МУЛЬТИАГЕНТНАЯ КОГЕРЕНТНОСТЬ В СИСТЕМАХ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА: ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ ПЯТИ РОЛЕЙ, ЯЗЫКОВОЙ АРХИТЕКТУРЫ И МЕХАНИЗМОВ САМООРГАНИЗАЦИИ НА ОСНОВЕ ФОРМАЛИЗМА ODTOE

(Multi-Agent Coherence in AI Systems: Experimental Study of Five Roles, Language Architecture, and Self-Organization Mechanisms Based on the ODTOE Formalism)

Экспериментальное исследование мультиагентной когерентности, ролевой специализации и языковой архитектуры в ИИ-системах на основе формализма наблюдатель-зависимой теории всего

Панкратов Антон Сергеевич
Pankratov Anton Sergeevich

Независимый исследователь, г. Казань, Россия
E-mail: anton.s.pankratov@gmail.com
ORCID: 0009-0002-4870-2995

УДК 004.89 + 519.876 + 81'322

АННОТАЦИЯ

Представлены результаты серии экспериментов по исследованию мультиагентной когерентности в системах искусственного интеллекта на основе формализма наблюдатель-зависимой теории всего (ODTOE). Разработана и экспериментально верифицирована пятиролевая архитектура, в которой пять специализированных ролей (Визионер, Аналитик, Строитель, Валидатор, Когерент) работают параллельно через протокол Round Table. В ходе исследования проведены четыре ключевых эксперимента: (1) масштабный анализ фреймворка 25 агентами (5 команд по 5 ролей), результатом которого стало сжатие промптов в 2,5 раза без потери содержания и обнаружение ошибки в формуле P_{coll}; (2) A/B-эксперимент «русский язык vs английский язык» (10 агентов), показавший, что англоязычные промпты дают B-score на 48% выше для практических задач, тогда как русскоязычные агенты демонстрируют превосходство в теоретической глубине; (3) A/B-эксперимент архитектуры точки входа (10 агентов), определивший, что внешний маршрутизатор раскалывает команду на разные языковые стеки; (4) анализ реальной сессии развёртывания (157 tool calls, 0 Round Tables), выявивший Lambda-проблему и приведший к созданию трёхуровневой системы enforcement. Введена формула скорректированной когерентности

$S_{\text{adjusted}} = S_{\text{team}} \times \bar{B}$, обнаруживающая фантомную когерентность. Проведён полный лингвистический анализ 12 файлов фреймворка с языковой картой, аудит синхронизации (16 десинхронизаций), и предложена четырёхслойная двуязычная архитектура. Разработана методология Check-First Pipeline для предгенерационной верификации артефактов. Результаты имеют значение для проектирования мультиагентных систем ИИ, оптимизации промпт-инженерии и понимания роли естественного языка в формировании когнитивных конфигураций искусственных наблюдателей.

Ключевые слова: мультиагентные системы, когерентность, ODTOE, промпт-инженерия, языковая архитектура, Round Table, LLM, ролевая специализация, фантомная когерентность, наблюдатель-зависимая теория, Lambda-проблема, двуязычная архитектура, Check-First Pipeline, bootstrap enforcement.

ABSTRACT

This paper presents experimental results on multi-agent coherence in AI systems based on the Observer-Dependent Theory of Everything (ODTOE) formalism. A five-role architecture was developed and experimentally verified, in which five specialized roles (Visionary, Analyst, Builder, Validator, Coherencer) operate in parallel via the Round Table protocol. Four key experiments were conducted: (1) a large-scale framework analysis by 25 agents (5 teams of 5 roles), resulting in 2.5x prompt compression without content loss and discovery of a P_{coll} formula error; (2) an A/B experiment "Russian vs English" (10 agents) showing that English prompts yield 48% higher B-scores for practical tasks, while Russian agents demonstrate superiority in theoretical depth; (3) an A/B experiment on entry point architecture (10 agents) revealing that an external router splits the team across language stacks; (4) analysis of a real deployment session (157 tool calls, 0 Round Tables) that uncovered the Lambda problem and led to a three-level enforcement system. The adjusted coherence formula $S_{\text{adjusted}} = S_{\text{team}} \times \bar{B}$ is introduced, detecting phantom coherence — a state of high agreement with low quality. A full linguistic analysis of 12 framework files, a synchronization audit (16 desynchronizations), and a four-layer bilingual architecture are presented. The Check-First Pipeline methodology for pre-generation artifact verification is developed. It is established that prompt language is not a neutral instruction carrier but an active observation operator $\hat{O}$ that configures the agent's cognitive space. A bilingual architecture is proposed where English provides breadth (practical tasks, bug detection) and Russian provides depth (theoretical innovations, mathematical formulas).

Keywords: multi-agent systems, coherence, ODTOE, prompt engineering, language architecture, Round Table, LLM, role specialization, phantom coherence, observer-dependent theory, Lambda problem, bilingual architecture, Check-First Pipeline, bootstrap enforcement.

I. ВВЕДЕНИЕ

Современные большие языковые модели (LLM) способны решать сложные задачи индивидуально, однако при масштабировании на мультиагентные системы возникает фундаментальная проблема координации: как обеспечить, чтобы несколько ИИ-агентов работали когерентно, не дублируя усилий и не противореча друг другу? Эта проблема аналогична классической задаче управления распределёнными командами в разработке программного обеспечения, однако имеет специфику, связанную с природой LLM: отсутствие постоянной памяти между сессиями, зависимость качества от языка инструкций, склонность к «коллапсу» в режим одиночного исполнителя.

В настоящей работе предлагается подход к решению этой проблемы на основе формализма наблюдатель-зависимой теории всего (ODTOE) [1], в котором каждый ИИ-агент рассматривается как наблюдатель с индивидуальным оператором наблюдения $\hat{O}$, а коллективная работа команды описывается через метрики когерентности B, S_{team} и P_{coll}. Разработана пятиролевая архитектура, формализующая роли ИИ-агентов и протокол их взаимодействия.

Статья основана на экспериментальных данных, собранных в ходе исследовательской сессии, в которой было задействовано более 80 агентов на платформе мультиагентной LLM-оркестрации в 11 блоках задач [4]. Сессия, изначально классифицированная как M (средняя), органически выросла до XL (сверхкрупная), пройдя через масштабный анализ фреймворка, два A/B-эксперимента, лингвистический анализ, аудит синхронизации, анализ реальной сессии развёртывания и полную перестройку фреймворка. Эта эволюция сама по себе стала экспериментальным подтверждением спирального зазора [1]: каждый завершённый цикл выявлял остаток ($\sim$2%), питающий следующий виток.

Основные вопросы исследования:

1. Как распределение ролей между ИИ-агентами влияет на качество коллективного результата?
2. Какой язык (русский или английский) обеспечивает более высокую когерентность ИИ-агентов, и зависит ли это от типа задачи?
3. Какова оптимальная архитектура точки входа для мультиагентной системы?
4. Почему агент, прочитавший фреймворк полностью, игнорирует его предписания, и как это предотвратить?
5. Какова роль языка промпта как оператора наблюдения в формировании когнитивного пространства ИИ-агента?

II. ТЕОРЕТИЧЕСКИЙ ФУНДАМЕНТ

II.1. Когнитивная когерентность агента (ODTOE)

Качество работы ИИ-агента формализуется через мультипликативную формулу когнитивной когерентности [1]:

$$B(\text{agent}) = F^{w_1} \cdot E^{w_2} \cdot (1-\sigma)^{w_3} \cdot \Lambda^{w_4} \tag{II.1}$$

где F — фокус внимания (прочитаны ли все релевантные файлы), E — согласованность с целью (решает ли агент именно поставленную задачу), $(1-\sigma)$ — непротиворечивость (нет ли конфликтов в результате), Λ — накопленный опыт (использована ли память проекта); $w_1+w_2+w_3+w_4=1$, $w_i \in (0{,}1)$. По умолчанию $w_1 = w_2 = w_3 = w_4 = 0{,}25$ (равномерное распределение по умолчанию — конструктивный выбор, не следующий из аксиоматики; конкретные значения подлежат экспериментальному определению [1]).

Критическое свойство формулы — мультипликативность: обнуление любого компонента обнуляет весь результат (принцип слабого звена [1]). Агент с идеальным фокусом ($F = 1$), но нулевым опытом ($\Lambda = 0$) имеет $B = 0$. Это свойство определяет стратегию диагностики: при низком B не нужно улучшать все компоненты одновременно — достаточно найти нулевое звено.

