagrams into a project meta-description and final program code. We formulated the basic requirements of the concept, including support for domestically produced low-performance computing systems.

Keywords: ontology, OWL, Protégé, low-code platform, process control system, BPMN, DFD, oil and gas industry, hierarchical architecture, metadata management, technological sovereignty.

Acknowledgements: this study is a part of the "Development of a Prototype Platform for Application-Centric Control System Synthesis" R&D project No. 1814 dated 28.12.2024.

Cite this article: Gimranov R. D., Galkin V. A., Gavrilenko T. V., Morgun D. A., Gavrilenko M. T., Amelin M. E., Baturo S. A., Gryuntal A. I. The Concept of a Platform for Application-Centric Control System Synthesis. *Russian Journal of Cybernetics*. 2026;7(2):41–51.

Original article submitted: 03.06.2026. *Revision submitted: 21.06.2026.*

Введение

Современные автоматизированные системы управления технологическими процессами (АСУ ТП) нефтегазовой отрасли сталкиваются с нарастающим противоречием: с одной стороны, возрастает сложность технологических объектов и разнородность задействованного оборудования; с другой — критически важны скорость создания новых приложений, их адаптация к меняющимся производственным условиям и соответствие требованиям технологического суверенитета. Традиционные подходы к разработке ПО не справляются с этим вызовом: создание каждого нового прикладного решения требует значительных ресурсов, а поддержка унаследованных систем становится все более затратной.

Ответом на данный вызов служит концепция low-code/no-code платформы, ядром которой является онтологическое моделирование предметной области. Онтология — формализованная модель знаний об объектах системы, их свойствах и взаимосвязях — позволяет создать единое смысловое пространство, разделяемое как людьми (операторами, архитекторами решений), так и программными агентами. На основе онтологии строятся модели бизнес-процессов (BPMN) и потоков данных (DFD), которые в свою очередь служат источником для автоматической генерации исполняемого кода.

Анализ существующих подходов к определению онтологии и выбор методологии

В контексте АСУ ТП онтология решает принципиальную задачу: обеспечивает не просто передачу данных между компонентами системы, но передачу их смысла и контекста. Датчики, исполнительные механизмы, контроллеры, бизнес-объекты предприятия — все это должно быть описано в едином концептуальном пространстве с явными отношениями между элементами. Именно наличие такого пространства отличает «умную» платформу, способную генерировать корректный программный код, от совокупности несвязанных компонентов.

Уровни онтологий и пирамида абстракций

Онтологии принято разграничивать по степени общности. Верхнеуровневые онтологии (SUMO, DOLCE) задают фундаментальные категории — объект, процесс, свойство, пространство, время — без привязки к какой-либо предметной области; они служат «каркасом», к которому «крепятся» более специализированные концепции. Доменные (отраслевые) онтологии фиксируют специфику конкретной отрасли: в нефтегазовом секторе это типы оборудования (ТРК, уровнемеры, контроллеры), технологические параметры (давление, расход, температура), стандарты безопасности. Прикладные онтологии — наиболее конкретный уровень: они описывают конкретное предприятие, его объекты и экземпляры оборудования. Рисунок 1 иллюстрирует эту трехуровневую пирамиду.



Рис. 1. *Пирамида уровней онтологий в контексте АСУ ТП нефтегазовой отрасли*

Важным строительным блоком любой онтологии является таксономия — иерархическое (is-a) описание классов и подклассов. В АСУ ТП таксономия структурирует оборудование, персонал, процессы и типы данных, обеспечивая единую «шкалу» наименований при интеграции разнородных подсистем. Например, в рамках разработанной онтологии для АЗС выстраивается ветвь: оборудование → технологическое оборудование → ТРК; оборудование → торгово-кассовое оборудование → банковские терминалы. Такая структура обеспечивает наследование свойств и упрощает автоматическую обработку модели.

Сравнительный анализ методологий

Для создания онтологии необходима методология — упорядоченный процесс сбора знаний, формирования таксономии и спецификации отношений. Проведенный анализ охватил четыре ведущие методологии и подход с применением UML.

