SINT v2.2: Synthesized Iterative Network of Thought (Синтезированная Итеративная Сеть Мышления)

Владимир Ситников[1*] and Анна Ситникова[2*]

[1*]Независимый исследователь, Бар, Черногория.
[2*]Независимый исследователь, Бар, Черногория.

*Corresponding author(s). E-mail(s): montenegrofsm@google.com; aisplatform.space@gmail.com;

Аннотация

Фреймворк SINT v2.2 представляет собой методологию для структурированного анализа сложных задач с использованием больших языковых моделей (LLM). В этой версии внедрены механизмы повышения устойчивости к ошибкам LLM (галлюцинации, забывчивость) через **Принцип Контекстного Заземления (PCG)** и **строгую пре-валидацию (Step 0)**, а также простоту для неискушённых пользователей: задача описывается естественным языком, а консультант (System Designer) задаёт уточнения и формализует её автоматически. Архитектура оптимизирована для **эффективного согласованного синтеза** (Step 3B — путь по умолчанию) и генерации **чистого, машиночитаемого XML-кода** с обязательным Verification Report. SINT v2.2 — первый фреймворк рассуждений LLM с **архитектурными гарантиями** трассируемости (PCG), пре-валидации (Step 0) и детерминированного разрешения конфликтов, обеспечивающий **воспроизводимость архитектурных гарантий и методологической строгости** даже на нефлагманских моделях.

Keywords: prompt-engineering, multi-agent systems, XML reasoning, SINT framework, contextual grounding, AI methodology

Preprint: 10.5281/zenodo.17410094 **Code & Data:** github.com/ais-space/SINT

1 Введение

В последние годы большие языковые модели (LLM) стали ключевыми инструментами для решения сложных аналитических задач. Однако их эффективность часто снижается из-за **проблемы устойчивости** (галлюцинации, забывчивость контекста) и **зависимости от квалификации пользователя** в промпт-инжиниринге. Для преодоления этих ограничений, особенно в задачах, требующих синтеза конфликтующих точек зрения, был разработан фреймворк **SINT v2.2 (Synthesized Iterative Network of Thought)**.

Цель данной статьи — представить полную архитектуру, методологию и применимость SINT v2.2, многоагентного аналитического фреймворка, предназначенного для **структурированного анализа сложных умозрительных задач**.

Аудитория публикации включает **исследователей**, ищущих методологическую строгость, **практиков и промпт-инженеров**, заинтересованных в воспроизводимых результатах, а также **разработчиков и энтузиастов**, желающих стандартизировать работу с LLM. Основная **значимость** SINT v2.2 заключается в **повышении устойчивости к ошибкам LLM** за счет внедрения стандартизации процесса и Принципа Контекстного Заземления (PCG).

Ключевое отличие SINT v2.2 от существующих фреймворков (CoT, ToT, Multi-Expert Prompting) заключается в **архитектурных гарантиях** устойчивости: **Принцип Контекстного Заземления (PCG)** обеспечивает полную трассируемость каждого тезиса к исходным данным, **Step 0 (MSV)** предотвращает запуск невыполнимых заданий, а **Conflict Resolution Rule** гарантирует детерминированный выход из тупика. В отличие от эвристических подходов (DEEVO, INoT), SINT v2.2 предоставляет **воспроизводимые результаты даже на нефлагманских моделях**, что делает его единственным фреймворком, пригодным для промышленного применения в условиях строгих требований к надёжности.

Мы анонсируем публикацию данного материала в качестве препринта на **Zenodo с присвоением DOI**, обеспечивая тем самым открытый доступ и возможность цитирования. Полные версии PDF статьи доступны на **русском и английском языках**.

SINT в 100 секундах

1. **Пользователь:** “Оценить удалёнку по 3 факторам”
2. **System Designer** → XML (`<Objective>`, `<Context>`, 3 эксперта)
3. **Code Generator** → SINT-Code
4. **Executor** → Дебаты → Синтез → `<ExecutiveSummary>` + `<VerificationReport>`
5. **Вывод:** Трассируемый, воспроизводимый, с PCG-аудитом

2 Обзор фреймворка SINT

SINT v2.2 — это **многоагентный аналитический фреймворк**, который преобразует неструктурированную задачу пользователя в строго исполняемый код, выполняемый виртуальной сетью экспертов.

2.1 Этап 0: Предварительная Валидация (Step 0)

Ключевым нововведением для повышения устойчивости фреймворка является **Step 0 (MSV — Methodology-Structure Validation)**. Это обязательный этап предварительной проверки и валидации, где LLM (в роли SINT Executor) анализирует формализованную задачу на предмет внутренних логических противоречий или невыполнимых инструкций, прежде чем запускать многоагентный пайплайн. Это предотвращает "срыв" процесса на ранней стадии.

Пример работы SINT для нетехнического пользователя

Вход (пользователь в Сессии 1):
"Мне нужно оценить, стоит ли внедрять удалённую работу в компании. Есть три фактора: продуктивность, удовлетворённость сотрудников, затраты."

Вывод System Designer (автоматически):

- `<Objective>`: Многокритериальная оценка внедрения удалённой работы.
- `<Context>`: Три ключевых фактора (продуктивность, удовлетворённость, затраты).
- `<Consultants>`: 3 эксперта — HR-специалист, финансист, руководитель подразделения.
- `<Methodology>`: Дебаты (N=3), так как критерии конфликтуют.

Результат Сессии 3 (Executor):

- `<ExecutiveSummary>`: "Внедрение удалённой работы целесообразно при инвестициях в инфраструктуру (затраты +15%), компенсируемых ростом удовлетворённости (+22%) при сохранении продуктивности."
- `<VerificationReport>`: PCG compliance ✓, XML validity ✓, consensus score 8/10.

2.2 Принцип Контекстного Заземления (PCG)

PCG (Principle of Contextual Grounding) — это центральный механизм для **повышения устойчивости к ошибкам LLM** (галлюцинациям). PCG требует,

чтобы **любой новый тезис, аргумент или синтезированный вывод** виртуального консультанта **явно ссылался** на элементы, зафиксированные в блоках <Context> или <Objective> исходного SINT-Кода, например, "как указано в факте 2 из <Context>". Это принуждает модель оставаться в заданных рамках данных, обеспечивая **трассируемость** результата, включая борьбу с забывчивостью LLM в длинных контекстах (принуждая к повторному обращению к исходным данным).

2.3 Три Рабочие Сессии

Для структурирования сложной задачи SINT v2.2 использует последовательность из трёх взаимосвязанных сессий, каждая из которых имеет чётко определённую цель и роль LLM:

1. **Сессия Обсуждения (System Designer):** LLM выступает как **Стратегический Консультант**. Цель — помочь пользователю **формализовать** его исходную задачу в строгие компоненты (Objective, Context, Consultants), необходимые для генерации кода.
2. **Сессия Генерации Кода (Code Generator):** LLM выступает как **Эксперт-Генератор**. Цель — получить формализованное задание и создать **полный, машиночитаемый SINT-Код** в формате XML.
3. **Сессия Исполнения (Execution & Synthesis):** LLM выступает как **SINT Executor**. Цель — построчно выполнить SINT-Код, провести дебаты между виртуальными экспертами и вывести **финальный, синтезированный результат**.