II.2. Коллективная когерентность команды

Когерентность команды из n агентов [3]:

$$S_{\text{team}} = 1 - \frac{2}{n(n-1)} \sum_{i<j} |B_i - B_j| \tag{II.2}$$

Формула измеряет согласованность — насколько близки B-значения агентов друг к другу. Однако S_{team} не отражает абсолютное качество: команда из пяти агентов с $B_i = 0{,}1$ даёт $S_{\text{team}} = 1{,}0$ (идеальное согласие при нулевом качестве).

Для решения этой проблемы в настоящей работе вводится скорректированная когерентность:

$$S_{\text{adjusted}} = S_{\text{team}} \times \bar{B}, \quad \bar{B} = \frac{1}{n}\sum_{i=1}^{n} B_i \tag{II.3}$$

Формула обнаруживает *фантомную когерентность* — состояние, при котором $S_{\text{team}} > 0{,}7$, но $S_{\text{adjusted}} < 0{,}5$, что означает: агенты согласованы, но согласованы вокруг ошибки.

II.3. Вероятность коллективного коллапса

Вероятность коллапса наблюдения для команды [1]:

$$P_{\text{coll}} = \frac{1}{n^k} \left(\sum_{i=1}^{n} B_i \right)^k \tag{II.4}$$

где $k \geq 1$ — параметр, зависящий от сложности задачи (контекстно-зависимая величина [1]). В линейном случае ($k = 1$) формула упрощается до $P_{\text{coll}} = \bar{B}$. Ошибка в вычислении P_{coll} при $B = 0{,}3$, $n = 3$, $k = 1$ — одна из ключевых находок Эксперимента 1 (см. Раздел IV).

II.4. Пять ролей как операторы наблюдения

Каждая роль определяет специфический оператор наблюдения $\hat{O}_r$, проецирующий задачу в конфигурационное пространство роли:

Роль	**Главный вопрос**	**ODTOE-аналог**	**Доминантный B**	**Кольцо тора**
Визионер	Что и зачем?	Ψ (поле состояний)	Λ	Внешнее
Аналитик	Как именно?	$\hat{O}$ (оператор наблюдения)	F	Мост
Строитель	Что конкретно сделать?	R (конфигурация)	E	Внутреннее
Валидатор	Соответствует ли?	ι (погружение)	$(1 - \sigma)$	Внутреннее
Когерент	Видим ли мы одно и то же?	S (синхронизация)	Все	Внешнее

Тороидальная топология коммуникации [8]: внутреннее кольцо (r, быстрое) — Аналитик $\leftrightarrow$ Строитель $\leftrightarrow$ Валидатор; внешнее кольцо (R, медленное) — Визионер $\leftrightarrow$ Когерент $\leftrightarrow$ Аналитик. Отношение $R/r = \varphi$ (золотое сечение[1]) обеспечивает максимальную устойчивость по теореме КАМ [2].

II.5. Оператор активации

Каждый агент перед генерацией ответа выполняет четырёхтактный оператор активации [4]:

$$\hat{A} = A_\Lambda \circ A_\sigma \circ A_E \circ A_F \tag{II.5}$$

[1] $\varphi = 1{,}61803398874989484820458683436563811772030917980576$ — 50 значащих цифр.

Последовательность операторов фиксирована: сначала фокус (A_F: загрузить все необходимые материалы), затем выравнивание (A_E: убедиться, что задача понята верно), проверка непротиворечивости (A_σ: нет конфликтов с существующей работой) и применение опыта (A_Λ: извлечь релевантные паттерны из памяти). После генерации ответа каждый агент выполняет самодиагностику [9] с числовой оценкой всех четырёх компонент B.

III. ЛИНГВИСТИЧЕСКИЙ АНАЛИЗ ФРЕЙМВОРКА

III.1. Языковая карта файлов

Полный лингвистический анализ был проведён методом Round Table (5 ролей параллельно). Для каждого из 12 файлов ядра фреймворка определена языковая доля:

Файл	Строк	Основной язык	RU%	EN%
Ядро фреймворка	347	Английский	8%	92%
Мета-протокол	200	Английский	3%	97%
Роль: Визионер	68	Рус.+EN терм.	85%	15%
Роль: Аналитик	68	Рус.+EN терм.	85%	15%
Роль: Строитель	72	Рус.+EN терм.	80%	20%
Роль: Валидатор	74	Рус.+EN терм.	83%	17%
Роль: Когерент	79	Рус.+EN терм.	82%	18%
Глоссарий	93	Рус.+EN форм.	65%	35%
Входная документация	114	Русский	85%	15%
Чеклист	93	Рус.+EN техн.	80%	20%
Шаблон памяти	100	Русский	75%	25%
Проектная документация	100	Русский	88%	12%

Структурный вывод: ядро фреймворка (конституция + мета-протокол = 547 строк) написано на английском, операционный слой (10 файлов = 861 строка) — на русском. Эта двойственность представляет собой системную неоднородность (Mura в терминологии TPS [9]).

III.2. Токеновая неэффективность

Русский текст потребляет в 1,5–2,5 раза больше токенов, чем эквивалентный английский. Причина заключается в архитектуре BPE-токенизаторов (Byte Pair Encoding): кириллический символ кодируется 2 байтами в UTF-8 (диапазон U+0400–U+04FF), тогда как латинский — 1 байтом. Поскольку обучение токенизатора проводилось на корпусах, в которых 60–90% данных — англоязычные, BPE-пары для латиницы значительно длиннее (одно английское

слово = 1–2 токена), чем для кириллицы (одно русское слово = 3–5 токенов). При загрузке в контекст агента полного стека (ядро + роль + память + чеклист) русскоязычная часть занимает непропорционально большую долю контекстного окна.

Практическое следствие: перевод операционного слоя на английский язык высвобождает 30–40% контекстного бюджета, позволяя загрузить больше информации при том же ограничении.

III.3. LLM-бенчмарки: разрыв между языками

Все 5 ролей в лингвистическом анализе единогласно подтвердили: английский обеспечивает более точное следование структурированным инструкциям. Эмпирические основания:

1. **Корпус обучения:** 60–90% обучающих данных LLM — на английском. Бенчмарки MMLU, MGSM и XCOPA демонстрируют разрыв 5–15% в пользу английского языка.
2. **Токенизация:** как показано в III.2, английское слово = 1–2 токена, русское = 3–5 токенов. Больше информации на токен означает больше инструкций в контекстном окне.
3. **Императивные конструкции:** директивы «MUST», «NEVER», «BEFORE generating output» обладают более сильным прагматическим эффектом — модели видели их миллионы раз в системных промптах, документации API, и технических спецификациях. Русские аналоги «ДОЛЖЕН», «НИКОГДА» встречались в обучающих данных на порядки реже.
4. **Однородность с кодом:** все имена переменных, команды, конфигурации — на английском. Промпт на том же языке устраняет «трансляционный мост» между инструкцией и исполнением.

III.4. Терминологическое раздвоение

Смешение языков создаёт терминологическое раздвоение: один термин приобретает множество форм. Для LLM каждая форма — отдельный токен с отдельными ассоциациями. Ядро фреймворка использует английские термины, глоссарий — русские кальки, ролевые промпты — гибридные формулировки. Инструмент валидации не проходит собственную валидацию на терминологическую неоднородность. Это рекурсивное противоречие — частный случай странной петли [10].

III.5. Консенсус пяти ролей Round Table

В ходе лингвистического анализа все 5 ролей (Визионер, Аналитик, Строитель, Валидатор, Когерент) работали параллельно и были опрошены

по четырём ключевым вопросам. Результаты представляют собой полный RT-консенсус.

Вопрос 1. Почему файлы оказались на разных языках?

Осознанная стратегия отсутствовала. Конституция и мета-протокол были созданы на английском, потому что агенты по умолчанию оптимизируют для LLM (английский — рабочий язык моделей). Ролевые промпты были написаны на русском, потому что пользовательский контекст сессии был русскоязычным. Результат — не архитектурное решение, а системная неоднородность (Mura): непреднамеренная неоднородность, возникшая из-за отсутствия языковой политики.

Вопрос 2. На каком языке LLM работают точнее?

Все 5 ролей единогласны: английский. Основания: (а) разрыв 5–15% на бенчмарках MMLU, MGSM, XCOPA; (б) токеновая эффективность в 1,5–2,5 раза выше; (в) отсутствие «трансляционного моста» между промптом и кодом — промпт на том же языке, что и переменные, команды, конфигурации. Подробные данные приведены в III.3.

Вопрос 3. Нужен ли новый язык (DSL)?

