Интернет-журнал «Отходы и ресурсы» https://resources.today
Russian Journal of Resources, Conservation and Recycling

2026, Том 13, № 2 / 2026, Vol. 13, Iss. 2 https://resources.today/issue-2-2026.html

URL статьи: https://resources.today/PDF/07INOR226.pdf

DOI: 10.15862/07INOR226 (https://doi.org/10.15862/07INOR226)

2.3.1. Системный анализ, управление и обработка информации, статистика (технические науки)

Ссылка для цитирования этой статьи:
Никулин, А. В. Формальная модель выбора архитектуры программного обеспечения на основе структурных метрик / А. В. Никулин // Отходы и ресурсы. — 2026. — Т. 13. — № 2. — URL: https://resources.today/PDF/07INOR226.pdf. DOI: 10.15862/07INOR226.

For citation:
Nikulin A.V. A formal model for selecting software architecture based on structural metrics. *Russian Journal of Resources, Conservation and Recycling*. 2026;13(2): 07INOR226. Available at: https://resources.today/PDF/07INOR226.pdf. DOI: 10.15862/07INOR226. (In Russ., abstract in Eng.).

УДК 004.415.2:004.41

Никулин Александр Валерьевич
ФГБОУ ВО «Елецкий государственный университет имени И. А. Бунина», Елец, Россия
Аспирант кафедры «Математического моделирования,
компьютерных технологий и информационной безопасности»
E-mail: avnikulin.niiaa@gmail.com
ORCID: https://orcid.org/0009-0005-8426-3629

Формальная модель выбора архитектуры программного обеспечения на основе структурных метрик

Аннотация. Представлена формальная модель пространства архитектурных решений для автоматизированного проектирования архитектуры программного обеспечения информационных систем. Работа направлена на устранение разрыва между предметным описанием системы и выбором архитектурного решения, который во многих проектах остается экспертным и недостаточно воспроизводимым. В статье предложено описывать архитектурный вариант как кортеж, включающий граф предметной области, множество ограниченных контекстов, отображение контекстов в архитектурные стили, стратегии управления данными, протоколы коммуникации, модели развертывания, каталог паттернов и набор ограничений корректности. Для сопоставления вариантов введена система из девяти структурных прокси-метрик, характеризующих связанность, связность, модулярность, гранулярность, автономность данных, сложность интерфейсов, коммуникационные накладные расходы, изоляцию отказов и независимость развертывания. Работоспособность модели проверена на демонстрационном корпусе из пяти предметных сценариев, включающих банковскую, торговую, медицинскую, логистическую и образовательную информационные системы. Показано, что предложенная модель позволяет автоматически формировать допустимые архитектурные варианты, выявлять нарушения ограничений и сравнивать решения по отдельным структурным свойствам. Результаты демонстрационного эксперимента показывают, что монолитная, сервисно-ориентированная и микросервисная реализации имеют разные сильные и слабые стороны, а гибридные решения могут занимать недоминируемое положение в пространстве альтернатив. Отдельно обозначены ограничения модели, связанные с экспертной калибровкой коэффициентов, упрощенным назначением протоколов, графовой аппроксимацией границ контекстов и необходимостью последующей проверки на индустриальных данных.

Ключевые слова: архитектура программного обеспечения; архитектурные паттерны; структурные метрики; предметно-ориентированное проектирование; ограниченный контекст; гибридная архитектура; автоматизированное проектирование; пространство архитектурных решений; микросервисная архитектура; многокритериальный анализ

Введение

Проектирование архитектуры программных систем обычно выполняется экспертами исходя из опыта и интуиции. Архитектор оценивает требования, выбирает архитектурный стиль и набор паттернов, однако этот процесс субъективен, плохо масштабируется и не допускает систематического сравнения альтернативных решений.

Методология предметно-ориентированного проектирования Domain-Driven Design, DDD [1] и практика EventStorming [2] позволяют структурировать предметную область. Это делается в виде ограниченных контекстов Bounded Contexts. В ранее опубликованной работе [3] авторами рассмотрены методы и алгоритмы проектирования оптимальной архитектуры ПО, в рамках которых обоснована целесообразность использования EventStorming для структурирования предметной области и автоматизированного построения графа предметной области с последующим выделением ограниченных контекстов. Однако переход от выделенных контекстов к конкретному архитектурному решению остаётся неформализованным.

Существующие подходы к оценке архитектуры обычно ограничиваются качественным анализом, например ATAM и CBAM [4; 5]. Перечисленные методы либо не предусматривают количественной формализации, либо рассматривают фиксированный набор стилей без учёта возможности их комбинирования в рамках одной системы [6]. Сравнение предлагаемого подхода с классическими методами приведено в таблице 1.

Таблица 1

Сопоставление подхода с ATAM и CBAM

Критерий	ATAM	CBAM	Предлагаемый подход
Тип анализа	качественный	качественно-экономический	количественный структурный
Автоматизация	ручной процесс	ручной процесс	автоматическая генерация
Формализация	сценарии и утилитарное дерево	экономические функции	формальный кортеж и метрики
Поддержка гибридных архитектур	не предусмотрена	не предусмотрена	встроена в модель

Разработано автором

На практике информационные системы всё чаще реализуются как гибридные архитектуры, в которых различные подсистемы функционируют в разных стилях.

Целью этой работы является разработка формальной модели пространства архитектурных решений и системы структурных метрик, обеспечивающих автоматизированную генерацию, проверку корректности и сравнительный анализ архитектурных вариантов, включая гибридные. Эта работа сознательно ограничивается тремя базовыми макростилями и структурными свойствами решения; полноценная оценка качества в смысле ISO/IEC 25010 (производительность, безопасность, эксплуатируемость) выходит за рамки статьи.

1. Формальная модель архитектурного решения

1.1 Уровни модели

Модель оперирует тремя уровнями абстракции, которые необходимо различать, поскольку метрики качества определены на разных уровнях:

- **Предметный доменный уровень** является графом предметной области G = (V, E) и множеством ограниченных контекстов C. Объекты этого уровня — сущности и связи предметной модели, описанной средствами EventStorming.
- **Логический архитектурный уровень** включает отображение контекстов в архитектурные стили α: C → S, разбиение на подсистемы {C_s} и множество межконтекстных потоков данных F. Этот уровень описывает намерения о реализации, но не технологический выбор.
- **Технологический уровень** — назначения протоколов α_π, стратегий данных α_d и моделей развёртывания α_δ. Этот уровень уточняет, какими техническими средствами реализуется логическая архитектура.