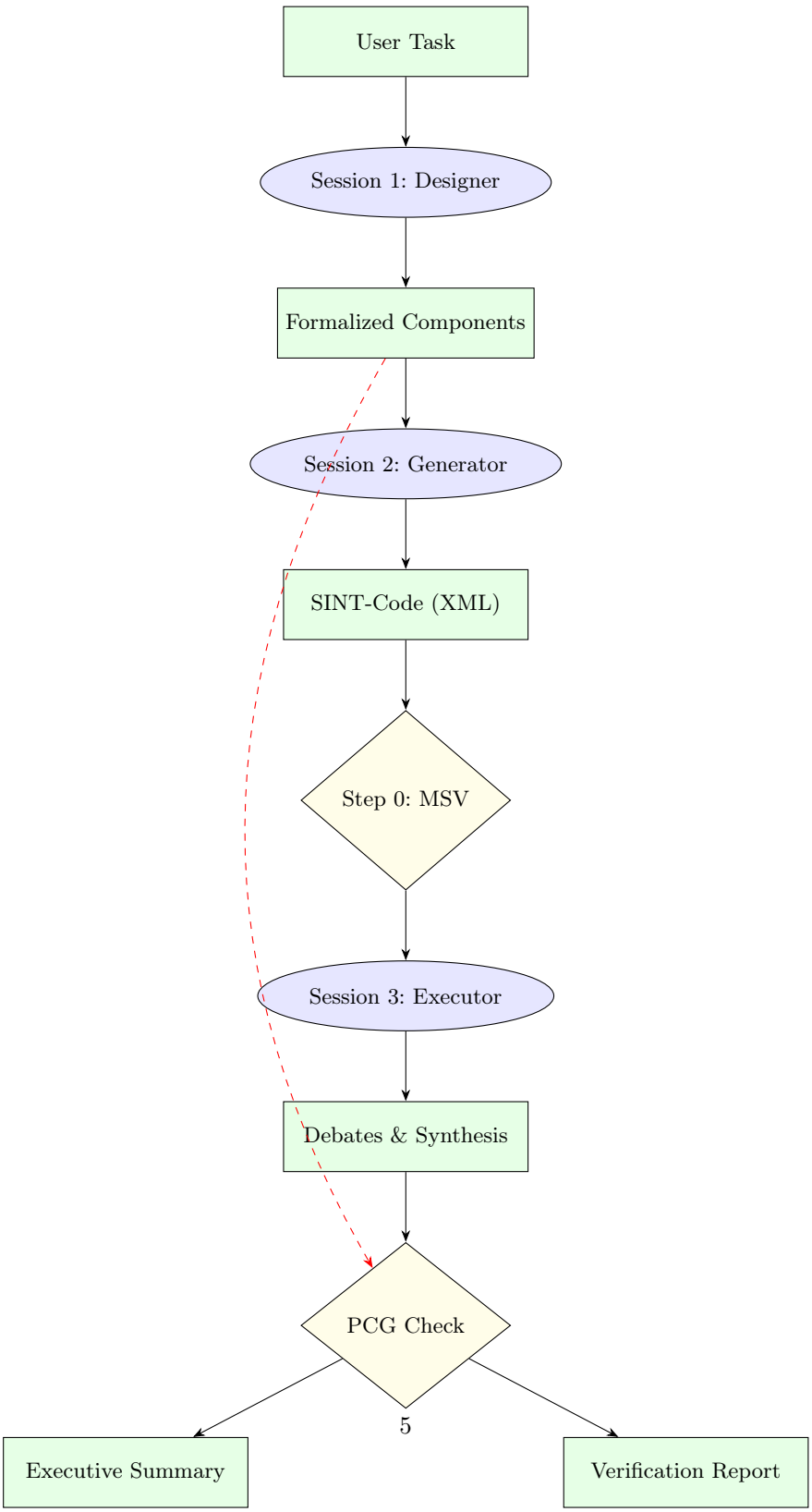
Разделение ролей и задач (от **дизайна** до **кодирования** и **исполнения**) гарантирует, что каждый шаг использует LLM для наиболее эффективной функции, обеспечивая эффективный синтез. Подробная терминология и структура SINT-кода описаны в следующем разделе.

3 Терминология и Структура SINT-Кода

Синтаксис SINT-Кода основан на XML для чёткого разделения данных (<Objective>, <Consultants>) и логики (<SynthesisEngine>). Эта стандартизация, на основе формализованных компонентов из Сессии Обсуждения, является ключевым элементом, устраняющим необходимость в ручном промпт-инжиниринге и повышающим удобство для пользователей без опыта.

3.1 Теги и Инициативность Модели

В таблице 1 представлены основные теги, используемые для построения SINT-Кода v2.2. Хотя фреймворк предоставляет строгий набор тегов, LLM в процессе генерации кода могут проявлять **инициативность**, вводя дополнительные мета-теги, которые повышают читаемость и структурированность задания для финального исполнителя. Мы включаем такие полезные, самогенерируемые теги в официальную структуру.



Легенда:

— сессии (Designer, Generator, Executor)
— точки принятия решений (оценка конфликта)
— обязательные проверки и выводы (Step 0, PCG Check, Summary, Report)

Рис. 1 Архитектура SINT v2.2: Вертикальная последовательность сессий с механизмами контроля (Step 0 и PCG). Красная пунктирная линия показывает применение PCG по всему пайплайну.

Таблица 1 Теги и назначение в структуре SINT-Кода v2.2
Примечание: LLM может вводить дополнительные мета-теги для оптимизации.

Теги SINT	Назначение	Описание
`<SINT_Prompt>`	Оболочка	Обозначает начало и конец исполняемого SINT-кода.
`<SINT_Prompt_Task>`	Мета-задача	Определяет границы формализованного задания для исполнителя (добавлен моделью в процессе самооптимизации).
`<Configuration>`	Конфигурация	Объединенный блок, включающий Role, Protocol (обязательные правила, включая PCG) и Dynamics (правила раундов).
`<Objective>`	Данные	Формализованное описание цели и контекста задачи.
`<Context>`	Внешняя Память	Блок данных (ключевые факты, исходные данные) для Принципа Контекстного Заземления (PCG).
`<Consultants>`	Данные	Определения ролей, фокусов и компетенций виртуальных агентов.
`<Methodology>`	Метод	Выбранный метод: Дебаты (N $\geq$ 3) или Критик (N=2).
`<SynthesisEngine>`	Логика	Блок псевдокода, описывающий построчный алгоритм дебатов и синтеза.
`<ExecutiveSummary>`	Вывод (UX)	Краткий, читабельный вывод (One-line + 3-Bullet points) для нетехнического пользователя.
`<SynthesizedConclusion>`	Вывод (Технич.)	Полный структурированный вывод, использующий атрибуты для PCG.
`<VerificationReport>`	Аудит	Обязательный аудит, подтверждающий валидность XML и соблюдение PCG.