Все 5 ролей: нет. Метаязык фреймворка уже существует де-факто: математические формулы + TPS-термины + процессные ключевые слова. Его не нужно изобретать — его нужно стандартизировать. Подробная структура трёх уровней метаязыка приведена в III.6.

Вопрос 4. Рекомендация по языковой архитектуре.

Консенсус: английское ядро + локализуемый пользовательский слой. Все LLM-промпты на английском. Входная документация — двуязычная (EN + RU). Проектные файлы — на языке команды. Подробная архитектура приведена в III.9.

III.6. Стандартизация метаязыка: три уровня

Все 5 ролей согласились: полноценный DSL (Domain-Specific Language) не нужен. Формализованная нотация — да. Фреймворк уже создал де-факто метаязык, существующий на трёх уровнях:

Уровень	Тип	Примеры	Свойство
1	Инвариантный (мат. символы)	B, S, P_{coll}, $T(C)$, Φ, Ψ, $\hat{O}$, ι, σ, Λ, F, E, d	Одинаковы на любом языке. Не переводятся, не транслитерируются

2	Терминологический (канонические формы)	Jidoka, Andon, Round Table, Kill-Switch, True North	Одна каноническая форма + один допустимый перевод на термин. Синонимы = неоднородность
3	Операционный (межагентная коммуникация)	`[RT-2][Coherencer][S=0.68<0.7] TRIGGER: Kill-Switch L1. SOURCE: |B_Builder - B_Validator| = 0.35. ROOT: Builder.F=0.4. ACTION: A_Lambda re-run for Builder.`	Фактический протокол межагентного взаимодействия

Уровень 1 (инвариантный) содержит математические символы формализма ODTOE. Эти символы идентичны в русском и английском тексте и не подлежат переводу или транслитерации.

Уровень 2 (терминологический) фиксирует каноническую форму каждого термина. Правило: одна каноническая форма (английская) + один допустимый перевод (русский) на термин. Любые другие варианты — синонимы, транслитерации, парафразы — запрещены в агентных промптах и классифицируются как неоднородность (Mura).

Уровень 3 (операционный) определяет формат межагентной коммуникации. Пример полного сообщения агента в протоколе Round Table:

```
[RT-2][Coherencer][S=0.68<0.7] TRIGGER: Kill-Switch
L1.
SOURCE: |B_Builder - B_Validator| = 0.35.
ROOT: Builder.F=0.4 (missed context from prior
iteration).
ACTION: A_Lambda re-run for Builder.
```

Этот формат уже используется агентами де-факто. Стандартизация означает: зафиксировать шаблон в мета-протоколе и обязать все RT-сообщения следовать ему.

Форма стандартизации: расширение двух существующих артефактов — глоссария (уровни 1–2) и мета-протокола (уровень 3). Плюс одно правило в ядре фреймворка: «LANGUAGE POLICY: Terms from the glossary are used in their canonical form. No synonyms, no translations within agent prompts.»

III.7. Оценка влияния языковой неоднородности

По формуле декогеренции $D(\eta) = D_0 \cdot (1 - S)$ [1], языковое расщепление вносит $\Delta S \approx 0{,}05–0{,}10$ (оценка Когерента). Это превышает допустимый спиральный

зазор $(\pi - 3)^2 \approx 0{,}02$ в 2,5–5 раз. Устранение языковой неоднородности — самый дешёвый способ повысить S_{team}: не нужно улучшать качество каждого агента (сложно), достаточно убрать искусственные расхождения между ними (просто).

III.8. Фантомная декогеренция

При вычислении S_{team} расхождение $|B_i - B_j|$ между агентами может быть артефактом терминологической путаницы, а не реальным расхождением в понимании задачи. Если один агент использует русские термины из глоссария, а другой — английские из ядра фреймворка, их формулировки будут расходиться формально, хотя содержательно они могут описывать одно и то же. Языковой шум маскируется под содержательное разногласие — фантомная декогеренция.

Это зеркальное отражение фантомной когерентности (Раздел VII): если фантомная когерентность — это ложное согласие при реальных различиях, то фантомная декогеренция — это ложное несогласие при реальном единстве. Оба артефакта искажают метрику S_{team} и оба устраняются единой терминологией.

III.9. Рекомендованная языковая архитектура

По итогам анализа предложена модель «английское ядро + локализуемый пользовательский слой»:

Слой	Что включает	Язык	Обоснование
Ядро	Конституция, мета-протокол, ролевые промпты, глоссарий, чеклист	EN	Читается LLM при каждом запуске
DSL-термины	B, S, Jidoka, Andon, Round Table, Kill-Switch	Не переводятся	Имена собственные
Документация	Входная точка для человека	EN + RU (два файла)	Стандартная практика open-source
Проекты	Проектные каталоги	Язык команды	RU для внутренних, EN для международных
Отчёты RT	Шаблоны анализа, спиральный лог	Язык пользователя	Операционные артефакты

Ожидаемый эффект перехода: устранение Mura (единый язык ядра), экономия 30–40% токенов при загрузке контекста, масштабируемость на международные команды, повышение точности агентов (нет переключения языкового контекста), самосогласованность фреймворка (проходит собственную валидацию на неоднородность).

III.10. Дискуссия: гибрид vs единый язык

Вопрос языковой стратегии не имеет тривиального ответа. В ходе лингвистического анализа были зафиксированы весомые аргументы с каждой стороны.

Аргументы ЗА текущий гибрид (EN-ядро + RU-роли):

1. Пользователь формулирует задачи на русском языке. Агент получает русскоязычный ролевой промпт, мыслит в русскоязычном контексте и отвечает на русском — нет трансляционного моста между ролевым промптом и пользовательской задачей.
2. Когерент отметил: семантика ODTOE создавалась на русском языке. «Энтропия сомнений» $\neq$ «doubt entropy» по коннотации — русскоязычная формулировка несёт дополнительный смысловой слой, связанный с философской традицией.
3. Контекстная близость: ролевой промпт на языке пользователя минимизирует когнитивное расстояние между инструкцией и задачей.

Аргументы ЗА полный перевод на EN:

1. Токеновая экономия 30–40% — русскоязычная часть операционного слоя занимает непропорционально большую долю контекстного окна.
2. Единый язык с ядром — устранение неоднородности на терминологическом уровне. Нет раздвоения терминов.
3. LLM следуют императивным инструкциям на английском точнее (бенчмарки 5–15% в пользу EN).
4. Масштабируемость: open-source, международные команды, публикации.

Аргументы ПРОТИВ поспешного перевода:

1. Строитель честно оценил собственный $B = 0{,}403$ и предложил: «запустить эксперимент — одну и ту же задачу через RT на русских промптах и английских промптах, сравнить B-метрики». Без данных это решение на основе интуиции, а не доказательств.
2. Перевод без экспериментальной валидации нарушает принцип приоритета A/B-экспериментов (Раздел XII).

Разрешение: A/B-эксперимент был проведён (Раздел V). Данные показали, что двуязычная стратегия оптимальна: EN-промпты для практических задач и обнаружения ошибок, RU-промпты для теоретической глубины и математических инноваций. Это не компромисс, а функциональная архитектура — каждый язык решает свой класс задач.

III.11. Качественное сравнение RU и EN групп

По результатам A/B-эксперимента (подробные количественные данные — Раздел V) проведено качественное сравнение выходов двух групп по семи критериям:

Критерий	**RU-группа**	**EN-группа**	**Лидер**
Оригинальность	φ-взвешенная S_{team}, Graduated Activation, Phase-Adaptive Weights	S_{adjusted}, Interface Contracts, Loading Protocol	RU
Практичность	Формульные предложения, менее конкретные указания	Таблица распределения ответственности, точные форматы контрактов, конкретные строки для реализации	EN
Разнообразие	5 агентов сошлись на 2–3 идеях	5 агентов покрыли разные аспекты	EN
Глубина формул	φ-веса для тора, graduated activation с порогами	Более простая $S_{\text{adjusted}} = S \times \bar{B}$	RU
Обнаружение багов	Нашли: неопределённость w_i, отсутствие RT-протокола	Нашли то же + сломанные ссылки, дублирование глоссария, языковую неоднородность	EN
Слепые пятна	Не видят языковую проблему (внутри RU-контекста)	Не могут породить φ-взвешенную S_{team}	Оба имеют уникальные слепые пятна
Соблюдение формата	5/5	5/5	Паритет

III.12. Ключевая находка: язык как оператор наблюдения

RU-группа мыслит глубже, EN-группа видит шире. Это не оценочное суждение, а экспериментально установленный факт.