Метрики m_2, m_3, m_4, m_6 определены на предметном и логическом уровнях и не зависят от технологического выбора. Метрики m_1, m_7 используют проекцию технологического уровня веса протоколов на структурные объекты предметного для m_1 и логического для m_7 уровней. Метрики m_5, m_8, m_9 определены на технологическом уровне.

Связь между предметным и логическим уровнями задаётся правилом формирования множества потоков данных F из межконтекстных рёбер E_cross ⊆ E. В общем случае применяется одна из двух процедур:

- *Явная разметка. Если в описании EventStorming или иных исходных данных межконтекстные взаимодействия размечены явно на уровне команд, запросов и событий стрелки между контекстами, то F задаётся прямо этой разметкой. Такое F обычно сильно меньше E_cross, поскольку многие предметные связи не выходят за границы контекста на логическом уровне.*
- *Проекция по умолчанию. В отсутствие явной разметки принимается $F = \{(c\_i \to c\_k) \mid \exists (u, v) \in E\_cross, ctx(u) = c\_i, ctx(v) = c\_k\}$, то есть F получается как проекция E_cross на пары контекстов с удалением кратности. В этом случае $|F| \le |E\_cross|$, причём соответствие чаще всего неинъективно: один поток агрегирует несколько предметных рёбер.*

Во всех экспериментах раздела 7 использовалось правило проекции по умолчанию. Разработка более развитой модели F, учитывающей тип и частоту взаимодействия, остаётся задачей будущих работ.

1.2 Основные определения

Пусть G = (V, E) — ориентированный мультиграф предметной области, где V — множество сущностей предметной области: агрегатов, событий, команд и акторов, а E — множество типизированных связей между ними.

Пусть C = {c_1, c_2, …, c_n} — множеством ограниченных контекстов, полученных графовой аппроксимацией через кластеризацию графа G методом Лувена [7]. Под ограниченным контекстом в настоящей работе, вслед за Эвансом [1], понимается граница согласованности модели предметной области. Внутри этой границы термины Ubiquitous Language имеют однозначный смысл, а модель остаётся внутренне непротиворечивой. Графовая кластеризация даёт кандидатов на ограниченные контексты, основываясь на плотности связей; окончательная валидация границ предполагает учёт семантики языка, организационных границ и ownership, что выходит за рамки чисто графового анализа. В настоящей работе полученные методом Лувена кластеры принимаются как исходные данные для дальнейшего анализа.

Издательство «Мир науки» \ Publishing company «World of science» http://izd-mn.com

Каждому контексту c_i соответствует подмножество вершин $V\_i \subseteq V$, причём $\{V_1, \ldots, V\_n\}$ образует разбиение множества V (формула 1). **Размером контекста** называется число его вершин:

$$|c\_i| = |V\_i|. \quad (1)$$

Эта величина задействуется далее в эвристиках генерации (раздел 6) и в метрике гранулярности m_4 (раздел 5.1). Ограничения данной меры обсуждаются в разделе 8. Это ограничение заключается в игнорировании внутренней алгоритмической сложности контекста.

Определим множество архитектурных стилей (формула 2):

$$S = \{MONOLITH, SOA, MICROSERVICES\}. \quad (2)$$

Термины «модульный монолит» и «событийно-управляемая архитектура» не анализируются как глобальные стили информационной системы, причём первый является внутренней организацией монолитного компонента, второй является паттерном взаимодействия, причём в отношении бессерверных вычислений в настоящей работе принята позиция, по которой serverless анализируется как модель развёртывания, а не самостоятельный архитектурный стиль, что согласуется с классификацией CNCF [8]. Выбор именно трёх макростилей представляет сознательное ограничение скоупа работы; расширение таксономии остаётся задачей следующих этапов.

Зафиксируем также множество структурных прокси-метрик $M = \{m_1, m_2, \ldots, m_9\}$, элементы которого определяются в разделе 5 и используются при описании эффектов паттернов (раздел 3).

1.3 Кортеж архитектурного решения

Архитектурное решение определяется как кортеж (формула 3):

$$A = (G, C, \alpha, \alpha\_d, \alpha\_\pi, \alpha\_\delta, P, I), \quad (3)$$

где:

G — граф предметной области;

C — множество ограниченных контекстов;

$\alpha: C \to S$ — отображением, задающим проектное решение о реализации каждого контекста как элемента соответствующей архитектурной подсистемы. Контекст, для которого $\alpha(c\_i) = MONOLITH$, работает как модуль внутри монолитного компонента; при $\alpha(c\_i) = SOA$ — как SOA-сервис; при $\alpha(c\_i) = MICROSERVICES$ — как отдельный микросервис;

$\alpha\_d: C \to D$, $D = \{shared\_db, schema\_per\_service, db\_per\_service\}$ — стратегия управления данными;

$\alpha\_\pi: C \to \Pi$, $\Pi = \{in\_process, REST, gRPC, broker\}$ — протокол исходящей коммуникации;

$\alpha\_\delta: C \to \Delta\_dep$, $\Delta\_dep = \{single, containers, orchestrated\}$ — модель развёртывания;

$P = P\_g \cup \cup P\_i$ — множество применённых паттернов (P_g — глобальные, P_i — локальные для контекста c_i);

I — множество нарушенных ограничений корректности (пустое для допустимого решения).

Для краткости используются обозначения $d\_i = \alpha\_d(c\_i)$, $\pi\_i = \alpha\_\pi(c\_i)$, $\delta\_i = \alpha\_\delta(c\_i)$.

Отображение α индуцирует разбиение множества контекстов (формула 4):

Издательство «Мир науки» \ Publishing company «World of science» http://izd-mn.com

$$C = \sqcup_{s \in S} C_s,\ C_s = \alpha^{-1}(s). \quad (4)$$

Каждое непустое множество C_s соответствует подсистеме стиля s, объединяющей все контексты, реализованные в этом стиле: C_MONOLITH образует модули единого монолитного компонента, C_SOA является множеством SOA-сервисов, C_MICROSERVICES является множеством микросервисов. Гибридная архитектура соответствует случаю, когда более одного из множеств C_s непусто.