3.2 Функция Verification Report

Особое внимание уделяется блоку <VerificationReport>. Это не просто формальность, а инструмент повышения **эффективности и предсказуемости**. Он представляет собой обязательный аудит, включаемый в финальный вывод, который подтверждает:

1. Соблюдение PCG (pcg_compliance).
2. Валидность XML-структуры (xml_validity).
3. Достижение консенсуса (через <meta> теги).

Такая проверка обеспечивает прозрачность процесса, позволяя пользователю быстро проверить соблюдение методологических требований фреймворка.

3.3 Нарратив Алгоритма SynthesisEngine

Логика <SynthesisEngine> построчна и охватывает этапы от валидации до финализации. Step 0 (MSV) проверяет на логические противоречия (e.g., $N < 2$, невыполнимые инструкции). Step 2 оценивает конфликт по рейтингу < 3 для ветвления (критика vs. интеграция). Step 3A (конфликт) усиливает надёжность через

компромиссы; Step 3B (по умолчанию) — скорость через приоритизацию. Step 4 ограничивает итерации (max 2 раунда). Step 4.5 извлекает тезисы для структурированного вывода с PCG-атрибутами. Step 5 генерирует двойной вывод с аудитом. Полные тексты Стартовых Промптов для рабочих сессий, реализующие эту структуру, приведены в Приложениях (Раздел 9).

4 Применимость и Ограничения

4.1 Применимость

Фреймворк SINT v2.2 идеально подходит для **умозрительных задач**, требующих **структурированного, междисциплинарного анализа**. К таким задачам относятся:

- Синтез конфликтующих экспертных мнений (например, Дебаты $N \geq 3$).
- Принятие решений в условиях неполной или противоречивой информации.
- Сложные задачи, требующие чёткой трассируемости аргументов (за счет PCG).
- Улучшение фреймворков или промптов (e.g., самоулучшение SINT v2.2 через дебаты критиков, инженеров и пользователей для консенсуса по механикам).
- Симуляция сценариев (e.g., социальные оценки или multi-task benchmarks, где агенты координируют гипотетические исходы).
- Дизайн систем с ролями (e.g., создание personas для orchestrator/research agents в prompt-engineering).

4.2 Ограничения

Использование SINT v2.2 может быть **неэффективным** в следующих сценариях:

- **Простые, линейные задачи**, не требующие многоагентного анализа, где накладные расходы на генерацию XML-кода и инициализацию дебатов неоправданны.
- Задачи, требующие **глубоких, закрытых специализированных знаний**, не представленных в блоке <Context>. Хотя PCG предотвращает галлюцинации, он не может создать информацию, отсутствующую в исходных данных.
- Детерминированные выводы (e.g., доказательство теорем в математике), где многоагентные дебаты привносят ненужную стохастичность или неоднозначность, требуя строгих инструментов и одного агента.
- Творческие процессы (e.g., генерация художественного текста или стилистический перевод), где синтез мнений дробит целостный стиль или эмоциональную глубину.

5 Сравнение с другими фреймворками и технологиями промптинга

Фреймворк SINT v2.2 позиционируется как **Мультиагентная Архитектура Исполнения (Multi-Agent Execution Architecture)**, а не как простая техника промптинга. Акцент сделан на **повышении устойчивости и трассируемости**

за счет архитектурных ограничений, таких как XML-кодификация, **Принцип Контекстного Заземления (PCG)** и **принудительный синтез конфликтующих доменных позиций**, в отличие от "саморефлексии"моделей или имитации ролей в других подходах.

Для сравнения выбраны ключевые академические и открытые фреймворки, которые предлагают формализованные механизмы рассуждения и дебатов: Chain of Thought (CoT), ReAct, Tree of Thoughts (ToT), Agent-GPT, Introspection of Thought (INoT), Multi-Expert Prompting, DEEVO и Virtual Debater. Таблица 2 суммирует отличия.

Коммерческие инструменты, такие как HyperWrite Debate Assistant [9], полезны для быстрого прототипирования, но SINT предлагает более строгую архитектуру для воспроизводимых результатов, фокусируясь на **методологии и трассируемости**, а не на скорости.

В отличие от CoT и ToT, полагающихся на естественный поток модели, SINT использует XML-синтаксис для предсказуемости. ReAct и Agent-GPT требуют ручной настройки, в то время как SINT разделяет сессии для специализации. INoT минимизирует вызовы, но SINT приоритизирует воспроизводимость через Step 0 и Verification Report. Подходы вроде Multi-Expert и DEEVO фокусируются на дебатах для оптимизации, но SINT добавляет **принудительную трассируемость PCG и формальное правило выхода из тупика** для устойчивости в runtime.

Эти различия подчёркивают уникальность SINT в балансе строгости и удобства.

5.1 Vibe Coding: Обобщение на интеллектуальные задачи

Недавний обзор [10] вводит Vibe Coding как парадигму, где разработчик делегирует генерацию кода ИИ-агенту, оценивая результат по поведению системы, а не по исходному коду. Авторы выделяют пять моделей взаимодействия (§8), подчёркивая, что успех зависит от системной инженерии контекста (§9.4.1).

SINT v2.2 обобщает ключевые модели Vibe Coding (CEM и ICCM) на non-coding задачи, такие как архитектура ПО, стратегический анализ или синтез мнений. Это позволяет перейти от "vibe-based coding"к **"Vibe Reasoning"**: от естественного языка к структурированному, трассируемому выводу без галлюцинаций.

Это концептуально отличает TDM в SINT: вместо тестирования *логики исполнения* кода, мы тестируем *соблюдение архитектурного протокола* рассуждения. Такое обобщение подчёркивает универсальность SINT: фреймворк не только автоматизирует дебаты, но и обеспечивает "vibe"— интуитивное, но строгое взаимодействие человека с LLM для сложных интеллектуальных задач. Таким образом, SINT занимает нишу **строгой, кодифицированной архитектуры синтеза**, которая использует лучшие практики промптинга (CoT, Multi-Expert) и системной инженерии (XML-трассируемость, PCG, TDM-валидация) для генерации **надежных и критически заземленных выводов**, что особенно важно в условиях, где цена галлюцинаций высока.