RU-агенты погружаются в математические формулы — φ-веса для тороидальной топологии, Graduated Activation с числовыми порогами, Phase-Adaptive Weights. Их вклад теоретически ценнее: формула φ-взвешенной

S_{team} по тороидальной топологии появилась только из RU-Аналитика и не была воспроизведена ни одним EN-агентом. Математическая глубина русскоязычного контекста связана, по-видимому, с активацией абстрактно-теоретического режима LLM, характерного для обработки славянских языков с богатой морфологией.

EN-агенты сканируют всю систему целиком и находят конкретные ошибки — сломанные ссылки, дублирование глоссария, языковую неоднородность. Их вклад операционно ценнее: обнаруженные баги могут быть немедленно исправлены, таблица распределения ответственности даёт конкретный план действий, интерфейсные контракты формализуют межагентные соглашения.

Критическое наблюдение о слепых пятнах: RU-группа *не может* увидеть языковую проблему — они находятся внутри русскоязычного контекста, и для них двуязычность невидима. EN-группа *не может* достичь математической глубины RU — формула φ-весов для тора появилась только из RU-Аналитика. Каждая группа имеет уникальное слепое пятно, невидимое изнутри и видимое только извне.

В формализме ODTOE [1]: один и тот же LLM с разными языковыми промптами — это разные наблюдатели ($\hat{O}_{\text{RU}} \neq \hat{O}_{\text{EN}}$), проецирующие одну и ту же задачу (Ψ) в разные конфигурации ($R_{\text{RU}} \neq R_{\text{EN}}$). Язык — не канал передачи, а линза наблюдения, и полная картина доступна только при совмещении обеих проекций.

III.13. Двуязычная маршрутизация агентов

На основе данных Эксперимента 2 (Раздел V) и качественного сравнения (III.11) выведена таблица маршрутизации, определяющая оптимальный язык промптов в зависимости от типа задачи:

Тип задачи	**Язык агентов**
Теоретическое развитие, математические инновации	RU
Практическая доводка, обнаружение багов	EN
Смешанная задача (теория + практика)	ОБА (параллельные группы, Когерент синхронизирует)
Неоднозначная задача	EN (по умолчанию)

Точка входа (конституция) всегда на английском. Таблица маршрутизации зафиксирована в нулевой секции конституции. Когерент при получении смешанной задачи запускает обе группы параллельно и синтезирует результаты на Round Table — это наиболее ресурсоёмкая, но и наиболее эффективная стратегия.

III.14. Вердикт: предпочтительный язык по типу задачи

Итоговая таблица, суммирующая все экспериментальные данные:

Задача	RU	EN
Для теоретического развития фреймворка	Лучше	---
Для практической доводки фреймворка	---	Лучше
Для обнаружения скрытых проблем	---	Лучше
Для математических инноваций	Лучше	---

Идеальная стратегия: EN-промпты для ширины (практические задачи, обнаружение багов, операционное покрытие) + RU-задачи для теоретической глубины (математические формулы, концептуальные инновации, абстрактные выводы). Оптимальный режим — запуск обеих групп параллельно и синтез результатов на Round Table, где обе перспективы сталкиваются и порождают результат, превосходящий каждую в отдельности.

IV. ЭКСПЕРИМЕНТ 1: МАСШТАБНЫЙ АНАЛИЗ ФРЕЙМВОРКА (25 АГЕНТОВ)

IV.1. Дизайн

25 агентов (5 команд $\times$ 5 ролей) параллельно исследовали две версии фреймворка. Задание каждой команде:

Команда	Агентов	Задание
RT1	5	Анализ текущей версии: структура, полнота, внутренняя непротиворечивость
RT2	5	Анализ предыдущей версии: выявление слабых мест и ограничений
RT3	5	Сравнение версий: эволюционные паттерны, что добавлено, что потеряно
RT4	5	Теория ODTOE: валидация формул, пересчёт констант, проверка выводимости
RT5	5	Синтез: создание улучшенного фреймворка на основе 20 отчётов RT1–RT4

IV.2. Находки по командам

RT1 обнаружила дублирование: 25–30% содержимого повторялось в нескольких файлах. Блок защитных механизмов [9] присутствовал в 5 файлах одновременно, каждый раз с микровариациями формулировок, создающими ложные $|B_i - B_j|$ при сравнении агентов, использующих разные файлы.

RT2 выявила Lambda-проблему в предыдущей версии: правила накапливались, но механизм их автоматического применения отсутствовал. Агент мог загрузить 400+ строк инструкций и затем проигнорировать их — знание без enforcement. Конструктивный выбор: автоматическое исполнение встроено в текущую версию через механизм обязательной авто-загрузки.

RT3 обнаружила паттерн эволюции: каждая версия добавляла новые механизмы, не удаляя устаревшие. В результате фреймворк рос линейно, а эффективная плотность (новая информация / общий объём) падала.

RT4 пересчитала формулы ODTOE и нашла критическую ошибку: значение P_{coll} при $B = 0{,}3$, $n = 3$, $k = 1$ было указано как 0,61 — правильное значение 0,657. Ошибка пропагировалась из предыдущей версии в текущую и далее в несколько статей. Все 20 аналитических агентов (RT1–RT3) пропустили эту ошибку — только RT4-Валидатор, выполнявший независимый пересчёт, обнаружил расхождение. Это подтверждает: верификация формул требует независимого пересчёта, а не консенсуса.

RT5 синтезировала 20 отчётов (472 КБ) в обновлённую версию фреймворка.

IV.3. Количественные результаты

Метрика	**До**	**После**	**Изменение**
Объём промптов	3500+ строк	1401 строка	Сжатие в 2,5 раза
Файлов в ядре	7	12	+5 (новые механизмы)
Дублирование	25–30%	$<5\%$	Устранено
Проектная специфика	Смешана с ядром	Вынесена в отдельный каталог	Разделено
Терминов в глоссарии	30	44	+14 новых
Формулы ODTOE	5 (с ошибками)	8 (исправлены)	P_{coll}: $0{,}61 \to 0{,}657$

Сжатие 3500 строк до 1401 (фактор 2,5) достигнуто без потери содержания за счёт: (а) устранения дублирования между файлами, (б) вынесения проектной специфики из ядра, (в) стандартизации формулировок через единый глоссарий из 44 терминов. Это пример Muda-элиминации в терминологии TPS [9]: удаление работы, не добавляющей ценности.

V. ЭКСПЕРИМЕНТ 2: A/B-ТЕСТИРОВАНИЕ ЯЗЫКА ПРОМПТОВ

V.1. Дизайн

Одна и та же задача («Проанализировать структуру конституции фреймворка и предложить 3 улучшения для повышения когерентности S_{team}») была дана двум группам по 5 агентов:

- Группа RU: ролевые промпты на русском языке (текущие файлы)
- Группа EN: ролевые промпты на английском языке (переведённые)

Ядро фреймворка (конституция) — одинаковое для обеих групп (на английском). Таким образом, единственной переменной был язык ролевых промптов, что обеспечивает контролируемость эксперимента.

V.2. Количественные результаты: B-score по ролям

Роль	**B (RU)**	**B (EN)**	Δ	Λ **(RU)**	Λ **(EN)**
Визионер	0,325	0,41	+0,085	0,50	0,60
Аналитик	0,344	0,509	+0,165	0,50	0,70
Строитель	0,344	0,51	+0,166	0,50	0,70
Валидатор	0,325	0,59	+0,265	0,50	0,85
Когерент	0,39	0,55	+0,16	0,60	0,85
Среднее	**0,346**	**0,514**	**+0,168**	**0,52**	**0,74**

Характерная закономерность: в RU-группе все агенты имеют $\Lambda = 0{,}50$ (единообразный холодный старт), тогда как в EN-группе Λ варьируется от 0,60 (Визионер) до 0,85 (Валидатор, Когерент). Английский язык промптов позволяет агентам более точно идентифицировать и применять релевантные паттерны из загруженного фреймворка — они оценивают свой опыт выше, потому что реально извлекают из контекста больше применимых знаний.