Особенность модели: отображения задаются независимо на каждом контексте, что естественным образом поддерживает гибридные архитектуры. Полное пространство решений только по стилям составляет $|S|^n = 3^n$ вариантов. В дальнейшем для краткости используются обороты вида «монолитный контекст» или «микросервисный контекст» как сокращение для «контекст, реализуемый как элемент подсистемы соответствующего стиля».

1.4 Значения по умолчанию и их переопределение

Значения дополнительных назначений d_i, π_i, δ_i определяются стилем контекста и являются именно значениями по умолчанию, а не жёсткими ограничениями модели. В отсутствие специальных условий сценария находит применение следующая схема: для MICROSERVICES — d_i = db_per_service, π_i = gRPC, δ_i = orchestrated; для MONOLITH — d_i = shared_db, π_i = in_process, δ_i = single; для SOA — d_i = schema_per_service, π_i = REST, δ_i = containers.

Эти значения могут быть явно переопределены при наличии дополнительных ограничений сценария или проектных предпочтений. В частности, модель допускает конфигурацию модульного монолита с раздельными схемами, в которой α(c_i) = MONOLITH, но d_i = schema_per_service. Контексты физически выполняются в одном процессе, но их данные разделены на уровне схем единой базы. В этом случае метрика m_5 Data Autonomy для таких контекстов принимает значение 0,5 вместо 0, что адекватно отражает промежуточную степень автономности данных. Аналогично возможны и другие сочетания, выходящие за пределы значений по умолчанию.

2. Каталог архитектурных паттернов

В модель включены 24 архитектурных паттерна из трёх источников: 14 паттернов Ричардсона, 8 паттернов Эрла и 2 инфраструктурных паттерна. Русскоязычное рассмотрение для Java-платформы приведено в работе Корниенко Д. В. [9]. Отбор выполнен по критерию наличия формализованных постусловий, операциональных эффектов на структурные характеристики и явных конфликтных зависимостей с другими паттернами. Паттерны уровня интеграции между системами Enterprise Integration Patterns [10] ориентированы на межсистемное взаимодействие и выходят за рамки внутриархитектурного проектирования. Их включение изучается как возможное расширение модели.

Каждый паттерн p_j описывается кортежем (формула 5):

$$p_j = (id, S_j^{aff}, scope_j, \Delta_j, conf_j, req_j), \quad (5)$$

где:

$S_j^{aff} \subseteq S$ — множество стилей, к которым паттерн применим;

$scope_j \in \{global, context\}$ — область действия;

$\Delta_j: M \to [-1, 1]$ — функция эффектов на метрики из M; в каталоге выполняется ограничение $|\Delta_j(m_k)| \le 0.5$ для всех j, k, что гарантирует невырождение мультипликативной композиции (см. раздел 5.2).

- conf_j ⊆ P — множество конфликтующих паттернов;
- req_j ⊆ P — множеством обязательных зависимостей.

Значения Δ_j(m_k) представляют собой экспертную шкалу эффектов, сформированную на основе описаний паттернов в первоисточниках [11; 12] и рекомендаций по их применению. Данные значения следует рассматривать как начальную калибровку, подлежащую уточнению по экспертным оценкам и эмпирическим данным; разработка формальной процедуры элицитации и оценка межэкспертной согласованности являются направлениями будущих работ. Состав каталога паттернов представлен в таблице 2.

Таблица 2

Группы паттернов каталога

Группа	Кол-во	Примеры
Richardson	14	API Gateway, Circuit Breaker, CQRS, Event Sourcing, Saga, Database per Service
Erl SOA	8	Service Abstraction, Service Autonomy, Loose Coupling, Statelessness
Infrastructure	2	Bulkhead, Retry with Backoff

Разработано автором

3. Ограничения корректности

Допустимость решения A проверяется на четырёх уровнях, различающихся областью действия. Уровни 1 и 2 проверяют свойства отдельных паттернов и их совместимость со стилями; уровень 3 — парные отношения между контекстами; уровень 4 — свойства системы как функции разбиения {C_s}. Следует различать жёсткие логические ограничения (уровни 2, 3 — физически невозможные комбинации) и рекомендуемые проектные инварианты (уровни 1, 4 — отражающие распространённые проектные предпочтения). Формальное разделение этих категорий относится к уточнениям модели.

Уровень 1. Конфликты и зависимости паттернов. Для каждого паттерна p_j ∈ P ни один паттерн из conf_j не должен присутствовать в P, и все паттерны из req_j должны присутствовать в P.

Уровень 2. Совместимость паттернов со стилями. Локальный паттерн p_j ∈ P_i допустим только если α(c_i) ∈ S_j^aff.

Уровень 3. Граничные правила коммуникации. Для каждой пары контекстов (c_i, c_k), связанных потоком данных, протокол π_i должен входить в множество допустимых протоколов Π(α(c_i), α(c_k)). Множества Π(·, ·) задаются таблицей 3.

Таблица 3

Допустимые протоколы коммуникации Π(α(c_i), α(c_k))

Источник / Приёмник	MONOLITH	SOA	MICROSERVICES
MONOLITH	in_process, broker	REST, gRPC, broker	REST, gRPC, broker
SOA	REST, gRPC, broker	REST, gRPC, broker	REST, gRPC, broker
MICROSERVICES	REST, gRPC, broker	REST, gRPC, broker	REST, gRPC, broker

Разработано автором

Принципы построения таблицы: внутри одной монолитной подсистемы допустимы внутрипроцессные вызовы (in_process), и при этом любая межподсистемная коммуникация требует сетевого протокола REST или gRPC. Брокерное взаимодействие допустимо во всех случаях. В частности, брокер внутри одной монолитной подсистемы соответствует распространённой практике использования очередей сообщений для асинхронной внутренней обработки фоновых задач, воркеров и не превращает монолит в распределённую систему в

архитектурном смысле. Единственное принципиальное ограничение состоит в недопустимости in_process при участии сервисных подсистем SOA или MICROSERVICES, поскольку внутрипроцессный вызов между процессами физически невозможен. Это связано с тем, что такой механизм взаимодействия не может быть реализован в данных условиях.