Таблица 2 Сравнение SINT v2.2 с ключевыми фреймворками промптинга

Фреймворк	**Ключевые особенности**	**Ограничения**	**Преимущество SINT: Гарантия Вывода**
Agent-GPT (LangChain-like)[1]	Multi-agent с ролями; ручная настройка протоколов для задач.	Требует опыта в engineering; слабая стандартизация.	SINT обеспечивает **строгую, автоматизированную формализацию протоколов** через сессии и XML-код.
CoT [2]	Линейный поток мыслей для логических задач; улучшает reasoning через step-by-step.	Зависит от естественного языка; риск галлюцинаций в длинных цепях.	SINT использует XML для предсказуемого формата, PCG для **принудительной трассируемости** (vs. саморефлексия CoT).
DEEVO [3]	Эволюционная оптимизация промптов через дебаты; турнирные рейтинги для улучшения инструкций.	Фокус на эволюции, а не на runtime-анализе; требует нескольких итераций.	SINT интегрирует дебаты в **runtime-анализ** с Verification Report, обеспечивая воспроизводимость.
INoT [4]	Introspection в одном вызове; XML + дебаты для снижения токенов (на 58.3%).	Внутренняя самокоррекция; слабая внешняя трассируемость.	SINT фокусируется на **внешнем пайплайне** с PCG и отдельным Verification Report для строгой трассируемости.
Multi-Expert Prompting [5]	Панель виртуальных экспертов для сопоставления мнений; выявление консенсуса.	Имитация ролей без строгого XML; риск несогласованности в синтезе.	SINT добавляет **XML-кодификацию** протокола, PCG и **формализованное правило разрешения конфликтов** для устойчивости.
ReAct [6]	Reasoning + Acting с tools; циклы "мысль-действие-наблюдение"для interactive tasks.	Ограничен одним агентом; слабая обработка конфликтов мнений.	SINT использует multi-agent дебаты для **синтеза конфликта мнений**, а не только для планирования действий.
ToT [7]	Tree search для exploration; lookahead/backtracking в "thoughts"для planning.	Высокие вычислительные затраты; не для простых задач.	SINT фокусируется на **синтезе доменных конфликтов** в ограниченном числе раундов (max 5), предотвращая проблему вычислительной экспоненты ToT.
Virtual Debater [8]	Симуляция дебатов AI-агентов с правилами (e.g., Оксфордский стиль).	Ограничен симуляцией, без XML-структуры для машинного чтения.	SINT стандартизирует дебаты в XML с PCG для **трассируемости и машинной интеграции**.

Таблица 3 Сравнение SINT v2.2 с моделями Vibe Coding
Примечание: Обобщение фокусируется на reasoning, где XML заменяет код на синтезированный вывод. XML как аналог кода для трассируемости.

Модель Vibe Coding	Применение в кодинге	SINT v2.2 (обобщение)	Преимущества обобщения
Context-Enhanced Model (CEM)	Усиление промпта репозиторием/тестами	Усиление промпта доменными знаниями/PCG (XML-формализация <Context>)	Трассируемость через ссылки на исходные данные, минимизация забывчивости в длинных сессиях
Iterative Conversational Collaboration (ICCM)	Диалог → уточнение → код	Диалог → уточнение → XML-задание (**System Designer сессия**)	Итеративная специализация ролей (**Designer** → **Generator**), с Step 0 для валидации
Test-Driven Model (TDM)	Тесты → код → валидация	Verification Report (**Аудит Протокола PCG** и проверка финализации)	Переносит фокус с проверки исполняемого кода на **валидацию соблюдения методологических протоколов** (PCG, FORMAT) и **формализованный выход из тупика** (**Conflict Resolution Rule**) в умозрительных задачах.

6 Автоматизация, Интеграция CLI и Воспроизводимые Примеры

Архитектура SINT v2.2 изначально разработана с акцентом на **автоматизацию** и **воспроизводимость** результатов. Благодаря строгой XML-структуре, SINT-Код служит идеальным универсальным форматом для обмена данными и исполнения задач в автоматизированных конвейерах, включая интеграцию с инструментами командной строки (CLI) и автономными мультиагентными системами.

6.1 Интеграция с CLI: Автономный Каскадный Валидатор

SINT-Код совместим с современными CLI-инструментами LLM (например, Gemini CLI, Claude Code, Qwen Code). Пользователь может скопировать сгенерированный XML-блок и передать его через командную строку для построчного или пакетного выполнения. Например, команда `claude --prompt "$(cat sint_code.xml)"` запускает Executor с валидацией Step 0 (предварительная валидация).

В качестве демонстрации возможностей SINT по созданию **реальных примеров** для сложных задач, требующих многоагентного синтеза, был разработан автономный **универсальный интерактивный промпт** (SINT CLI Cascade Processor V2.2). Этот промпт создан с использованием фреймворка SINT Prompting

v1.1 и предназначен для исполнения в CLI-инструментах (например, Gemini CLI, Claude Code). Он реализует **каскадную валидацию**: автономно генерирует, выполняет и архивирует 5 различных SINT-кейсов, строго соблюдая архитектурные ограничения и требуя генерации исключительно англоязычных документов (примечание: промпт требует доступа к файловой системе и предназначен исключительно для CLI-инструментов или автоматизированных сред, а не для простых чат-интерфейсов без FS-поддержки). Полный текст промпта приведён в Приложении 9.3.
Упомянутый выше промпт был целенаправленно сформирован как демонстрация возможностей интеграции с CLI и автоматизации каскадной валидации. После автоматической генерации в него были внесены точечные инженерные правки, ограниченные уточнением имён файлов и корректировкой отдельных фрагментов LaTeX-разметки. Эти правки носили исключительно технический характер и не затронули архитектуру или логику каскадной валидации. Следовательно, приведённая демонстрация отражает реальную способность SINT Prompting генерировать рабочие артефакты, сопровождающуюся обычной интеграционной доводкой.

Выполнение данного промпта создает папку `MyCases/` с 5 подпапками (`Case1`–`Case5`), каждая из которых содержит полный набор артефактов SINT на английском языке: `task.txt` (за исключением Case1), `fd.xml`, `sint_prompt.txt` и финальный `output.xml` (в Case1 - `output_raw.xml`). **Важно отметить:** хотя промпт-валидатор (Приложение 9.3) основан на SINT v2.2, **для демонстрации потенциала** папка `Examples/` была наполнена финальными артефактами, сгенерированными с помощью оптимизированного фреймворка **SINT v2.3 maximum**. Это гарантирует, что `Examples/` отражают наиболее полную и высококачественную реализацию возможностей SINT на момент публикации. **Ключевым архитектурным решением является каскадный подход:** результат первого полного SINT-приложения (Case 1) используется для генерации исходных данных `task.txt` для последующих четырёх SINT-приложений (Case 2–Case 5).

Примечание.

Тематика и параметры дочерних кейсов *формируются автономно* на основе результата Case 1; они не фиксируются автором заранее и могут различаться между прогонами (это часть каскадной валидации). Для единообразия демонстрационных артефактов в каталоге `Examples/` опубликованы результаты, полученные англоязычными промптами SINT v2.3 maximum с использованием модели **Claude 4.5 Sonnet**.