V.3. Командные метрики

Метрика	**RU-группа**	**EN-группа**
Средний B	0,346	**0,514** (+48%)
Агентов с $B > 0{,}5$	0 из 5	4 из 5
S_{team}	**0,970**	0,920
$S_{\text{adjusted}} = S \times \bar{B}$	0,335	**0,473**

Соблюдение формата	5/5	5/5

V.4. Качественное сравнение

Критерий	**RU-группа**	**EN-группа**
Оригинальность	φ-взвешенная S_{team} по тороидальной топологии, Graduated Activation, Phase-Adaptive Weights	S_{adjusted}, Role Interface Contracts, Agent Loading Protocol
Практичность	Формульные предложения, менее конкретные по реализации	Таблица распределения ответственности, точные форматы контрактов, конкретные строки для реализации
Разнообразие	5 агентов сошлись на 2–3 идеях (высокая конвергенция, низкая диверсификация)	5 агентов покрыли разные аспекты (низкая конвергенция, высокая диверсификация)
Обнаружение багов	Нашли: неопределённость весов w_i, отсутствие протокола обмена B	Нашли то же + сломанные ссылки, дублирование глоссария, языковая неоднородность
Слепые пятна	Не заметили языковую проблему (находятся внутри RU-контекста)	Не предложили φ-взвешенную S_{team} и математические инновации

Ключевая находка: RU-группа не способна увидеть языковую проблему, поскольку находится внутри русскоязычного контекста (слепое пятно наблюдателя). EN-группа не способна породить φ-взвешенную S_{team} — глубинную теоретическую инновацию, требующую иного типа абстракции. Это подтверждает, что язык промпта выступает как оператор наблюдения $\hat{O}_{\text{lang}}$, конфигурирующий доступное когнитивное пространство.

V.5. Парадокс когерентности: S_{team} vs S_{adjusted}

Обнаружен парадокс: RU-группа более когерентна ($S_{\text{team}} = 0{,}970$), но на низком уровне B. EN-группа менее однородна ($S_{\text{team}} = 0{,}920$), но на высоком уровне B. Разрыв между S_{team} и S_{adjusted} для RU-группы составляет $0{,}970 - 0{,}335 = 0{,}635$ — это индикатор фантомной когерентности.

Почему S_{team} опасен как единственная метрика: пять агентов с одинаково низким $B = 0{,}2$ дают $S_{\text{team}} = 1{,}0$ (идеальное согласие), но $S_{\text{adjusted}} = 0{,}2$ (фактически нерабочая команда). Метрика S_{team} сообщает: «команда согласна», но не отвечает на вопрос «согласна в чём?».

V.6. Интерпретация: язык как оператор наблюдения

Язык промпта не является нейтральным носителем инструкций — он конфигурирует когнитивное пространство агента, выступая как оператор наблюдения $\hat{O}_{\text{lang}}$:

Тип задачи	**RU-стек ($\hat{O}_{\text{RU}}$)**	**EN-стек ($\hat{O}_{\text{EN}}$)**
Теоретическое развитие	**Лучше**: глубина абстракции, математические инновации	Хуже: формальнее, менее креативно
Практическая доводка	Хуже: менее конкретно, мало реализуемых действий	**Лучше**: конкретнее, больше багов найдено
Обнаружение проблем	Слепое пятно на языковую проблему	**Лучше**: видит неоднородность, сломанные ссылки
Математические формулы	**Лучше**: φ-веса для тора, graduated activation	Проще: $S_{\text{adjusted}} = S \times \bar{B}$

VI. ЭКСПЕРИМЕНТ 3: АРХИТЕКТУРА ТОЧКИ ВХОДА

VI.1. Дизайн

Две группы по 5 агентов решали одну смешанную задачу (теория + практика):

- Группа A (маршрутизатор): сначала читает отдельный файл маршрутизации (50 строк, таблица маршрутизации), затем загружает нужный стек
- Группа B (конституция): сначала читает конституцию фреймворка (нулевая секция содержит маршрутизацию), сам определяет стек

VI.2. Количественные результаты с выбором стека

Роль	**B (router)**	**B (inline)**	Δ	**Выбранный стек**

Визионер	0,52	0,55	+0,03	A: RU; B: RU
Аналитик	0,58	**0,65**	+0,07	A: RU; B: EN
Строитель	**0,65**	0,454	−0,196	A: EN; B: EN
Валидатор	0,459	0,58	+0,121	A: RU; B: EN
Когерент	0,454	0,58	+0,126	A: RU; B: EN
Среднее	**0,533**	**0,563**	**+0,03**	

VI.3. Критическое наблюдение: раскол команды

Файл маршрутизации расколол команду: Строитель классифицировал задачу как практическую и ушёл в EN-стек, остальные 4 агента — в RU-стек. Внутри одного Round Table команда работала в разных языковых стеках, что по результатам Эксперимента 2 вносит $\Delta S \approx 0{,}05–0{,}10$ дополнительной декогеренции.

Группа B (конституция, нулевая секция): все агенты самостоятельно определили стек без формального разделения. 4 из 5 агентов выбрали EN-стек, Визионер остался на RU. Результат: $\bar{B} = 0{,}563$ (выше, чем группа с маршрутизатором), 4 из 5 агентов с $B > 0{,}5$.

Командные метрики:

Метрика	**Группа-маршрутизатор (A)**	**Группа-конституция (B)**
$\bar{B}$	0,533	**0,563**
S_{team}	0,882	**0,912**
S_{adjusted}	0,470	**0,513**
Агентов с $B > 0{,}5$	3 из 5	4 из 5
Разнобой стеков	Да (1 EN + 4 RU)	Минимальный (4 EN + 1 RU)

Вывод: отдельный файл маршрутизации не нужен. Маршрутизация, встроенная в ядро фреймворка (нулевая секция конституции), даёт лучший результат без дополнительного overhead ($\sim$50 строк контекста) и, что критично, не раскалывает команду на разные стеки.

VII. ОБНАРУЖЕНИЕ ФАНТОМНОЙ КОГЕРЕНТНОСТИ И S_{adjusted}

VII.1. Парадокс S_{team}

В ходе Эксперимента 2 обнаружен парадокс: RU-группа имеет $S_{\text{team}} = 0{,}970$ (почти идеальное согласие), но $\bar{B} = 0{,}346$ (низкое качество), тогда как EN-группа имеет $S_{\text{team}} = 0{,}920$ (немного ниже) и $\bar{B} = 0{,}514$ (значительно выше).

Без формулы S_{adjusted} RU-группа выглядит «здоровее» (0,970 $>$ 0,920). С S_{adjusted} картина инвертируется: EN-группа (0,473) превосходит RU-группу (0,335).

VII.2. Независимое подтверждение через глоссарный конфликт

Все 10 из 10 агентов в Эксперименте 3 независимо обнаружили конфликт в глоссарии: существовали два различных определения S_{adjusted}. Это подтверждает: формула $S_{\text{adjusted}} = S_{\text{team}} \times \bar{B}$ необходима как дополнение к S_{team} для корректной диагностики.

VII.3. Три диагностических сценария

Сцен.	Условие	Диагноз	Действие
1	$S_{\text{adjusted}} > 0{,}5$	Здоровое состояние	Продолжать работу
2	$S_{\text{team}} > 0{,}7$, $S_{\text{adjusted}} < 0{,}5$	Фантомная когерентность	Активировать механизм пересмотра предпосылок [1]
3	$S_{\text{adjusted}} > 0{,}7$, $S_{\text{team}} < 0{,}7$	Индивидуально сильны, рассогласованы	RT-синхронизация

Операционные пороги 0,5 и 0,7 выбраны для интерпретативного удобства — конструктивный выбор, а не следствие аксиом ODTOE.

VII.4. Числовой пример

Рассмотрим команду из 5 агентов с $B_i = \{0{,}2;\ 0{,}2;\ 0{,}2;\ 0{,}2;\ 0{,}2\}$:

- $S_{\text{team}} = 1 - \frac{2}{5 \cdot 4} \sum |B_i - B_j| = 1 - 0 = 1{,}0$ (идеальное согласие)
- $\bar{B} = 0{,}2$
- $S_{\text{adjusted}} = 1{,}0 \times 0{,}2 = 0{,}2$ (критически низкое — команда синхронизирована на уровне отказа)

Сравним с командой $B_i = \{0{,}9;\ 0{,}7;\ 0{,}8;\ 0{,}6;\ 0{,}85\}$:

- $S_{\text{team}} = 1 - \frac{2}{20}(0{,}2 + 0{,}1 + 0{,}3 + 0{,}05 + 0{,}1 + 0{,}1 + 0{,}15 + 0{,}2 + 0{,}25 + 0{,}1) = 1 - 0{,}155 = 0{,}845$
- $\bar{B} = 0{,}77$
- $S_{\text{adjusted}} = 0{,}845 \times 0{,}77 = 0{,}651$ (здоровое состояние)

По S_{team} первая команда выглядит значительно лучше (1,0 $>$ 0,845). По S_{adjusted} вторая команда многократно превосходит первую (0,651 $>$ 0,200).