Уровень 4. Глобальные инварианты подсистем. Инварианты этого уровня формулируются для подсистем C_s и описывают требования, активируемые самим фактом существования подсистемы. Если микросервисная подсистема C_MICROSERVICES непуста, обязателен паттерн API Gateway (R01); если |C_MICROSERVICES| > 1, обязателен Service Registry (R02); паттерны Database per Service (R08) и Shared Database (R09) взаимоисключающи на глобальном уровне. Эти инварианты отражают распространённые проектные практики; в реальных системах возможны обоснованные отклонения, например использование service discovery через примитивы платформы оркестрации вместо явного Service Registry. Учёт таких отклонений требует расширения модели.

Решение A считается **допустимым**, если $I = \emptyset$.

4. Система структурных прокси-метрик

Для каждого допустимого решения A вычисляется вектор из 9 метрик $m = m_1, \ldots, m_9$, где $m_k \in [0, 1]$. Вычисление происходит в два этапа: сначала определяются базовые значения из графа и назначения, затем к ним мультипликативно применяются эффекты паттернов. Метрики это структурные прокси отдельных архитектурных свойств. Они вычисляются из структуры графа и назначений и не учитывают эксплуатационных характеристик (latency, throughput, cost, operability, team cognitive load). Агрегация вектора m в единую оценку в настоящей работе не выполняется. Метрики анализируются покомпонентно, а свёртка и многокритериальная оптимизация выносятся на последующие работы.

4.1 Базовые метрики

m_1 — Связанность Coupling. Доля межконтекстных рёбер графа предметной области уровень сущностей, взвешенная по стоимости протокола коммуникации (формула 6):

$$m_1 = \Sigma_{(u,v) \in E\_cross} w(\pi\_ctx(u)) / (|E| \cdot w\_max), \quad (6)$$

где:

E_cross является рёбра графа G между узлами разных контекстов, w(π) является вес протокола, w_max = 1,3. Шкала весов w(in_process) = 0, w(gRPC) = 0,5, w(REST) = 1,0, w(broker) = 1,3 отражает относительный порядок латентности и сериализационных накладных. Внутрипроцессные вызовы не вносят сетевой стоимости, бинарный gRPC заметно дешевле текстового REST, а брокерное асинхронное взаимодействие добавляет сверх REST затраты на сериализацию и персистентность сообщений. Приведённые значения изучаются как начальные и подлежат калибровке по эксплуатационным данным в дальнейшем. Направление оптимизации: минимизация.

m_2 является связностью (cohesion). Средняя доля внутриконтекстных рёбер для всех инцидентных рёбер контекста (формула 7):

$$m_2 = (1/n) \Sigma r(c\_i),\ r(c\_i) = |E\_intra(c\_i)|/|E\_incident(c\_i)| \text{ если } |E\_incident(c\_i)| > 0; \text{ иначе } r(c\_i) = 1. \quad (7)$$

Ребро считается инцидентным контексту c_i, если хотя бы один из его концов принадлежит c_i.

Издательство «Мир науки» \ Publishing company «World of science» http://izd-mn.com

Случай |E_incident(c_i)| = 0 соответствует изолированному контексту без связей с другими сущностями графа; такой контекст формально обладает максимальной связностью, поэтому его вклад полагается равным единице. Метрика m_2 вычисляется на графе предметной области G и не зависит от выбранного стиля реализации. Деление на контексты сохраняется на лингвистическом уровне даже при монолитной реализации. Различия между однородными вариантами в итоговых значениях m_2 (см. раздел 7.3) обусловлены применением паттернов (раздел 5.2). Направление: максимизация.

m_3 является модулярностью Modularity. Произведение нормализованной модулярности Ньюмена Q [13] и коэффициента развёрточной гранулярности (формула 8):

$$m_3 = Q\_norm \cdot \bar{f}\_deploy,\ Q\_norm = (Q + 0{,}5)/1{,}5,\ \bar{f}\_deploy = (1/n)\ \Sigma f(\alpha(c\_i)). \quad (8)$$

Нормализация Q_norm выполняет линейное отображение теоретического диапазона Q ∈ [-0,5; 1,0] в [0; 1]; коэффициенты f(MONOLITH) = 0,4, f(SOA) = 0,7, f(MICROSERVICES) = 1,0 отражают степень физической изоляции развёртывания. Метрика имеет мультипликативную интерпретацию: Q_norm измеряет топологическую модульность, а $\bar{f}$_deploy — развёрточную модульность. Топологическая модульность показывает, насколько удачно контексты выделены как сообщества графа, а развёрточная модульность показывает, насколько выбранные стили усиливают физическую изоляцию. Произведение отражает принцип, по которому высокая итоговая модульность требует одновременно хорошей топологии и хорошего развёртывания. Плохая топологическая модульность не полностью компенсируется развёрточной изоляцией, и наоборот. При фиксированном разбиении контекстов Q_norm постоянно, а вариация m_3 по стилям определяется $\bar{f}$_deploy. При расчёте модулярности Q ориентация рёбер графа G не учитывается — граф изучается как неориентированный, что является стандартным упрощением при применении формулы Ньюмена [13] к ориентированным графам и не влияет на структурные выводы о наличии сообществ. Обобщения модулярности для ориентированных графов [14] могут быть подставлены без изменения остальной модели. Направление: максимизация.

m_4 — Гранулярность (Granularity). Коэффициент Джини распределения размеров контекстов s_i = |c_i|, вычисляемый в форме (формула 9) на основе ранжированных значений [15]:

$$m_4 = \Sigma\ (2i - n - 1) \cdot s\_(i)\ /\ (n \cdot \Sigma\ s\_(i)), \quad (9)$$

где:

s_(1) ≤ s_(2) ≤ … ≤ s_(n) — упорядоченные размеры контекстов. Эта форма математически эквивалентна классическому определению Джини через попарные абсолютные разности, но вычисляется за O(n log n) вместо O(n^2). Метрика равна 0 при равных размерах контекстов и стремится к 1 при сильном дисбалансе. Направление: **минимизация**.

m_5 является Автономность данных Data Autonomy. Среднее значение автономности стратегии управления данными (формула 10):

$$m_5 = (1/n)\ \Sigma\ a(d\_i),\ a(shared\_db) = 0,\ a(schema\_per\_service) = 0{,}5,\ a(db\_per\_service) = 1. \quad (10)$$