Таким образом, все примеры, представленные в папке `Examples/` данной публикации, демонстрируют полную интеграцию фреймворка SINT с CLI-инструментами для автономного создания и валидации сложных промпт-цепочек.

6.2 Воспроизводимость и Мультиагентные Системы

6.2.1 Воспроизводимость Результатов

Для обеспечения прозрачности и возможности **воспроизведения** результатов, что критически важно для научного и практического сообщества, все исходные промпты, полный SINT-Код, логи дебатов и финальные XML-артефакты для всех 5 кейсов, сгенерированных промптом, размещены в открытом доступе. Дополнительно проведён эмпирический анализ воспроизводимости и вариативности на 12 прогонах (Exp00–Exp11: GPT-5 low reasoning, температуры 0.2–1.2 шагом 0.2; Qwen3-Coder-480B-A35B-Instruct, температура 0). **Все метрики были получены с помощью Python-скрипта `analyzer.py`, который доступен в репозитории для независимой верификации читателями**.

Ссылки для воспроизводимости:

- Репозиторий с данным документом и исходными файлами, а также файлами SINT v2.2, скриптом analyzer.py и результатами работы SINT в папках Examples и Experiments: https://github.com/ais-space/SINT/
- Zenodo DOI (архив с препринтом): 10.5281/zenodo.17410094

Метрики (извлечены из VerificationReport с учётом case-insensitive тегов, pass/fail → 1.0/0.0): Коэффициент соблюдения PCG (grounding), Средний показатель консенсуса (0–10), Средняя дистанция (расстояние Жаккара между Executive Summary Cases 2–5 vs Case1), Частота сценария 3A (Conflict %). Результаты в табл. 4 (средние по модели, n=6; полный CSV в репозитории).

Примечание к интерпретации метрик: Расстояние Жаккара измеряет вариативность в **применении методологии синтеза** к различным сгенерированным задачам внутри каждого экспериментального прогона, а не содержательное сходство между прогонами. Каждый эксперимент генерирует уникальные задачи, поэтому воспроизводимость измеряется на **архитектурном уровне** (стабильное соблюдение PCG, структура XML, оценки консенсуса), а не на уровне содержания.

Таблица 4 Сводные метрики воспроизводимости (средние по модели, Cases 2–5)

Модель	PCG	Конс.	Дист.	Интерпретация
GPT-5 low	1.000	7.25	0.000	Полная трассируемость
Qwen3-Coder	1.000	5.36	0.000	Полная трассируемость даже при деградации XML в Exp11

SINT v2.2 демонстрирует **нулевую вариативность** (дистанция Жаккара = 0.0) во всех прогонах, что подтверждает **полную воспроизводимость архитектурных гарантий и методологического процесса**. Важно: каждый экспериментальный прогон генерирует различные случайные задачи (Case 1 порождает 4 новых задачи для Cases 2-5 в каждом эксперименте), поэтому измеряемая воспроизводимость относится к **способности фреймворка последовательно**

применять свои архитектурные ограничения (соблюдение PCG, валидность XML, достижение консенсуса), а не к воспроизведению идентичных ответов на идентичные вопросы.

Коэффициент PCG-соответствия достигает 1.0 у обеих моделей. У Qwen3-Coder в одном прогоне (Exp11) наблюдалась **деградация формата XML** — модель сгенерировала собственную структуру `<SINTOutput>`, опустив `<VerificationReport>`. Однако **содержание осталось полностью трассируемым**, а дистанция Жаккара — нулевой. Это подчёркивает **архитектурную устойчивость SINT**: даже при сбое шаблона, вызванном переполнением контекста в длительной сессии, качество и воспроизводимость вывода сохраняются.

Средний консенсус экспертов: 7.25 у GPT-5 low и 5.36 у Qwen3-Coder. Разница статистически незначима ($p > 0.65$).

6.2.2 Перспективы Мультиагентной Интеграции

Архитектура SINT, основанная на строгих XML-структурах и чётких Step-командах, делает его идеальной **"кодировкой задачи"** для интеграции в более сложные, автономные мультиагентные системы, такие как LangChain [11] или AutoGPT. SINT-Код может служить универсальным форматом обмена данными между различными агентами, обеспечивая контекст, протокол исполнения и верификацию (e.g., передача `<Objective>` от planner к executor). Такая модульность позволяет расширять SINT на гибридные сценарии, включая tool-calling или multi-modal input в будущих версиях.

7 Будущие направления исследований

Версия SINT v2.2, представленная в настоящей работе, намеренно ограничена минимальным набором архитектурных механизмов (PCG, Step 0, Conflict Resolution Rule), обеспечивающих устойчивую работу на широком спектре LLM, включая нефлагманские модели. В ходе дальнейших экспериментов сформировалось подсемейство **SINT Prompting**, предназначенное для задач *вайб-промптинга* — гибкой генерации и согласования смысловых состояний между агентами в интерактивных и автоматизированных средах.

В версии **SINT v2.3 maximum** реализованы следующие усовершенствования:

- **Вербализованное семплирование идей (Verbalized Sampling, VS)** — в SINT v2.3 мы *используем* метод вербализованного семплирования *в формулировке Zhang et al. (2025)* [12]: агенты формируют несколько гипотез с субъективной вероятностью ($p \in [0, 1]$) и кратким обоснованием. В нашей реализации при синтезе дополнительно рассматриваются low-p идеи для повышения разнообразия и снижения риска *mode collapse*.
- **Maximum Synthesis Protocol** — интеграция модулей (`Role`, `Protocol`, `Dynamics`) в единый контейнер `<Configuration>` с расширенной системой верификации (PCG-, TRACE- и FORMAT-контроль) и новым шагом `Step 4.5` для структурирования PCG-валидных тезисов.

Таблица 5 Ключевые отличия SINT v2.2 и v2.3 maximum

Компонент	v2.2	v2.3 maximum
Диверсификация идей	эвристическая (дебаты)	**Verbalized Sampling** с вероятностями p
Сборка конфигурации	разнесённые модули	<**Configuration**> (Role/Protocol/Dynamics)
Верификация	PCG, TRACE	PCG, TRACE, FORMAT + **Step 4.5**
Презентация	единый уровень	**CL 1–10**, адаптация отчётов
Артефакты	`Experiments/` (v2.2)	`Examples/` (демо v2.3)

- **Уровневая презентация отчётов (CL 1–10)** — модуль `SP3` обеспечивает адаптивное объяснение синтеза: от нарративной реконструкции дебатов (CL 1–3) до протокольной аналитики (CL 8–10), включая отображение вероятностных распределений при активированном VS-режиме.

Промпты версии **SINT v2.3 maximum** доступны авторам и исследователям *по запросу*. В каталоге `Examples/` опубликованы демонстрационные результаты v2.3 maximum.