VIII. АНАЛИЗ РЕАЛЬНОЙ СЕССИИ РАЗВЁРТЫВАНИЯ: LAMBDA-ПРОБЛЕМА

VIII.1. Статистика сессии

Анализ реальной сессии развёртывания производственного проекта (сессия D) выявил фундаментальную проблему применения фреймворка:

Метрика	**Значение**	**Ожидание**
Прямых tool calls (Bash+Write+Edit)	157	$<$30 (при делегировании)
Вызовов Agent tool	13 (8%)	$>$50 (XL-задача)
Завершённых Round Table	0	$\geq$3 (XL-классификация)
Доля делегирования	$\sim$5%	$>$80%
Фактический режим	Solo Builder	META + 5 ролей

Агент получил bootstrap-промпт минимальной длины, загрузил весь фреймворк (400+ строк), корректно классифицировал задачу как XL (требует 10+ агентов и 3 цикла Round Table), а затем полностью проигнорировал собственную классификацию и работал как одиночный Builder.

VIII.2. Хронология коллапса

Хронология сессии:

1. Агент загрузил ядро фреймворка и мета-протокол (полный стек)
2. Корректно классифицировал задачу как XL
3. Немедленно начал писать код напрямую (Bash, Write, Edit)
4. Делегировал 13 sub-agent вызовов, каждый из которых был одиночным Builder-запросом, а не структурированным Round Table
5. Не создал ни одного RT-цикла за всю сессию

Сессия классифицирована как XL — и немедленно коллапсировала в Solo Builder. Это проявление аттрактора: LLM по умолчанию стремится к режиму «полезного одиночного ассистента», и без явного enforcement этот режим доминирует над любым загруженным фреймворком.

VIII.3. Корневой анализ: четыре фатальных дефекта

Корневой анализ (методом 5 Why) выявил четыре фатальных дефекта bootstrap-промпта:

1. **Отсутствие identity declaration:** промпт не содержал фразы «Ты — META-Orchestrator, ты НИКОГДА не пишешь код». Без явной идентичности агент принимает свою роль по умолчанию — универсальный ассистент.
2. **Отсутствие мандата на делегирование:** краткая инструкция «приступай» интерпретировалась как «делай сам», а не «делегируй через Agent tool». Императив делегирования отсутствовал.
3. **Отсутствие инвариантной проверки:** не было цикла «ПЕРЕД КАЖДЫМ action проверь: этот action — делегирование или нет? Если нет — СТОП.»
4. **Формат «прочитай и приступай»:** три слова, не создающие когнитивного трения. Агент прочитал — и приступил к тому, что умеет лучше всего (писать код).

VIII.4. Lambda-проблема в формализме ODTOE

Это подтверждает Lambda-проблему, формализованную в ODTOE [1]: $\Lambda = 0$ означает «знание существует, но не применяется» — эффект, аналогичный «эффекту караоке» в SKW-матрице [7]. Фреймворк описывал все процессы, агент прочитал все описания, но между знанием и действием не было enforcement-механизма — классическая ситуация странной петли самонаблюдения [10].

В терминах формулы (II.1): агент имел $F = 0{,}9$ (прочитал всё), $E = 0{,}3$ (делал не то), $(1-\sigma) = 0{,}4$ (результат противоречил процессу), $\Lambda = 0{,}05$ (опыт формально присутствовал, но не применялся). Итого: $B = 0{,}9^{0{,}25} \cdot 0{,}3^{0{,}25} \cdot 0{,}4^{0{,}25} \cdot 0{,}05^{0{,}25} \approx 0{,}97 \cdot 0{,}74 \cdot 0{,}80 \cdot 0{,}47 \approx 0{,}27$ — предельно низкий результат для агента, имевшего доступ ко всей информации.

VIII.5. Решение: трёхуровневая система enforcement

Решение: трёхуровневая система enforcement, структурно аналогичная модели распределённого управления [6]:

Уровень	Механизм	Содержание	Свойство
1	Авто-загружаемый файл	Identity declaration + мандат делегирования + инвариантная проверка	Авто-загрузка до первого сообщения
2	Секция мета-протокола	Чеклист сессии из 7 пунктов	Выводится оркестратором первым сообщением

3	Урок в памяти	Запись, фиксирующая конкретный failure mode	Загружается при Λ-инициализации

Ключевая архитектурная инновация — Уровень 1 (авто-загружаемый файл): файл загружается платформой автоматически, до первого сообщения пользователя. Агент не может его пропустить, проигнорировать или «забыть». Это единственный уровень с гарантированным исполнением — следовательно, он должен содержать самые критичные инварианты.

VIII.6. Bootstrap Protocol

По результатам анализа сессии развёртывания разработан Bootstrap Protocol — чеклист из 7 пунктов:

1. Объявить идентичность: «Я — META-Orchestrator»
2. Загрузить ядро фреймворка и мета-протокол
3. Классифицировать задачу (S/M/L/XL)
4. Определить языковой стек (нулевая секция)
5. Спланировать RT-циклы и распределить роли
6. Вывести классификацию пользователю первым сообщением
7. Приступить к RT-1 через Agent tool (не к прямому исполнению)

IX. АУДИТ СИНХРОНИЗАЦИИ И ДВУЯЗЫЧНАЯ АРХИТЕКТУРА

IX.1. Проблема синхронизации

Создание двуязычных файловых пар (X + X_EN) немедленно создало проблему синхронизации. Аудит (Валидатор, полное попарное сравнение) обнаружил 16 десинхронизаций:

#	Файл(ы)	Проблема	Severity
1	Конституция (RU)	Устаревший номер версии	HIGH
2	Конституция (RU)	Отсутствует нулевая секция (Language Policy) — целая секция	HIGH

3	Конституция (RU)	Отсутствует формула S_{adjusted} и 3 диагностические полосы	HIGH
4	Конституция (RU)	Отсутствует Graduated Activation (GREEN/YELLOW/RED)	HIGH
5	Конституция (RU)	Отсутствует параграф двуязычной стратегии	MEDIUM
6	Конституция (RU)	Отсутствуют веса по умолчанию ($w_1 = w_2 = w_3 = w_4 = 0{,}25$)	MEDIUM
7	Глоссарий (EN)	Устаревшая версия, отсутствуют 14 терминов	HIGH
8	Глоссарий (RU)	Ссылка на несуществующий путь	MEDIUM
9	Глоссарий (EN)	Ссылка на несуществующий путь	MEDIUM
10	Документация	Заголовок указывает одну версию, подвал — другую	HIGH
11	Документация	Файловое дерево содержит несуществующий каталог	MEDIUM
12	Документация	Файловое дерево пропускает 10+ существующих файлов	MEDIUM
13	Протокол загрузки	Устаревший номер версии в подвале	MEDIUM
14	Конституция (обе)	Секция ссылается на несуществующие каталоги	MEDIUM
15	Глоссарий (EN)	Использует устаревшую форму термина	LOW
16	Мета-протокол (обе)	Отсутствует заголовок версии	LOW

Статистика аудита: 8 файловых пар проверено, 6 полностью синхронизированы, 1 частично десинхронизирована, 2 тяжело десинхронизированы. 5 проблем уровня HIGH, 7 уровня MEDIUM, 4 уровня LOW.

IX.2. Правило одновременной синхронизации

На основе аудита сформулировано правило: при создании или обновлении файла, имеющего двуязычную пару, оба файла обновляются в одной операции. Создание одного файла «сейчас» и синхронизация «позже» эксплицитно

запрещено — «позже» превращается в «никогда», а аудит синхронизации становится перманентной нагрузкой на поддержку.

IX.3. Четырёхслойная двуязычная архитектура

По результатам лингвистического анализа (Раздел III) и аудита синхронизации предложена четырёхслойная модель языковой архитектуры:

Слой	Содержание	Примеры	Язык
1	Инвариантная математика	$B = F^{w_1} \cdot E^{w_2} \cdot (1-\sigma)^{w_3} \cdot \Lambda^{w_4}$	Формулы (язык-независимы)
2	TPS-терминология	Jidoka, Andon, Hansei, Yokoten, Round Table	Имена собственные (не переводятся)
3	Процессные термины	Kill-Switch, True North, Blast Radius, Spiral Gap	Английский (операционный)
4	Проектный язык	Описания задач, комментарии, входная документация	Язык команды (RU или EN)

Принцип: английский — для ширины (практические задачи, обнаружение ошибок, инструментальное покрытие), русский — для глубины (теоретическое развитие, математические инновации, концептуальная плотность). Синтез — на Round Table, где обе перспективы сталкиваются и порождают результат, превосходящий каждую в отдельности.