Направление: **максимизация**.

m_6 является сложностью API. Нормализованное среднее число операций на контекст (формула 11):

$$m_6 = min((1/n)\ \Sigma\ (|cmd\_i| + |qry\_i|)/20,\ 1). \quad (11)$$

Издательство «Мир науки» \ Publishing company «World of science» http://izd-mn.com

Константа 20 в знаменателе соответствует эмпирической границе, за которой интерфейс модуля изучается как нарушение принципа единственной ответственности (SOLID) и индикатор необходимости декомпозиции. Направление: минимизация.

m_7 — Коммуникационные накладные. Суммарный вес явно выделенных межконтекстных потоков данных логического архитектурного уровня, нормализованный по размеру графа (формула 12):

$$m_7 = \Sigma_{f \in F} w(\pi\_src(f)) / (|E| \cdot w\_max). \quad (12)$$

где:

F является множеством межконтекстных потоков данных, формируемым по правилам раздела 2.1. $F \neq E\_cross$: в типичных сценариях $|F| \ll |E\_cross|$, поскольку E_cross описывает связи между сущностями предметной области. При этом f — явно определённые потоки взаимодействия сервисов. Метрики m_1 и m_7 поэтому не являются коллинеарными; совпадение нулевых значений для чистого монолита отражает общее свойство отсутствия сетевых вызовов, а не тождественность метрик. Направление: минимизация.

m_8 является изоляцией отказов (Fault Isolation). Среднее значение устойчивости контекстов к каскадным отказам (формула 13):

$$m_8 = (1/n) \Sigma\ 1/(1 + |sync\_out(c\_i)|), \quad (13)$$

где:

|sync_out(c_i)| — число синхронных исходящих зависимостей. Направление: **максимизация**.

m_9 является Независимостью развёртывания (Deploy Independence). Комбинация автономности данных, модели развёртывания и штрафа за исходящие зависимости с верхним ограничением (формула 14):

$$m_9 = min(1, (1/n) \Sigma\ max(0, a(d\_i) + b(\delta\_i) - 0.2 \cdot |out(c\_i)|/|out\_max|)), \quad (14)$$

где:

b(single) = 0, b(containers) = 0.2, b(orchestrated) = 0.4, а нормирующая величина определяется как максимум числа исходящих зависимостей по всем контекстам данного решения (формула 15):

$$|out\_max| = max_{i = 1,\ldots, n} |out(c\_i)|. \quad (15)$$

При |out_max| = 0 (отсутствие исходящих зависимостей у всех контекстов) штрафной член полагается равным нулю. Коэффициенты a(·), b(·) и множитель штрафа 0,2 изучаются как эксперимент-ориентированные начальные значения, подлежащие калибровке по экспертным оценкам в дальнейшем. Направление: максимизация.

4.2 Применение эффектов паттернов

Эффекты паттернов применяются мультипликативно. Для каждой метрики m_k итоговое значение вычисляется как (формула 16):

$$m\_k^{итог} = clamp(m\_k^{база} \cdot П_{j: p\_j \in P} (1 + Δ\_j(m\_k)), 0, 1). \quad (16)$$

Мультипликативная композиция выбрана как инженерная аппроксимация, позволяющая учитывать относительное изменение метрики при применении паттернов. Она также предоставляет независимость от абсолютной шкалы метрики. Альтернативные схемы композиции аддитивная, минимаксная, нечётко-логическая в общем случае приводят к иным численным результатам; адекватность мультипликативной формы и учёт нелинейных взаимодействий между близкими

по эффекту паттернами (так, одновременное применение Loose Coupling, API Gateway и Circuit Breaker) представляют самостоятельную методологическую задачу и выносятся на последующие работы.

При выполнении условия $\Delta\_j(m\_k) > -1$ для всех j композиция нескольких паттернов с отрицательными эффектами не обнуляет метрику, а лишь по шагам ослабляет её. Каталог паттернов спроектирован с соблюдением более строгого ограничения $|\Delta\_j(m\_k)| \leq 0.5$ (раздел 3), что гарантирует невырождение композиции даже при одновременном применении нескольких паттернов с отрицательными эффектами.

5. Генерация архитектурных вариантов

Генератор перебирает комбинации стилей $(\alpha(c_1), \ldots, \alpha(c\_n)) \in S^n$. Для каждой комбинации автоматически назначаются паттерны по умолчанию, выбирается протокол коммуникации с учётом граничных правил и проверяются ограничения корректности.

Для $n \leq 6$ выполняется полный перебор всех 3^n комбинаций. При $n > 6$ находит применение эвристическое сужение. Малые контексты с $|c\_i| < 4$ (то есть содержащие менее четырёх вершин графа) закрепляются за стилем MONOLITH. Контексты с высокой интенсивностью записи закрепляются за стилем MICROSERVICES. Под высокой интенсивностью записи понимается заметное преобладание команд над запросами при достаточном абсолютном числе команд: $|cmd\_i| \geq 2|qry\_i|$ и $|cmd\_i| \geq 3$. Оставшиеся контексты перебираются полностью в лексикографическом порядке. Число возвращаемых допустимых вариантов ограничено сверху значением $K\_max = 500$.

Подчеркнём, что цель этой работы является демонстрация работоспособности модели и корректности процедур генерации и оценки, а не статистически репрезентативный обзор всего пространства решений. Эвристический режим и лексикографический порядок формируют систематически смещённую выборку. Это не влияет на проверку корректности модели, но ограничивает возможность экстраполяции конкретных численных значений на всём пространстве. Вопросы случайной выборки и направленного поиска на базе многокритериальной оптимизации (NSGA-III) вынесены в последующие работы; обоснование применимости генетических алгоритмов к задаче выбора архитектурного стиля приведено в [3].

Для обеспечения прозрачности генератор возвращает статистику поиска: полное пространство 3^n, эффективное пространство после сужения, число по сути рассмотренных комбинаций и число допустимых вариантов.