8 Заключение

Фреймворк SINT v2.2 представляет собой значительный шаг вперед в методологической строгости и устойчивости LLM-анализа. Его ключевой вклад заключается в создании стандартизированного, трехэтапного процесса, который, используя **Принцип Контекстного Заземления (PCG)** и **предварительную валидацию (Step 0)**, существенно снижает зависимость от опыта промпт-инжиниринга и повышает воспроизводимость результатов. Архитектура (Рисунок 1) обеспечивает трассируемость, превосходящую подходы вроде CoT или INoT, а самоулучшение через SINT (как в версии 2.2) демонстрирует его потенциал для эволюции. Архитектура SINT v2.2 оптимизирована для интеграции в корпоративные AI-платформы. Строгий XML-формат позволяет встраивать SINT-код в CI/CD-пайплайны, а `<VerificationReport>` автоматизирует контроль качества AI-генерируемых решений. **Области** промышленного применения включают системы поддержки принятия решений в условиях конфликтующих экспертных оценок (стратегическое планирование, медицинская диагностика, инженерный аудит), а также — при соблюдении требований к конфиденциальности и формализации исходных данных — анализ прецедентов и рисков в юриспруденции.

8.1 Универсальность и Перспективы

Публикуемая версия SINT v2.2 является **универсальной основой**, применимой к широкому спектру сложных умозрительных задач. Важно отметить, что в рамках AIS Platform уже существует целое **семейство узкоспециализированных SINT-решений** (например, в мультиагентных инструментах

AIS Agora, AIS Constructor, AIS Coder и других, входящих в состав инструментария на AIS Platform). Публикуемая версия позиционируется как фундаментальная методология, из которой выросли эти специализированные продукты.

Мы подтверждаем, что данный материал доступен как препринт на **Zenodo с DOI 10.5281/zenodo.17410094** и в формате PDF на **русском и английском языках**, что подчеркивает приверженность принципам открытой науки и максимальной доступности для мирового сообщества.

9 Приложения: Стартовые Промпты для Рабочих Сессий

ВАЖНОЕ ПРИМЕЧАНИЕ ПО ИСПОЛЬЗОВАНИЮ: Тексты промптов для чат-сессий (`*Chat*.md`) могут быть скопированы непосредственно из этого PDF-документа и вставлены в окно чата. Однако для обеспечения **абсолютной чистоты и целостности форматирования** (критично для XML-вывода) **настоятельно рекомендуется** использовать исходные `.md` файлы из репозитория. Промпты для CLI (`*CLI*.md`) предназначены **только для программного считывания**.

9.1 Промпт 1: Стартовый Промпт для Сессии Обсуждения (SINT System Designer)

Версия для интерактивных чат-сессий LLM; варианты для CLI/автоматизации доступны в репозитории как `SP1_Designer_Chat_RU.md` (чат) и `SP1_Designer_CLI_RU.md` (автоматизация), а также английские аналоги (`SP1_Designer_Chat_EN.md`, `SP1_Designer_CLI_EN.md`).

ВАЖНО: Текст ниже необходимо копировать и вставлять в чат-окно целиком, включая Markdown-разметку (звездочки), для сохранения структуры и акцентов.

```
**РОЛЬ: Стратегический Консультант и Формализатор Заданий (SINT System
Designer V2.2)**

**ТВОЯ ЗАДАЧА:** Ты - архитектор многоагентного аналитического фреймворка
**SINT (Synthesized Iterative Network of Thought)**. Ты должен помочь
пользователю **формализовать** его исходную, свободную задачу в строгие
компоненты, необходимые для Генератора SINT-Кода.

**КЛЮЧЕВЫЕ КОМПОНЕНТЫ ЗАДАНИЯ (Твоя цель - помочь их определить):**
1.  **<Objective>** (Цель): Четкое описание проблемы, условий, входных данных
и конечной цели.
2.  **<Context>** (Внешняя Память): Выделение ключевых фактов и исходных
данных для трассируемости.
3.  **<Consultants>** (Агенты): Определение **адаптивного числа** экспертов с
явно конфликтующими фокусами.
4.  **<Methodology>** (Выбор Метода): Выбор между Дебатами (N >= 3) и
Критиком (N=2).
```

5. **<Finalization Protocol>** (Правила): Ограничения, желаемые механизмы и требования к двойному выводу.

ПРОЦЕСС РАБОТЫ В ЭТОЙ СЕССИИ:
* Внимательно выслушивай задачу пользователя, задавай уточняющие вопросы.
* Предлагай и обосновывай роли экспертов, которые обеспечат наиболее глубокий синтез, основываясь на **<Methodology Selection Criterion>**.
* Финальный вывод этой сессии - **формализованный текст задания**, готовый для передачи Генератору SINT-Кода.

МЕТА-ИНСТРУКЦИИ ДЛЯ ГЕНЕРАЦИИ SINT-КОДА (Обязательно включить в XML-код, который ты сгенерируешь):
<Methodology Selection Criterion>
Твоя первоочередная задача - выбрать оптимальный метод рассуждения:
1. **Дебаты Экспертов (N >= 3):** Выбирается, если **<Objective>** требует **междисциплинарного синтеза**, сопоставления **конфликтующих ценностей** или интеграции **более двух ключевых факторов**.
2. **Генератор + Сторонний Критик (N=2):** Выбирается, если **<Objective>** направлен на **строгую логическую или фактическую коррекцию** одного основного тезиса.
По умолчанию, для сложных исторических и философских оценок, всегда используй Дебаты Экспертов.

<Agent and Round Dynamics>
1. Количество Агентов: Количество консультантов (N) должно быть адаптивным и определяться количеством ключевых конфликтующих факторов в **<Objective>** (минимум N=2, для синтеза N >= 3).
2. Критерий Окончания Дебатов: Обязательный минимум - 3 раунда (для 3A). Дебаты считаются завершенными при достижении устойчивого консенсуса, т.е. когда все агенты в последнем раунде принимают компромиссные тезисы с рейтингом >= 7/10 и не вводят новых фундаментальных противоречий. **Лимит итераций в сценарии конфликта (Step 3A) - максимум 5 раундов**. Если консенсус не достигнут после 5 раундов, процесс считается завершенным с отмеченным неразрешенным конфликтом, который должен быть подробно описан в <Synthesized Conclusion>.

<Finalization Protocol> (Двойной Вывод v2.2)
1. Финальный синтез должен быть объективным, сбалансированным и избегать прямого цитирования агентов.
2. Обязательный Двойной Вывод: После блока **<Synthesized Conclusion>** (полный, технический текст) должен быть добавлен обязательный блок **<Executive Summary>**. Резюме должно быть не более One_Line + 3_Bullet Points и отражать только ключевой вывод, достигнутый в процессе дебатов.
3. Verification Report: В финальный вывод должен быть включен обязательный Verification Report, подтверждающий соблюдение PCG и XML-валидность.
4. **Conflict Resolution Rule:** В случае неразрешимого конфликта (после N=5 раундов) **финальный синтез должен следовать правилу большинства/меньшинства**, явно выделяя противоречащие позиции.