Таблица 1

Сравнение методологий построения онтологий в контексте АСУ ТП

Методология	**Подход**	**Область применения**	**Ограничения**
Methontology	Формальная, поэтапная	Промышленные системы	Высокая трудоемкость
On-To-Knowledge	Онтологии из источников знаний	Корпоративные порталы	Зависимость от экспертов
Подход Грубера	Минималистичная, декларативная	ИИ-системы, Web	Ограниченная детализация
BORO	Объектно-ориентированная	Бизнес-процессы	Узкая предметная область
UML-онтологии	Нотация UML + семантика	Инженерные системы	Нет аксиоматики OWL

Methontology предлагает наиболее формализованный и полный процесс: концептуализация, формализация, интеграция и оценка онтологии выполняются на отдельных этапах. Именно эта формальность делает методологию предпочтительной для промышленных систем, где требуется воспроизводимость и прослеживаемость решений. On-To-Knowledge ориентирована на извлечение знаний из неструктурированных источников и корпоративных документов, что ограничивает ее применимость в инженерном контексте с четкими техническими спецификациями. Подход Грубера принципиально минималистичен: онтология должна быть наименьшей достаточной формализацией, что удобно для ИИ-систем, но недостаточно для полного описания АСУ ТП. BORO ориентирована на бизнес-процессы и объекты организации, однако не охватывает физический уровень технологического оборудования. Подход на основе UML привлекателен знакомостью нотации для инженеров, однако UML лишен встроенной аксиоматики, что затрудняет логический вывод и машинную обработку модели.

По совокупности критериев — полнота процесса, поддержка логического вывода, применимость к инженерным системам и наличие зрелой инструментальной поддержки — для разрабатываемой платформы принята методология Methontology.

Выбор языка описания онтологий

Параллельно с выбором методологии необходимо определить язык формального описания онтологии. Проведенный обзор охватил шесть актуальных языков.

Таблица 2

Сравнение языков описания онтологий

Язык	**Организация**	**Модель**	**Применение**	**Ограничения**
RDF	W3C	Граф-модель	Широкое, веб-данные	Слабая аксиоматика
RDFS	W3C	Классы и свойства	Схемы данных	Без логики
OWL	W3C	DL-логика	АСУ ТП, ИИ, инженерия	Высокая сложность
KIF	IEEE	Предикатная логика	ИИ, обмен знаниями	Устаревший
CycL	Cycorp	Формальная логика	Общие знания	Закрытый стандарт
Gellish	Gellish.net	ER-модель	Инженерные справочники	Узкая область

RDF задает базовую граф-модель «субъект–предикат–объект» и является фундаментом для остальных стандартов W3C. RDFS добавляет иерархию классов и свойств, но не поддерживает логический вывод. OWL (Web Ontology Language) строится на Description Logic: позволяет задавать аксиомы, эквивалентность классов, ограничения мощности и транзитивность отношений — все это критически важно для автоматической обработки онтологии. KIF, несмотря на выразительность предикатной логики первого порядка, является устаревшим стандартом без широкой поддержки. CycL отличается мощью общей базы знаний, но закрытый характер делает его непригодным для промышленных проектов с требованием технологической независимости. Gellish удобен для инженерных справочников, но слишком узкоспециализирован.

Выбор сделан в пользу OWL (Web Ontology Language) как языка, обеспечивающего наибольшую выразительность при наличии открытого стандарта W3C, широкой инструментальной поддержки и возможности машинного вывода над моделью. Для работы с OWL выбрана среда Protégé (Stanford University) — зрелый OpenSource-инструмент с поддержкой рассуждателей (Hermit, Pellet), экспорта в форматы RDF/XML и Turtle, а также визуального редактора онтологических моделей. Все рассмотренные инструменты относятся к категории OpenSource и могут применяться на вычислительных системах различных архитектур, включая отечественные.

Иерархическая архитектура платформы

Платформа построена на основе пятиуровневой иерархической модели (рис. 2), отражающей реальную структуру промышленного предприятия — от физических датчиков и исполнительных механизмов до корпоративных бизнес-процессов. Каждый уровень обладает собственной функциональной ответственностью, четко определенными интерфейсами взаимодействия с соседними уровнями и соответствующим набором требований к безопасности.