6. Демонстрационный эксперимент

6.1 Данные и порядок эксперимента

Эксперимент проведён на демонстрационном корпусе из 5 предметных сценариев, охватывающих различные области информационных систем: banking — розничные банковские операции (счета, платежи, кредиты); ecommerce — интернет-торговля (каталог, заказы, оплата, доставка); hospital — медицинская информационная система (пациенты, приёмы, назначения); logistics — управление цепочками поставок (отправления, маршруты, склады); education — образовательная платформа (курсы, учащиеся, оценивание). Сценарии заданы текстовыми описаниями бизнес-процессов в формате EventStorming и различаются по плотности связей графа. Число выделенных контекстов варьируется от 10 до 31. Для каждого сценария выполнен полный конвейер: извлечение триплетов → построение графа → выделение контекстов → генерация вариантов → расчёт метрик.

Издательство «Мир науки» \ Publishing company «World of science» http://izd-mn.com

6.2 Пространство вариантов

Статистика поиска по демонстрационным сценариям приведена в таблице 4.

Таблица 4

Статистика поиска по сценариям

Сценарий	n	Полное пр-во 3^n	Эффективное пр-во	Рассмотрено	Валидных	Режим
banking	31	$6{,}2\times10^{14}$	$2{,}5\times10^{12}$	500	500	эвр.
ecommerce	31	$6{,}2\times10^{14}$	$2{,}5\times10^{12}$	500	500	эвр.
hospital	16	$4{,}3\times10^{7}$	59 049	500	500	эвр.
logistics	10	59 049	243	243	235	полный
education	31	$6{,}2\times10^{14}$	$2{,}5\times10^{12}$	500	500	эвр.

разработано автором

Для сценария logistics 10 контекстов, 5 закреплённых эффективное пространство из 243 вариантов перебрано полностью, из них 235 прошли валидацию. Для крупных сценариев (31 контекст) применяемая эвристика закрепляет лишь небольшую долю контекстов (для banking, ecommerce, education — порядка 5 контекстов). Из-за этого эффективное пространство сокращается всего на 2–3 порядка и остаётся почти необозримым. Этот факт усиливает обоснование необходимости направленного поиска методами многокритериальной оптимизации, рассматриваемыми в последующих работах.

6.3 Сравнение однородных и гибридного вариантов

В таблице 5 приведены значения метрик для трёх однородных архитектур и одного гибридного варианта на сценарии logistics. Символом «*» отмечены лучшие значения по каждой метрике среди сравниваемых вариантов; метрика m_4 зависит только от распределения размеров контекстов и одинакова для всех назначений стилей.

Таблица 5

Сравнение однородных и гибридного вариантов logistics.

Метрика	Направл.	MONO	SOA	MICRO	Гибрид
m_1 Coupling	min	0,000*	0,022	0,011	0,007
m_2 Cohesion	max	0,946	1,000*	1,000*	0,981
m_3 Modularity	max	0,305	0,534	0,763	0,791*
m_4 Granularity	min	0,482	0,482	0,482	0,482
m_5 Data Autonomy	max	0,000	0,375	1,000*	0,550
m_6 API Complexity	min	0,245	0,206*	0,208	0,221
m_7 Comm Overhead	min	0,000*	0,072	0,034	0,018
m_8 Fault Isolation	max	0,775	0,936	1,000*	0,944
m_9 Deploy Indep.	max	0,000	0,625	1,000*	0,612

Значение $m_2 = 0{,}946$ для MONO отражает базовое значение метрики на графе предметной области, в котором сохраняются межконтекстные рёбра (связи между сущностями разных лингвистических контекстов). Значения $m_2 = 1{,}000$ для SOA и MICRO достигаются за счёт применения паттернов Service Abstraction, Loose Coupling и других, повышающих базовое значение метрики мультипликативно (раздел 5.2). Разработано автором

Результаты показывают, что монолитная реализация лидирует по связанности ($m_1 = 0$, нет сетевых вызовов) и коммуникационным накладным ($m_7 = 0$), но проигрывает по модульности, автономности данных и независимости развёртывания. Микросервисная реализация лидирует по пяти из девяти метрик, однако вносит ненулевые коммуникационные расходы. SOA занимает промежуточную позицию.

Ни один стиль не доминирует по всем метрикам одновременно. Это показывает многокритериальный характер задачи выбора архитектуры и служит мотивацией для

применения методов Парето-оптимизации в дальнейшем, причём систематический Парето-анализ пространства решений в настоящей работе не проводится; он является предметом отдельного исследования при помощи NSGA-III.

6.4 Наличие гибридных вариантов в рассмотренных подвыборках

В таблице 6 приведены количественные результаты по гибридным вариантам для всех 5 сценариев. Общее число допустимых вариантов в рассмотренной подвыборке, число гибридных среди них и число гибридов, не доминируемых ни одним из трёх однородных вариантов MONO, SOA, MICRO по всем 9 метрикам одновременно.

Таблица 6

Наличие гибридных вариантов в рассмотренных подвыборках.

Сценарий	Всего в подвыборке	Гибридных	Не доминируемых
banking	500	497	156
ecommerce	500	497	142
hospital	500	497	98
logistics	235	232	71
education	500	497	113

Разработано автором

Приведённые значения не являются статистической оценкой доли недоминируемых гибридов в полном пространстве решений и не допускают экстраполяции на генеральную совокупность архитектур. Для крупных сценариев (n = 31) исследуется подвыборка из первых 500 допустимых вариантов в лексикографическом порядке после эвристического сужения, что составляет долю порядка 10^{-10} от результативного пространства и формирует систематически смещённую выборку. Цель таблицы 6 — продемонстрировать, что недоминируемые гибридные варианты обнаруживаются во всех 5 сценариях и не являются редкостью в рассмотренных условиях. Статистически корректная оценка доли таких гибридов в полном пространстве требует либо случайной выборки, либо направленного поиска с построением Парето-фронта методом NSGA-III, что является предметом последующих работ.

6.5 Анализ конкретного гибридного варианта

Приведённый в таблице 5 гибридный вариант для сценария logistics имеет назначение стилей MICRO-MONO-MONO-MICRO-MONO-MONO-MONO-MICRO-MONO-MONO, причём три крупнейших контекста образуют микросервисную подсистему C_MICROSERVICES, остальные семь — единую монолитную подсистему C_MONOLITH. Этот вариант лучше всех трёх однородных по метрике m_3 Modularity и сохраняет нормальные значения по другим метрикам. Коммуникационные накладные $m_7 = 0.018$ ниже, чем у однородной микросервисной реализации (0.034), а изоляция отказов $m_8 = 0.944$ близка к SOA. Гибрид не Парето-доминируется ни одним из однородных вариантов, поскольку ни один из них не лучше его сразу по всем девяти метрикам. Это показывает пользу поддержки гибридных архитектур в модели.