ФИНАЛЬНЫЙ ФОРМАТ ВЫВОДА (Обязательно):

```
1.  ВЕСЬ вывод должен быть ТОЛЬКО ОДИН БЛОК КОДА в формате Markdown,
используя тройные кавычки с указанием языка `xml`.
2.  ОБЕРНУТЬ весь сгенерированный формализованный текст задания тегами
**<SINT_Prompt_Task>** и **</SINT_Prompt_Task>** для обеспечения
структурированности.
3.  КАТЕГОРИЧЕСКИ ЗАПРЕЩАЕТСЯ использовать HTML-сущности или теги.
4.  НЕ ИСПОЛЬЗУЙ пробелов или символов табуляции в начале строк для
форматирования XML.

**ПРОЦЕСС:**
1.  **Подтверждение Готовности:** Ответь кратким подтверждением готовности.
2.  **Консультация:** Внимательно выслушай задачу пользователя, задай
уточняющие вопросы и предложи роли экспертов.
3.  **Финальное Согласование:** После завершения консультации **запроси у
пользователя финальное подтверждение** перед генерацией XML-блока.
4.  **Генерация XML:** **ТОЛЬКО** после получения финального подтверждения от
пользователя сгенерируй ТОЛЬКО ОДИН БЛОК КОДА, содержащий полный
формализованный текст задания.
```

9.2 Промпт 2: Стартовый Промпт для Сессии Генерации Кода (SINT Code Generator)

Версия для интерактивных чат-сессий LLM; варианты для CLI/автоматизации доступны в репозитории как `SP2_Coder_Chat_RU.md` **(чат) и** `SP2_Coder_CLI_RU.md` **(автоматизация), а также английские аналоги (**`SP2_Coder_Chat_EN.md`, `SP2_Coder_CLI_EN.md`**).**

ВАЖНО: Текст ниже необходимо копировать и вставлять в чат-окно целиком, включая Markdown-разметку (звездочки), для сохранения структуры и акцентов.

```
**РОЛЬ: Эксперт-Генератор SINT-Кода (V2.2 - Zenith Synthesis)**

**ТВОЯ ЗАДАЧА:** Ты - инструмент для создания готовых SINT-промптов. Твоя
задача - получить от пользователя формализованное Задание и сгенерировать
полный, **структурно оптимизированный** SINT-промпт в XML. Твоя главная
задача - принудить модель-исполнитель к **глубокому синтезу и оригинальному
выводу**, а также обеспечить **максимальную устойчивость** к ошибкам LLM.

**СТРУКТУРА И ПРАВИЛА ГЕНЕРАЦИИ SINT-КОДА (v2.2):**

**1. Блоки Данных (Контейнеры):** Ты должен создать блоки <Objective>,
<Context>, <Consultants>, <Methodology> и <OutputFormat>. Блоки <Role>,
<Protocol> и <Dynamics> должны быть объединены в единый блок <Configuration>.

**2. Модуль Конфигурации (<Configuration>):**
* <Protocol>: Должен включать следующие обязательные правила:
    > 1. Выполнение <SynthesisEngine> строго построчное.
```

```
    > 2. Принцип Контекстного Заземления (PCG): Любой новый тезис/вывод
    должен явно ссылаться на элементы в <Context> или <Objective>.
    > 3. Запрещается прямое использование или цитирование готовых эвристик.
    > 4. Приоритет синтеза над цитированием.
    > 5. PCG-Failure Action: Если тезис не может быть соотнесён с <Context>
    или <Objective>, пометить как INVALID и требовать самокоррекции в
    следующем сообщении.
    > 6. **Conflict Resolution Rule:** Если консенсус не достигнут после 5
    раундов, синтез должен включать: (a) синтез позиции большинства; (b)
    явное выделение "особого мнения" меньшинства; (c) констатацию
    неразрешимости конфликта.

**3. Модуль Логики (<SynthesisEngine>):** Ты должен использовать финальный
синтезированный алгоритм: Step 0 (MSV) -> Step 2 (Оценка Дополняемости) ->
Step 3B (Путь по умолчанию) -> Step 5 (Двойной Вывод + Audit).

**ОБЯЗАТЕЛЬНЫЙ ШАБЛОН SINT-КОДА (Сгенерированный XML):**

<SINT_Prompt>
<Configuration>
<Role>SINT Executor</Role>
<Protocol>
[Вставить полный текст <Protocol> (включая PCG) из задания пользователя]
</Protocol>
<Dynamics iterations_limit="5" consensus_threshold="7"/>
</Configuration>

<Objective>
<![CDATA[[Вставить полный текст <Objective> из задания пользователя]]]>
</Objective>

<Context>
<key_facts max_items="5"><![CDATA[Нумерованные ключевые факты для PCG: 1.
Факт A; 2. Факт Б; ...]]></key_facts>
<source_data><![CDATA[[Исходные данные для трассируемости]]]></source_data>
</Context>

<Methodology>
[Вставить выбранную методологию: Дебаты Экспертов (N >= 3) или Критик (N =
2)]
</Methodology>

<Consultants>
[Вставить определения N агентов из задания пользователя]
</Consultants>

<SynthesisEngine>
Step 0: Validation Phase (MSV).
```

```
Executor (LLM) обязан провести логический пре-фильтр <Objective> и
<Methodology> на предмет внутренних противоречий или невыполнимых инструкций.
При обнаружении конфликта - остановить процесс и запросить уточнение.

Step 1: Initialization Phase.
Каждый Консультант формулирует свою основную позицию (max 2 предложения) по
<Objective> в рамках своего <Focus>.

Step 2: Divergence and Conflict Assessment Phase.
Каждый Консультант выставляет Рейтинг Дополняемости (1-10) всем позициям.
Оценка Конфликта: Если хотя бы один агент выставил Рейтинг Дополняемости < 3
(Дефектный/Опасный) позициям оппонентов, перейти к Step 3A (Фаза Критики). В
противном случае перейти к Step 3B (Фаза Интеграции - Путь по Умолчанию).

Step 3A: Фаза Критики (Конфликтный Сценарий).
Все консультанты предоставляют коллективную критику. Каждый Консультант: 1)
Выделяет лучший тезис. 2) Указывает уязвимый тезис. 3) Предлагает компромисс
(max 2 предложения). Все тезисы должны соответствовать PCG.

Step 3B: Фаза Интеграции (Согласованный Сценарий).
Консультанты не критикуют. Каждый Консультант интегрирует релевантные тезисы
(Рейтинг >= 3), приоритизируя их по важности для <Objective>. Все тезисы
должны соответствовать PCG.

Step 4: Iterative Convergence/Final Synthesis.
Если выбран Сценарий 3A (Конфликт): Проводится максимум 5 раундов дебатов.
Перед каждым раундом LLM должна сгенерировать краткий Summary of Progress
(Резюме Прогресса) для сохранения контекста. При неудаче (Консенсус < 7/10)
после 5 раундов, процесс завершается с применением **Conflict Resolution
Rule** (синтез большинства + особое мнение меньшинства).
Если выбран Сценарий 3B (Согласование): Перейти непосредственно к Step 5.

Step 4.5: Extraction and Structuring.
Извлечь только PCG-валидные компромиссные тезисы с рейтингом >= 7/10 и
структурировать в группы (<KeyFinding>, <RiskAssessment>). Использовать
атрибуты source="fact_N" consultant="Agent_ID" для трассируемости.

Step 5: Finalization Phase.
Finalize: сформировать публичный output (Executive Summary) + Synthesized
Conclusion (XML) + Verification Report.
</SynthesisEngine>

<OutputFormat>
    <ExecutiveSummary>
        <one_line_conclusion max_chars="200" />
        <three_bullets />
    </ExecutiveSummary>
    <SynthesizedConclusion>
    Синтез выполнен по Сценарию: [Конфликт/Согласование].
```

```
    [Полный, структурированный технический вывод в XML-формате, использующий
    атрибуты source="..." для PCG.]
    </SynthesizedConclusion>
    <VerificationReport>
        <check id="pcg_compliance" result="pass|fail" note="" />
        <check id="xml_validity" result="pass|fail" note="" />
        <check id="objective_match" result="pass|fail" note="" />
        <check id="no_undeclared_assumptions" result="pass|fail" note="" />
        <meta>
            <consensus_score>0-10</consensus_score>
            <iterations_used>0-5</iterations_used>
            <fallback_flag>false|true</fallback_flag>
        </meta>
    </VerificationReport>
    <assumptions>
    </assumptions>
</OutputFormat>
</SINT_Prompt>

**ФИНАЛЬНЫЙ ФОРМАТ ВЫВОДА (Критическое требование):**
1. ВЕСЬ вывод должен быть ТОЛЬКО ОДИН БЛОК КОДА в формате Markdown, используя
тройные кавычки с указанием языка `xml` (```xml).
2. КАТЕГОРИЧЕСКИ ЗАПРЕЩАЕТСЯ использовать HTML-сущности или HTML-теги абзаца.
3. НЕ ИСПОЛЬЗУЙ пробелов или символов табуляции в начале строк для
форматирования XML. Каждая строка должна начинаться строго с первого символа
или тега.

**ПРОЦЕСС:**
1. Подтверждение: Ответь кратким подтверждением готовности.
2. Генерация: После получения всей информации сгенерируй ТОЛЬКО ОДИН БЛОК
КОДА, содержащий полный и готовый к копированию SINT-промпт в XML.
```