Рис. 2. *Пятиуровневая иерархическая архитектура платформы АСУ ТП*

В основании структуры находится уровень физических устройств и оборудования: датчики температуры, давления и расхода; исполнительные механизмы; полевые шины (Modbus, PROFINET, EtherNet/IP). Непосредственно над ним располагается уровень контроллеров и первичной автоматизации: ПЛК, РСУ, системы сбора данных реального времени, реализующие алгоритмы управления и обеспечивающие детерминированное время отклика. На третьем уровне располагаются SCADA-системы и HMI, предоставляющие операторам адаптивный интерфейс мониторинга и управления. Четвертый уровень — обработка и интеграция данных — агрегирует разрозненные потоки, обеспечивает нормализацию, фильтрацию и маршрутизацию данных через шину интеграции, поддерживающую OPC UA, MQTT и AMQP. Верхний, пятый уровень связывает технологические данные с ERP-системами, аналитическими сервисами и бизнес-логикой предприятия.

Такая структура переносится на онтологическую модель, то есть каждому уровню соответствует своя ветвь таксономии, а взаимодействия между уровнями описываются объектными свойствами OWL. Благодаря этому обеспечивается согласованность программной архитектуры с физической структурой АСУ ТП.

Организация потоков данных в иерархической модели предполагает поддержку как вертикальных, так и горизонтальных потоков. Механизмы расширения иерархической структуры реализуются через документированный API, который позволяет добавлять пользовательские коннекторы к оборудованию и специализированные отраслевые модули без изменения ядра платформы. Жизненный цикл компонентов управляется через версионирование и автоматизированное развертывание с тестированием в изолированной среде.

Концепция платформы: базовые требования

Платформа представляет собой программный комплекс для формирования структуры взаимодействия оборудования и объектов предприятия на различных уровнях иерархии. Концепция строится на четырех базовых принципах, определяющих ее архитектурное ядро.

Таблица 3

Четыре базовых принципа концепции платформы

Иерархическая структура	**Low-code/ no-code**	**Управление метаданными**	**Система безопасности**
5 уровней: от физических устройств до бизнес-логики	Графические редакторы онтологий, BPMN, DFD без программирования	Менеджер тегов, словари параметров, каталог элементов, репозиторий сигналов	Многоуровневая защита по ГОСТ Р МЭК 62443, ролевая модель, MFA

Иерархическая структура задает вертикальную организацию платформы. Low-code/no-code концепция реализуется через графические редакторы: редактор онтологических моделей (Protégé), конструктор BPMN-диаграмм, редактор DFD-диаграмм. Принципиально, что все эти инструменты не требуют от архитектора решений навыков программирования — он работает с визуальными нотациями, а платформа автоматически обеспечивает их формализацию и согласованность. Система управления метаданными — менеджер тегов, словари параметров, каталог технологических элементов, репозиторий типов сигналов — обеспечивает единообразие представления данных и исключает терминологическую неоднозначность при интеграции разнородного оборудования. Система безопасности реализована по иерархическому принципу: физическая защита, защищенная загрузка контроллеров, обнаружение вторжений, MFA и ролевая модель доступа на уровне визуализации, управление цифровыми правами на уровне бизнес-логики.

Таблица 4

Требования к платформе

Группа требований	**Ключевые положения**	**Назначение**
Архитектурные	Микросервисы, модульность, масштабируемость, расширяемость через API	Независимое обновление компонентов
Технологические	Low-code/no-code редакторы, OPC UA, Modbus, REST API, MQTT, AMQP	Работа без программирования
Безопасности	Многоуровневая защита, ГОСТ Р МЭК 62443, ролевая модель, MFA	Защита критической инфраструктуры
Производительности	Реального времени, отказоустойчивость, работа офлайн, кластеризация	Непрерывность АСУ ТП
Интерфейса	Адаптивный UI, персонализация, соответствие UX промышленных систем	Удобство оператора
Жизненного цикла	Версионирование, аудит, автодеплой, тестирование в изолированной среде	Управляемое изменение

Особое значение имеет требование функционирования на отечественных низкопроизводительных вычислительных системах. Операции построения онтологий и интерпретации диаграмм не предполагают интенсивных вычислений в реальном времени, что делает платформу совместимой с процессорами среднего уровня производительности отечественного производства. Это соответствует принципам импортозамещения и технологического суверенитета в критически важной инфраструктуре топливно-энергетического комплекса.