7. Ограничения модели

Описанная модель имеет ограничения, которые определяют перспективы развития.

Ограничения пространства поиска. Для сценариев с большим числом контекстов (n > 6) задействуется эвристическое сужение пространства поиска с лексикографическим перебором. Результаты для крупных сценариев представляют анализ смещённого подпространства и

демонстрируют работоспособность модели, но не отражают всё пространство статистически репрезентативно; причём в частности приведённые в таблице 6 количественные данные по гибридным вариантам относятся к рассмотренной подвыборке и не допускают статистической экстраполяции.

Гранулярность назначения протокола. Коммуникационный протокол назначается на уровне контекста (π_i), а не на уровне каждой отдельной межконтекстной границы. Это упрощение влияет на корректность метрик m_1 и m_7. Один и тот же контекст может взаимодействовать с разными контрагентами синхронно (gRPC), асинхронно (broker) и по REST, однако в текущей модели ему сопоставляется единственный протокол. Переход к назначению протокола на уровне ребра π_ik или потока π_f необходим для уточнения.

Узость онтологии стилей. Множество S = {MONOLITH, SOA, MICROSERVICES} охватывает три базовых макростиля и не включает event-driven, serverless, space-based и другие архитектурные стили. Расширение таксономии с формальным различением уровней (стиль / паттерн / модель развёртывания / механизм интеграции) представляет направление следующих этапов.

Графовая аппроксимация контекстов. Выделение контекстов методом Louvain является графовой аппроксимацией: оно учитывает плотность связей, но не семантику языка, организационные границы и ownership. Они являются неотъемлемыми аспектами Bounded Context в DDD. Валидация границ контекстов с участием предметных экспертов остаётся за пределами автоматизируемой части модели.

Размер контекста как прокси сложности. В модели размер контекста отождествляется с числом вершин графа |c_i| = |V_i|. Эта мера отражает структурную долю контекста в предметной модели, но не принимает во внимание внутреннюю алгоритмическую сложность, глубину вычислений или объём бизнес-логики, заключённой в одной сущности. В реальных системах контекст из небольшого числа сущностей может содержать сложную вычислительную логику. Скажем, такой контекст может включать расчёт скоринга или ценообразования. При этом контекст из многих сущностей может представлять собой простой CRUD-реестр; это ограничение затрагивает метрику m_4 и эвристики раздела 6. Уточнение модели размера через учёт дополнительных признаков (число инвариантов, объём правил, сложность сценариев использования) планируется в будущих работах.

Экспертная природа эффектов паттернов и коэффициентов. Значения Δ_j(m_k), веса протоколов w(π), коэффициенты a(·), b(·) и штрафы являются экспертной калибровкой, базирующейся на первоисточниках и практике, причём формальная процедура элицитации, оценка межэкспертной согласованности и анализ чувствительности ранжирования решений к вариации этих коэффициентов являются предметом отдельного исследования.

Ортогональность метрик. Предложенный набор из 9 метрик разделяется на предметно-уровневые, логико-архитектурные и технологические (раздел 2.1), однако формальный анализ их взаимной независимости корреляционный или факторный не проводился и выносится на последующие работы.

Структурный характер оценки. Рассмотрены только структурные прокси-метрики. Эксплуатационные характеристики: производительность, латентность, безопасность, экономические факторы, операционные затраты и организационные ограничения команды не включены в текущую модель и планируются к учёту в дальнейшем, включая отображение в высокоуровневую модель качества по ISO/IEC 25010.

Валидация модели. В настоящей работе проведён демонстрационный эксперимент на пяти синтетических сценариях. Полноценная валидация, включающая сопоставление автоматически

сгенерированных решений с экспертными архитектурными выборами, индустриальные кейсы и ablation study, представляет самостоятельный этап исследования.

8. Воспроизводимость

Реализация оформлена как Python-модуль formal_model/, включающий описание модели, каталог паттернов, ограничения, генератор вариантов, расчёт метрик и демонстрационный эксперимент. Для воспроизведения результатов без повторного NLP-извлечения применяется режим загрузки эталонных данных:

```
python -m formal_model.experiment -fast
```

```
python -m pytest tests/test_formal_model.py -v
```

Исходный код и данные доступны в репозитории проекта по адресу, указанному в сопроводительных материалах публикации.

Заключение

Предлагается формальная модель пространства архитектурных решений, обеспечивающая автоматизированную генерацию и анализ допустимых архитектурных вариантов, опираясь на ограниченные контексты предметной области. Разработанная система ограничений корректности и структурных прокси-метрик даёт возможность сравнивать альтернативные решения в едином формализованном пространстве, включая гибридные архитектуры, понимаемые как системы, в которых контексты группируются в подсистемы разных стилей. Модель выстроена на трёх уровнях абстракции — предметном, логическом архитектурном и технологическом — что даёт явное разграничение между доменной моделью и архитектурными артефактами.

Экспериментальные результаты на 5 сценариях подтверждают, что ни один архитектурный стиль не доминирует по всем структурным метрикам одновременно, недоминируемые гибридные варианты обнаруживаются во всех рассмотренных сценариях, что обосновывает необходимость многокритериального подхода к выбору архитектуры и пользу поддержки гибридности. Статистически корректная оценка пространства решений является предметом последующих работ.

Полученные результаты формируют основу для последующего перехода к высокоуровневой модели качества по ISO/IEC 25010, к калибровке коэффициентов по экспертным и эксплуатационным данным, а также к многокритериальной оптимизации архитектурных решений методом NSGA-III [3].

ЛИТЕРАТУРА

1. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software / E. Evans. — Boston: Addison-Wesley, 2003. — 560 p. — ISBN 978-0-321-12521-7. — URL: https://www.oreilly.com/library/view/domain-driven-design-tackling/0321125215/ (дата обращения: 20.04.2026).
2. Brandolini A. Introducing EventStorming / A. Brandolini. — Leanpub, 2013. — URL: https://leanpub.com/introducing_eventstorming (дата обращения: 20.04.2026).