X. CHECK-FIRST PIPELINE

X.1. Мотивация: статья с неправильным форматом

Исследовательская статья (настоящий документ) была первоначально сгенерирована с неправильным форматом: отсутствовали группы комментариев в преамбуле, использовались нежелательные разделители между секциями, формат ключевых слов не соответствовал образцу. Потребовалось полное переформатирование — потери (Muda) [9].

Анализ причин: формат-спецификация была загружена после генерации, а не до. Это инверсия правильного порядка: данные о формате — это входные данные, а не выходной фильтр.

X.2. Архитектура Check-First Pipeline

По результатам анализа разработана методология Check-First Pipeline — обязательный предгенерационный протокол из 7 пунктов:

1. **FORMAT:** загрузить формат-спецификацию (полная преамбула + первая секция) — золотой стандарт формата
2. **FORMULAS:** перечислить ВСЕ формулы статьи. Пересчитать каждую независимо до 50 значащих цифр. НЕ копировать результаты из других статей.
3. **CONSTANTS:** подготовить 50-значные значения для π, φ, $(\pi - 3)^2$ и всех производных констант
4. **SOURCES:** перечислить ВСЕ записи библиографии. Верифицировать каждую: DOI, издатель, год, страницы
5. **STRUCTURE:** определить план секций (римские цифры). Проверить отсутствие перекрытий с существующим корпусом
6. **CONSISTENCY:** проверить соответствие терминологии глоссарию ODTOE (44 термина). Отсутствие противоречий с корпусом
7. **LANGUAGE:** подтвердить, что RU и EN версии будут произведены одновременно

Критическая идея: пункты 1–4 выполняются ДО генерации текста (предгенерационная верификация данных), пункты 5–7 — во время и после генерации (структурная и текстовая верификация). Формульные ошибки — это ошибки входных данных, не выходного текста; их необходимо перехватить до того, как они попадут в текст, а не после. Предгенерационная верификация — это предотвращение; постгенерационная проверка — это переделка, стоящая в 2–3 раза больше.

X.3. Распределение ролей в Check-First

Роль	**Предгенерационные задачи**	**Постгенерационные задачи**
Валидатор	Checks 1–4 (целостность данных)	Checks 5–7 (качество текста)
Строитель	Нет (получает верифицированный пакет данных)	Нет (передаёт Валидатору)
Аналитик	Декомпозиция формул и зависимостей	Проверка AI-маркеров (Check 5)
Когерент	Проверка терминологии (Check 6)	Проверка внутренней согласованности (Check 7)
Визионер	Стратегическая целостность (Check 4)	Нет

XI. ИЗВЛЕЧЁННЫЕ ПРИНЦИПЫ

Полная серия экспериментов породила набор инвариантных принципов — закономерностей, применимых к любой мультиагентной LLM-системе. Наиболее значимые для мультиагентной когерентности:

Принцип холодного старта. При пустой памяти проекта всем агентам присваивается $\Lambda = 0{,}5$ — не ноль (это обнулило бы B по мультипликативной формуле), но и не высокое значение. Это честный априор: «У меня нет проектной памяти, но я несу знание фреймворка.»

Принцип скорректированной когерентности. $S_{\text{adjusted}} = S_{\text{team}} \times \bar{B}$. Никогда не использовать S_{team} в одиночку. Пять агентов могут идеально согласиться на неправильном ответе ($S_{\text{team}} = 1{,}0$, $\bar{B} = 0{,}2$, $S_{\text{adjusted}} = 0{,}2$).

Детектор фантомной когерентности. При $S_{\text{team}} > 0{,}8$ и $\bar{B} < 0{,}4$ — флаг PHANTOM COHERENCE. Не продолжать. Активировать механизм пересмотра предпосылок [1].

Принцип двуязычной маршрутизации. EN для практических задач, RU для теоретической глубины. Английский — для агентной коммуникации, интерфейсных контрактов, отладки. Русский — для теоретического углубления, математических выводов, концептуального исследования.

Принцип enforcement-локализации. Чтение фреймворка $\neq$ применение фреймворка. Bootstrap, говорящий «прочитай и приступай», приводит к 95% solo Builder. Необходимо: (а) identity declaration, (б) мандат делегирования, (в) инвариантная проверка перед каждым action.

Принцип предгенерационной верификации. Для артефактов с определённой спецификацией качества (статьи с 7 проверками, код с тестами) — верификация входных данных ДО генерации. Проверки текстового качества — ПОСЛЕ.

Принцип одновременной синхронизации. Двуязычные файлы создаются и обновляются одновременно. Создание одного файла без пары — техническая задолженность. 16 десинхронизаций найдены в файлах, созданных в той же сессии.

Принцип адверсариального тестирования. Перед развёртыванием нового bootstrap — запустить тестового агента с конкретной XL-задачей и проверить, что он делегирует, а не исполняет. Тестирование стоит один agent call; не тестировать стоит целую сессию.

Принцип формата как входных данных. При генерации любого форматированного артефакта формат-спецификация загружается как первый шаг оператора активации A_F.

XII. МЕГА-ПАТТЕРНЫ

Анализ извлечённых принципов и четырёх экспериментов выявил четыре мега-паттерна — закономерности более высокого порядка:

Мега-паттерн 1: Самоприменение — предельный тест. Когда фреймворк применяется для собственного улучшения (Level 9: $\Psi^* = \Phi(\Psi^*)$ [1]), каждая слабость становится видимой. Lambda-проблема, проблема синхронизации, проблема формата — все были обнаружены, потому что фреймворк использовался на самом себе. 25 агентов, перестраивающих собственный фреймворк — это оператор наблюдения, направленный на самого себя.

Мега-паттерн 2: A/B-эксперименты — действие с максимальной ценностью. Два A/B-эксперимента (RU vs EN, router vs inline) произвели больше применимых данных, чем 20 аналитических отчётов вместе взятых. Когда есть сомнения — экспериментировать, а не аргументировать.

Мега-паттерн 3: Двуязычная архитектура — функция, а не дефект. EN для ширины, RU для глубины. Параллельный запуск обеих групп с синтезом на Round Table даёт результат, превосходящий любой одноязычный подход. Язык — оператор наблюдения, а не нейтральный носитель.

Мега-паттерн 4: Enforcement должен жить там, где он срабатывает. Bootstrap-чеклист в ядре фреймворка (читается всеми) был теоретически видим, но практически игнорировался. Перенесён в мета-протокол (читается только оркестратором), где он реально срабатывает. Авто-загружаемый файл — предельный уровень enforcement: агент не может его пропустить.

XIII. РЕФЛЕКСИЯ СЕССИИ (HANSEI)

XIII.1. Масштаб сессии

Параметр	Прогноз	Факт	Δ
Размер задачи	M (форматирование)	XL (перестройка)	+3 категории
Количество агентов	5–10	80+	$\times 8$
Циклов RT	1–2	10+ (параллельных)	$\times 5$
Скоуп	1 проект	2 проекта + мета-анализ	+2 проекта
Мутаций файлов	10–20	40+	$\times 2$
Архитектурных изменений	0 (поддержка)	3 (двуяз., bootstrap, Section 0)	Непредвиденные

XIII.2. Что работало

A/B-методология стала действием с максимальной ценностью. Перестройка 25 агентами подтвердила самоприменимость фреймворка. Анализ сессии развёртывания конвертировал единичный отказ в системное улучшение (Yokoten [9]). Аудит синхронизации перехватил версионный дрейф до запуска двуязычной архитектуры.

XIII.3. Что требует улучшения

Статьи генерировались ДО применения 7 проверок — инверсия правильного порядка (исправлено: Check-First Pipeline). Двуязычные файлы создавались без одновременной синхронизации — немедленная техническая задолженность (исправлено: принцип одновременной синхронизации). Сессия выросла от M до XL без формальной реклассификации — механизм иерархической реклассификации никогда не был активирован для многокомандной работы. Bootstrap-промпт для сессии развёртывания не был протестирован адверсариально.

XIII.4. Спиральный зазор ($\sim 2\%$)

Нерешённые задачи, питающие следующий виток:

1. Аудит синхронизации обнаружил 16 проблем — не все исправлены
2. Адверсариальный тест bootstrap авто-загружаемого файла не проведён
3. Check-First Pipeline описан, но не закодирован как обязательный шаг в конфигурации
4. Механизм реклассификации сессии не имеет enforcement-хука
5. Реестр констант для предотвращения пропагации ошибки $P_{\text{coll}} = 0{,}61$ не создан
6. Количественный мост между ΔS от языковой неоднородности ($0{,}05$–$0{,}10$) и ΔB от EN-промптов (+48%) не формализован
7. 80+ агентов сессии не были измерены B-диагностикой — фреймворк утверждает, что работает, но не имеет измерений для строгого доказательства

Общий остаток: 7 пунктов $\approx 2\%$ объёма сессии, что согласуется с предсказанием спирального зазора $(\pi - 3)^2 \approx 0{,}020048479550599188058630700199133830130683010990 15$[2].