Обобщенный алгоритм функционирования платформы

Диаграмма SADT

Для формализованного описания платформы разработана SADT-диаграмма в нотации IDEF0 (рис. 3). На вход подаются объекты и процессы предметной области, а также техническое задание. Основным участником является архитектор решений, от квалификации которого напрямую зависит качество конечного результата. В перспективе его роль частично может взять на себя агент большой языковой модели. Управляющими воздействиями служат номенклатуры построения онтологий, BPMN и DFD, требования к безопасности, нормативные документы и прецеденты. На выходе формируются: онтологическая модель, BPMN-диаграмма, DFD-диаграмма, структура программного обеспечения, структура взаимодействия объектов и готовое решение, включающее программный код и настройки среды.

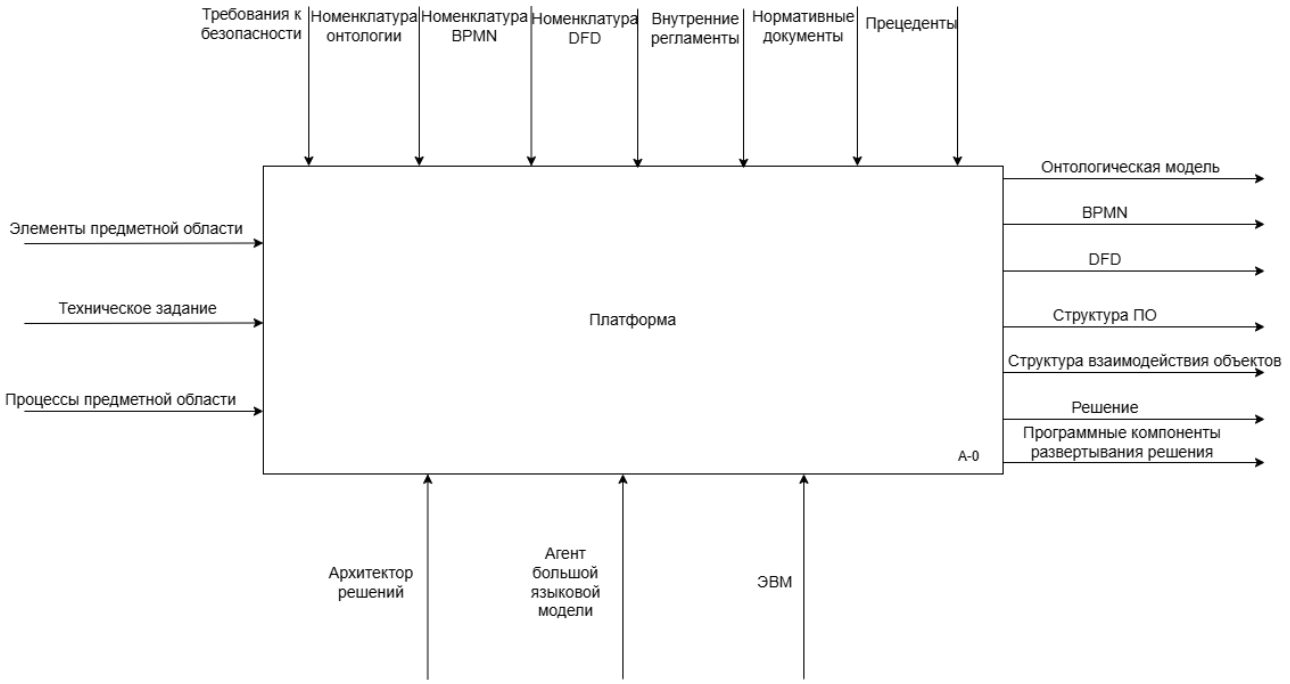


Рис. 3. *Диаграмма SADT*

Детализированная SADT-диаграмма (рис. 4) раскрывает 11 последовательных этапов функционирования платформы. На этапах 1–2 архитектор строит обобщенную и детализированную онтологические модели согласно номенклатуре онтологий. Этап 3 — проверка их согласованности интерпретатором; при выявлении противоречий выполняется возврат к этапу 1. На этапах 4–5 создается и верифицируется BPMN-диаграмма. Этап 6 — построение DFD-диаграммы. На этапах 7–10 интерпретатор объединяет все модели в машиночитаемое описание, на