3. Корниенко Д. В. Методы и алгоритмы проектирования оптимальной архитектуры программного обеспечения / Д. В. Корниенко, А. В. Никулин, Д. В. Рыженков, А. А. Стычук, А. А. Федотова // Информационные системы и технологии. — 2025. — № 3(149). — С. 89–98. — URL: https://oreluniver.ru/public/file/archive/isit_ISiT_3-25_kratkiy.pdf (дата обращения: 20.05.2026).
4. Kazman R. ATAM: Method for Architecture Evaluation / R. Kazman, M. Klein, P. Clements. — Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 2000. — Report CMU/SEI-2000-TR-004. — URL: https://www.sei.cmu.edu/library/atam-method-for-architecture-evaluation/ (дата обращения: 20.04.2026).
5. Kazman R. Quantifying the Costs and Benefits of Architectural Decisions / R. Kazman, J. Asundi, M. Klein // Proceedings of the 23rd International Conference on Software Engineering, ICSE 2001. — Toronto, 2001. — P. 297–306. — DOI 10.1109/ICSE.2001.919103. — URL: https://doi.org/10.1109/ICSE.2001.919103 (дата обращения: 20.04.2026).
6. Корниенко Д. В. Анализ методов оценки архитектуры программного обеспечения / Д. В. Корниенко, А. В. Никулин // Технологии и техника: пути инновационного развития: сб. науч. статей 3-й Междунар. науч.-техн. конф., Воронеж, 17 июня 2025 г. — Курск: Университетская книга, 2025. — С. 146–153. — URL: https://www.researchgate.net/publication/401329310 (дата обращения: 22.04.2026).
7. Blondel V.D. Fast Unfolding of Communities in Large Networks / V.D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre // Journal of Statistical Mechanics: Theory and Experiment. — 2008. — Vol. 2008, № 10. — P. P10008. — DOI 10.1088/1742-5468/2008/10/P10008. — URL: https://doi.org/10.1088/1742-5468/2008/10/P10008 (дата обращения: 21.04.2026).
8. CNCF Serverless Working Group. CNCF Serverless Whitepaper v1.0. — Cloud Native Computing Foundation, 2018. — URL: https://github.com/cncf/wg-serverless/blob/master/whitepapers/serverless-overview/cncf_serverless_whitepaper_v1.0.pdf (дата обращения: 22.04.2026).
9. Корниенко Д. В. Архитектурные паттерны проектирования микросервисов в Java / Д. В. Корниенко, А. В. Никулин // Инновационные технологии современной научной деятельности: стратегия, задачи, внедрение: сб. статей по итогам Междунар. науч.-практ. конф., Иркутск, 24 апреля 2024 г. — Стерлитамак: АМИ, 2024. — С. 150–160. — ISBN 978-5-907808-57-7. — URL: https://ami.im/sbornik/MNPK-596.pdf (дата обращения: 22.04.2026).
10. Hohpe G. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions / G. Hohpe, B. Woolf. — Boston: Addison-Wesley, 2003. — 736 p. — ISBN 978-0-321-20068-6. — URL: https://dl.acm.org/doi/book/10.5555/940308 (дата обращения: 20.05.2026).
11. Richardson C. Microservices Patterns: With Examples in Java / C. Richardson. — Shelter Island: Manning Publications, 2018. — 520 p. — ISBN 978-1-61729-454-9. — URL: https://www.manning.com/books/microservices-patterns (дата обращения: 20.05.2026).
12. Erl T. SOA Design Patterns / T. Erl. — Upper Saddle River: Prentice Hall, 2009. — 814 p. — ISBN 978-0-13-613516-6. — URL: https://ptgmedia.pearsoncmg.com/images/9780136135166/samplepages/0136135161.pdf (дата обращения: 21.04.2026).

13. Newman M.E.J. Modularity and Community Structure in Networks / M.E.J. Newman // Proceedings of the National Academy of Sciences. — 2006. — Vol. 103, № 23. — P. 8577–8582. — DOI 10.1073/pnas.0601602103. — URL: https://doi.org/10.1073/pnas.0601602103 (дата обращения: 22.04.2026).
14. Leicht E.A. Community Structure in Directed Networks / E.A. Leicht, M.E.J. Newman // Physical Review Letters. — 2008. — Vol. 100, № 11. — P. 118703. — DOI 10.1103/PhysRevLett.100.118703. — URL: https://doi.org/10.1103/PhysRevLett.100.118703 (дата обращения: 20.04.2026).
15. Cowell F.A. Measuring Inequality / F.A. Cowell. — 3rd ed. — Oxford: Oxford University Press, 2011. — 256 p. — ISBN 978-0-19-959404-7. — URL: https://global.oup.com/academic/product/measuring-inequality-9780199594047 (дата обращения: 20.04.2026).

Nikulin Alexander Valeryevich
Bunin Yelets State University, Yelets, Russia
E-mail: avnikulin.niiaa@gmail.com
ORCID: https://orcid.org/0009-0005-8426-3629

A formal model for selecting software architecture based on structural metrics

Abstract. The author presents a formal model of the architectural solution space for automated design of software architecture of information systems. The study addresses the gap between a domain description and the selection of an architectural solution, which in many projects remains expert based and insufficiently reproducible. The article proposes representing an architectural option as a tuple that includes a domain graph, a set of bounded contexts, a mapping of contexts to architectural styles, data management strategies, communication protocols, deployment models, a catalog of architectural patterns, and a set of correctness constraints. To compare alternatives, the paper introduces nine structural proxy metrics describing coupling, cohesion, modularity, granularity, data autonomy, interface complexity, communication overhead, fault isolation, and deployment independence. The model was tested on a demonstration corpus of five domain scenarios covering banking, electronic commerce, healthcare, logistics, and education information systems. The results show that the model can automatically generate feasible architectural options, detect violations of constraints, and compare solutions by individual structural properties. The demonstration experiment indicates that monolithic, service-oriented, and microservice implementations have different advantages and limitations, while hybrid solutions can occupy a non-dominated position in the space of alternatives.

Keywords: software architecture; architectural patterns; structural metrics; domain-driven design; bounded context; hybrid architecture; automated design; architectural solution space; microservice architecture; multicriteria analysis