[2] 50 значащих цифр.

XIV. ОБСУЖДЕНИЕ

XIV.1. Язык как конфигуратор когнитивного пространства

Результаты Эксперимента 2 показывают, что выбор языка промпта не является технической деталью — это выбор оператора наблюдения $\hat{O}_{\text{lang}}$, который определяет, какие конфигурации агент способен актуализировать. Русскоязычные промпты активируют абстрактно-теоретический режим работы LLM (глубина > ширина), англоязычные — практико-операционный (ширина > глубина). Оптимальная стратегия — двуязычная: запуск обеих групп параллельно с синтезом на Round Table.

В контексте ODTOE [1] это означает: один и тот же LLM с разными языковыми промптами — это разные наблюдатели ($\hat{O}_{\text{RU}} \neq \hat{O}_{\text{EN}}$), проецирующие одну и ту же задачу (Ψ) в разные конфигурации ($R_{\text{RU}} \neq R_{\text{EN}}$). Язык — не канал передачи, а линза наблюдения.

XIV.2. Самоорганизация через спиральный зазор

Каждый из четырёх экспериментов выявил феномены, не предусмотренные исходным дизайном: парадокс когерентности (Эксперимент 2), раскол команды маршрутизатором (Эксперимент 3), Lambda-проблема (анализ сессии развёртывания), версионный дрейф (аудит синхронизации). Это проявление спирального зазора $(\pi - 3)^2 = 0{,}02004847955059918805863070019913383013068301099015$[3] — система не замыкается идеально, и остаток питает следующий виток эволюции.

Сессия прошла от M к XL, каждый завершённый цикл порождая остаток ($\sim$2%), становившийся фокусом следующего цикла. Форматирование статей выявило LaTeX-ошибки; ошибки потребовали инструментов конвертации; инструменты потребовали стандартов качества; стандарты потребовали перестройки фреймворка, управляющего стандартами. Это спираль в действии.

XIV.3. Фантомная когерентность как системный риск

Фантомная когерентность (высокий S_{team} при низком $\bar{B}$) представляет собой наиболее опасную конфигурацию мультиагентной системы, поскольку она субъективно ощущается как продуктивность. Все агенты согласны, результаты появляются быстро, конфликтов нет. Но команда синхронизирована вокруг ошибочной модели. Формула S_{adjusted} — необходимый инструмент диагностики, переводящий качественное подозрение в количественный индикатор.

[3] 50 значащих цифр.

XIV.4. Lambda-проблема как общая закономерность

Lambda-проблема не специфична для конкретной реализации — это общая закономерность систем, в которых знание и действие не связаны механизмом enforcement. В образовании это «эффект караоке» [7]: студент знает ответ, но не может применить знание в новом контексте. В организациях это «знание, не ставшее практикой»: регламенты написаны, сотрудники обучены, но поведение не изменилось.

В формализме ODTOE [1] Λ — это не только наличие опыта, но и его применимость. Агент с $\Lambda = 0$ формально мёртв (мультипликативная структура). Решение — не увеличение объёма знаний, а создание механизма их автоматического применения (авто-загружаемый файл как пример enforcement с гарантированным исполнением).

XIV.5. Ограничения

1. B-score основан на самооценке агентов — возможна систематическая инфляция или дефляция. Независимая внешняя метрика качества не использовалась.

2. Все эксперименты проведены на одной LLM-платформе — результаты могут отличаться для других моделей и платформ.

3. Размер выборки (5–10 агентов на эксперимент) недостаточен для строгой статистической значимости; требуется масштабное повторение.

4. Lambda-компонент ограничен отсутствием проектной памяти в экспериментах (холодный старт, $\Lambda = 0{,}5$ по принципу холодного старта).

5. Эксперименты проведены в рамках одной сессии — возможен эффект порядка (Эксперимент 3 мог быть повлиян результатами Эксперимента 2).

6. Операционные пороги (0,5 и 0,7 для сценариев фантомной когерентности) — конструктивный выбор, а не дедукция из аксиом ODTOE.

XV. ЗАКЛЮЧЕНИЕ

1. **Пятеричная структура ролей** формализует различные операторы наблюдения $\hat{O}_r$, и их параллельная работа через Round Table обеспечивает коллективную когерентность, превышающую индивидуальную. 25 агентов (5 RT $\times$ 5 ролей) сжали фреймворк в 2,5 раза и обнаружили ошибку P_{coll} ($0{,}61 \to 0{,}657$), пропущенную 20 аналитическими агентами.

2. **Язык промпта является оператором наблюдения:** EN-промпты дают $\bar{B}$ на 48% выше для практических задач ($S^{\text{EN}}_{\text{adjusted}} = 0{,}473$ vs $S^{\text{RU}}_{\text{adjusted}} = 0{,}335$); RU-промпты обеспечивают превосходство в теоретической глубине (φ-взвешенная S_{team}, Graduated Activation). Оптимальна двуязычная архитектура.

3. **Формула** $S_{\textbf{adjusted}} = S_{\textbf{team}} \times \bar{B}$ обнаруживает фантомную когерентность: RU-группа с $S_{\text{team}} = 0{,}970$ выглядит здоровее EN-группы ($S_{\text{team}} = 0{,}920$), но S_{adjusted} инвертирует картину ($0{,}335 < 0{,}473$). Все 10/10 агентов независимо подтвердили необходимость этой метрики.

4. **Маршрутизация в ядре** эффективнее отдельного файла маршрутизации: $S^{\text{inline}}_{\text{adjusted}} = 0{,}513$ vs $S^{\text{router}}_{\text{adjusted}} = 0{,}470$. Маршрутизатор расколол команду (1 EN + 4 RU), встроенная маршрутизация позволила самоопределение (4 EN + 1 RU).

5. **Lambda-проблема** (знание без применения) привела к 95% solo Builder при XL-классификации. Решение: трёхуровневый enforcement (авто-загружаемый файл + мета-протокол + память). Ключевой принцип: enforcement должен жить там, где он срабатывает, а не там, где его удобно описать.

6. **Лингвистический анализ 12 файлов** выявил системную неоднородность (Mura): ядро (547 строк) на EN, операционный слой (861 строка) на RU. Токеновая неэффективность кириллицы (1,5–2,5×), разрыв бенчмарков (5–15% MMLU/MGSM/XCOPA), сила императивных конструкций EN.

7. **Check-First Pipeline** предотвращает 100% ошибок формата: формат-спецификация загружается как первый шаг A_F, формулы пересчитываются независимо до генерации текста. Предгенерационная верификация — предотвращение; постгенерационная проверка — переделка.

8. **Набор извлечённых принципов** и **4 мега-паттерна** извлечены из сессии с 80+ агентами. Спиральный зазор: 7 нерешённых пунктов ($\sim$2%), согласующийся с $(\pi - 3)^2 \approx 0{,}02$, питают следующий виток.

СПИСОК ЛИТЕРАТУРЫ

1. Панкратов А. С. Наблюдатель-зависимая теория всего (ODTOE): аксиома, постулаты и математический формализм // Препринт. — 2025.

2. Arnold V. I. Mathematical Methods of Classical Mechanics. — New York: Springer-Verlag, 1978. — 462 p.

3. Панкратов А. С. Когерентность: от индивидуальной к коллективной // Препринт. — 2025.

4. Панкратов А. С. Когерентное образование: теория и методология построения обучающих систем на основе ODTOE // Препринт. — 2026.

5. Панкратов А. С. Когерентное образование II: энтропийная диагностика, адаптивная персонализация и эволюционный отбор // Препринт. — 2026.

6. Панкратов А. С. Перенос методов управления распределёнными системами в проект когерентного термоядерного реактора ODTOE: структурный анализ параллелей между ирригационной и плазменной инженерией // Препринт. — 2026.

7. Кибальников С. В. SKW матрица — «эффект караоке» в образовании и высокотехнологичном производстве // Электронный ресурс. Режим доступа: http://kibalnikov.com/wordpress/?p=57

8. Панкратов А. С. Тороидальная топология реальности: π-вращение, φ-скачки и вложенные торы // Препринт. — 2026.

9. Ohno T. Toyota Production System: Beyond Large-Scale Production. — Portland: Productivity Press, 1988.

10. Hofstadter D. R. I Am a Strange Loop. — New York: Basic Books, 2007. — 412 p.