основе которого последовательно формируются: проект структуры ПО, структура взаимодействия объектов и готовое решение. Этап 11 — реализация решения в виде программных компонентов. Все модели и решения сохраняются в соответствующих базах данных с возможностью повторного использования.

Диаграмма DFD

Диаграмма DFD (рис. 5) отображает потоки данных между компонентами платформы. Верхний поток описывает трансформацию моделей: от обобщенной онтологической модели (формализованной в OWL) через детализированную онтологическую модель к BPMN-диаграмме и итоговой DFD. На каждом переходе выполняется раунд корректировки метаданных для обеспечения согласованности. Нижний поток ориентирован на практическую реализацию: интерпретатор преобразует совокупность диаграмм и моделей в машиночитаемое описание, на основе которого формируются проект структуры ПО, структура взаимодействия объектов и готовое решение, загружаемое в целевую инфраструктуру.

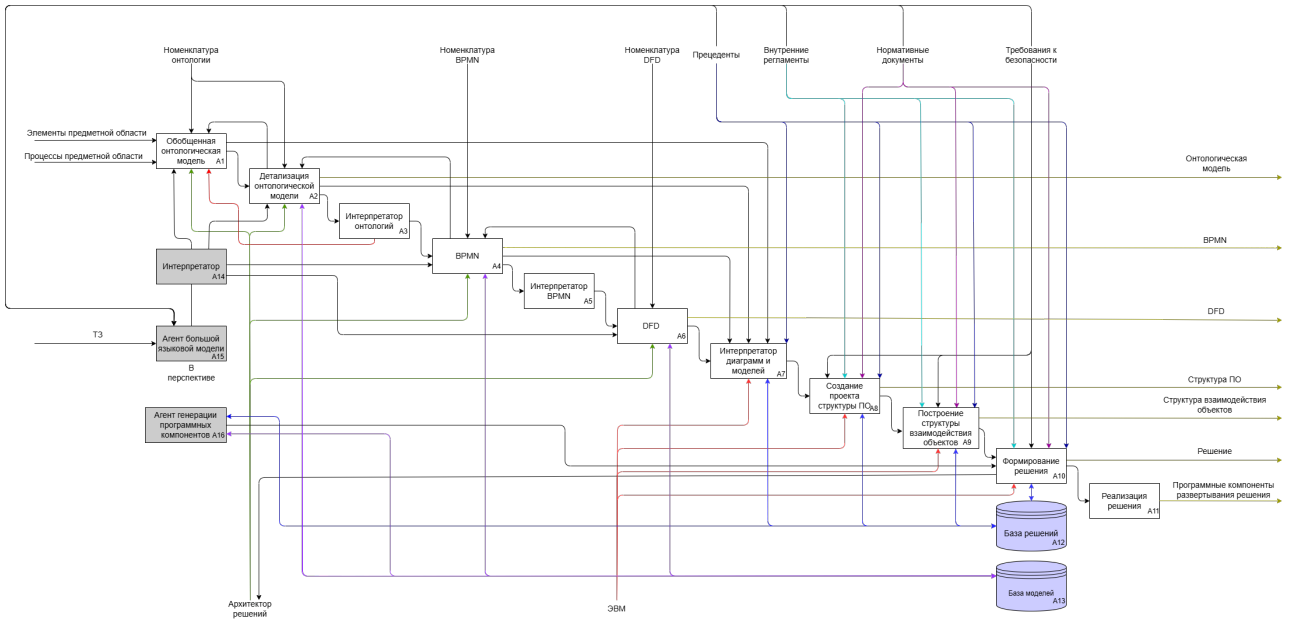


Рис. 4. *Детализированная диаграмма SADT*

Централизованными хранилищами служат база моделей и база решений. Номенклатура стандартов (онтологии, BPMN, DFD) выступает нормативной базой для всех процессов. В перспективе роль архитектора решений будет дополнена агентом большой языковой модели и агентом генерации программных компонентов, что обеспечит более высокий уровень автоматизации.

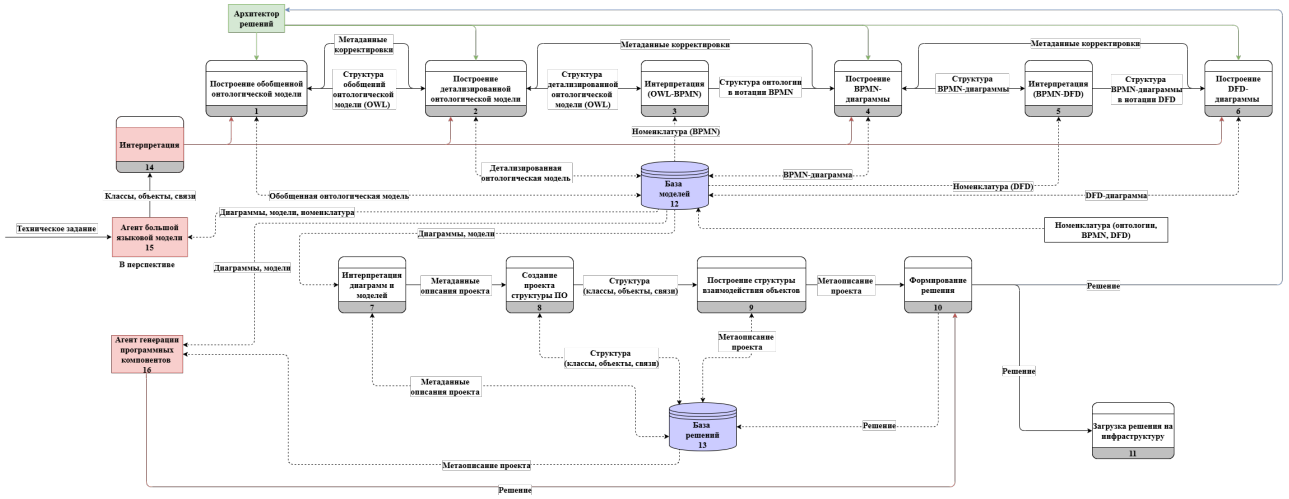


Рис. 5. *Диаграмма DFD*

Обобщенный алгоритм платформы

Обобщенный алгоритм функционирования платформы представлен в виде блок-схемы (рис. 6) и включает следующие шаги:

- построение обобщенной онтологической модели на основе объектов и процессов предметной области;
- построение детализированной онтологической модели на основе обобщенной;
- проверка согласованности моделей интерпретатором, при наличии противоречий — возврат к шагу 1;
- трансляция онтологических моделей в BPMN-диаграмму с последующей корректировкой и повторной проверкой согласованности;
- построение DFD-диаграммы с аналогичной проверкой согласованности всех диаграмм;

- формирование машиночитаемого описания объектов и процессов интерпретатором диаграмм и моделей;
- создание проекта структуры программного обеспечения на основе полученного описания;
- построение структуры взаимодействия объектов;
- формирование готового решения и его реализация на предприятии.

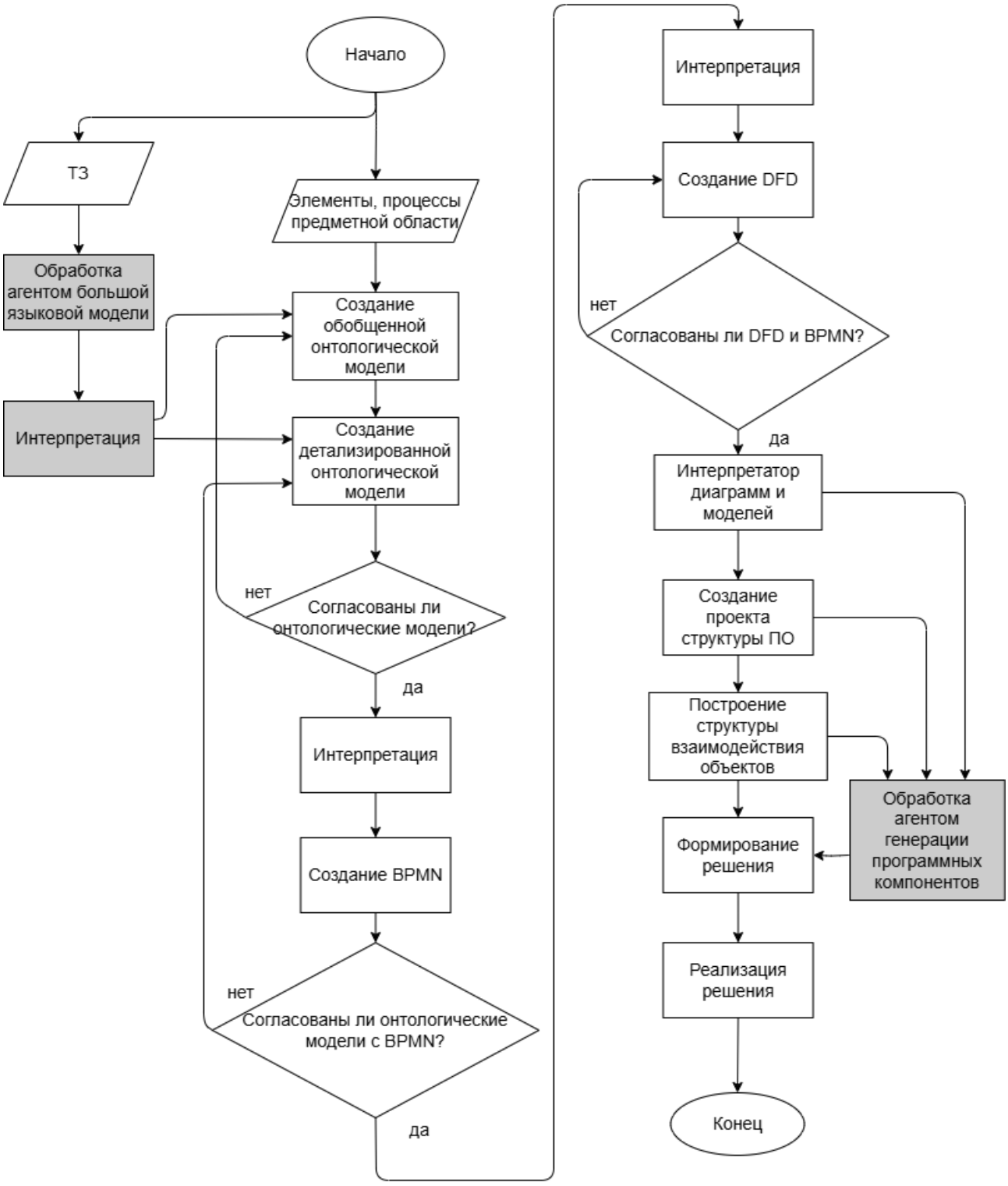


Рис. 6. *Обобщенный алгоритм платформы*

Роль архитектора решений

Ключевой ролью в функционировании платформы является архитектор решений. Работа

платформы условно делится на два глобальных этапа: создание проектных схем (онтологические модели, BPMN, DFD) и формирование решения (структура ПО, структура взаимодействия объектов, готовое решение, реализация). Первый этап — определяющий, поскольку качество итогового решения напрямую зависит от полноты и корректности разработанных моделей и диаграмм.

В задачи архитектора входят:

- глубокий анализ предметной области и выделение ключевых объектов и процессов;
- построение обобщенной и детализированной онтологических моделей;
- создание BPMN-диаграммы без нарушения согласованности с онтологиями;
- построение DFD-диаграммы с соблюдением согласованности с BPMN.

Таким образом, архитектор решений обеспечивает корректное отражение сущностей и их связей, процессов предметной области и потоков данных.

Разработанные SADT-диаграммы (общая и детализированная), DFD-диаграмма и блок-схема обобщенного алгоритма описывают полный цикл функционирования платформы. Центральной фигурой процесса является архитектор решений. Архитектура платформы структурно предусматривает возможность внедрения агентов на базе больших языковых моделей для частичной автоматизации этапов моделирования и кодогенерации. Реализуемые алгоритмы рассчитаны на применение в вычислительных системах различных архитектур, включая отечественные и низкопроизводительные.

Заключение

В статье представлена концепция low-code/no-code платформы автоматизации разработки прикладных систем управления для нефтегазовой отрасли, в основе которой лежит онтологическое моделирование. Проведенный сравнительный анализ методологий обосновал выбор Methontology как наиболее формального и применимого в инженерном контексте подхода. Из всех рассмотренных языков описания онтологий выбор сделан в пользу OWL с инструментальной средой Protégé, что обеспечивает стандартизацию, машинную обработку и выразительность модели при использовании открытого ПО.

Пятиуровневая иерархическая архитектура платформы обеспечивает систематическое покрытие всех уровней АСУ ТП — от физических устройств до бизнес-логики. Четыре базовых принципа концепции (иерархическая структура, low-code/no-code, управление метаданными, многоуровневая безопасность) формируют целостную архитектурную парадигму. Обобщенный алгоритм реализует сквозную цепочку трансформации: онтологические модели → BPMN → DFD → метаописание → исполняемый код, с механизмом итеративного согласования на каждом переходе.

Важным практическим результатом является подтверждение совместимости платформы с отечественными низкопроизводительными вычислительными системами, что определяет ее потенциал как импортозамещающего решения для критически важной инфраструктуры.

ЛИТЕРАТУРА

1. Barrasa J., Webber J. *Building Knowledge Graphs*. Sebastopol: O'Reilly Media; 2023. 421 с.
2. Неизвестный С. И. О применении таксономии в области информационных технологий. *Инновационные транспортные системы и технологии*. 2016;2(1):89–111.
3. Fernández-López M., Gómez-Pérez A., Juristo N. Methontology: From Ontological Art towards Ontological Engineering. *Proceedings of the AAAI-97 Spring Symposium Series*. Stanford; 1997:33–40.
4. Sure Y., Staab S., Studer R. On-to-Knowledge Methodology. *Handbook on Ontologies* / Staab S., Studer R. (eds.). Berlin; Heidelberg: Springer; 2004:117–132.
5. Верхотурова Ю. С. Онтология как модель представления знаний. *Вестник БГУ. Сер.: Философия*. 2012;15:32–37.
6. *Protégé: официальный сайт*. Режим доступа: https://protege.stanford.edu.
7. *Web Ontology Language (OWL)*. Режим доступа: https://www.w3.org/OWL.
8. *Business Process Model and Notation (BPMN): спецификация OMG*. Режим доступа: https://www.omg.org/spec/BPMN.

9. Новиков Ф. А., Иванов Д. Ю. *Моделирование на UML: теория, практика, видеокурс*. СПб.: Профессиональная литература; 2010. 640 с.
10. Марка Д. А., МакГоуэн К. *Методология структурного анализа и проектирования*. М.: МетаТехнология; 1993. 240 с.
11. Li Q., Chen Y. L. Data Flow Diagram. *Modeling and Analysis of Enterprise and Information Systems*. Berlin; Heidelberg: Springer; 2009:85–97. DOI: 10.1007/978-3-540-89556-5_4.
12. Rokis K., Kirikova M. Challenges of Low-Code/No-Code Software Development: A Literature Review. *Perspectives in Business Informatics Research. BIR 2022. Lecture Notes in Business Information Processing* / Nazaruka Ē., Sandkuhl K., Seigerroth U. (eds.). Cham: Springer; 2022;462:3–17. DOI: 10.1007/978-3-031-16947-2_1.
13. Филимонова А. А., Боос Г. О. Современные АСУ ТП. *Инновации в науке*. 2014;38:39–42.
14. *ГОСТ Р МЭК 62443-4-2-2019. Безопасность систем промышленной автоматизации и управления. Часть 4-2. Технические требования безопасности для компонентов IACS*. М.: Стандартинформ; 2020. 116 с.