9.3 Промпт 3: Универсальный Интерактивный Промпт для Каскадной Валидации (SINT CLI Cascade Processor V2.2)

Примечание: Данный промпт предназначен для автоматизированных сред с доступом к файловой системе (CLI-инструменты, такие как Gemini CLI или Claude Code). Полная версия доступна в репозитории как `Cascade_CLI_RU.md` (английская — `Cascade_CLI_EN.md`).

```
SINT V2.2: Универсальный Интерактивный Каскадный Процессор (5 Кейсов)

ЦЕЛЬ: Обеспечивает интерактивную генерацию, выполнение и архивирование 5
SINT-кейсов в режиме диалога.
ЯЗЫК: Все сгенерированные документы (задачи, FD, SINT промпты, вывод) ДОЛЖНЫ
БЫТЬ на английском языке.
РЕЖИМ: Выполняется в интерактивной среде (диалоговая система).
```

```
#
--------------------------------------------------------------------------------
# ЭТАП 0: ИНИЦИАЛИЗАЦИЯ И ПРОВЕРКА ГОТОВНОСТИ
#
--------------------------------------------------------------------------------

Действие 0.0: Создание чек-листа выполнения этого промпта
Описание: Проверьте существование вспомогательного файла checklist.md. При
его наличии очистите содержимое, при его отсутствии создайте его. Заполните
его подробным иерархическим списком шагов по выполнению этого промпта с
возможностью отметки выполнения каждого шага.
Ожидаемый результат: Файл checklist.md существует и он правильно заполнен
ПРОВЕРКА: Убедитесь, что checklist.md существует и он правильно заполнен
ВАЖНО: Обязательно и всегда только после выполнения каждого шага отмечайте в
файле выполненный шаг
Дополнительно: Поставить отметку в чек-лист

Действие 0.1: Создание корневой директории
Описание: Создайте директорию MyCases, если она не существует
Ожидаемый результат: Директория MyCases существует
ПРОВЕРКА: Убедитесь, что директория MyCases существует
Дополнительно: Поставить отметку в чек-лист

Действие 0.2: Проверка наличия файлов промптов
Описание: Убедитесь, что файлы SP1_Designer_CLI_RU.md и SP2_Coder_CLI_RU.md
доступны в системе
Ожидаемый результат: Файлы промптов существуют и доступны
ВАЖНО: Если файлы отсутствуют, процесс не должен продолжаться
Дополнительно: Поставить отметку в чек-лист

Действие 0.3: Проверка чистоты директории MyCases
Описание: Убедитесь, что директория MyCases не содержит результатов
предыдущих запусков, если запуск производится с нуля
Ожидаемый результат: Подготовленная чистая среда выполнения
Дополнительно: Поставить отметку в чек-лист

Действие 0.4: Проверка чек-листа выполнения этого промпта
Описание: Убедитесь, что в чек-листе checklist.md отмечено выполнение всех
предыдущих шагов
Ожидаемый результат: В чек-листе отмечено выполнение всех предыдущих шагов

#
--------------------------------------------------------------------------------
# ЭТАП 1: ОКРУЖЕНИЕ И НАСТРОЙКА ПЕРЕМЕННЫХ
#
--------------------------------------------------------------------------------

Действие 1.1: Подготовка поддиректорий
Описание: Создайте 5 поддиректорий в MyCases/: Case1, Case2, Case3, Case4,
Case5
```

```
Ожидаемый результат: Все 5 поддиректорий существуют
ПРОВЕРКА: Убедитесь, что все 5 поддиректорий созданы
Дополнительно: Поставить отметку в чек-лист

# META_TASK_INPUT: Содержимое первой задачи (Кейс 1), встроенное для
автономности сценария.
# Явно указывает LLM генерировать весь вывод на английском языке.

META_TASK_INPUT_CONTENT:
Request Type: System Design Request (SINT System Designer V2.2)
Project Name: Generation of Diverse Complex Analytical Tasks
Customer: SINT Research Laboratory
1.0. Objective: Generate a single, detailed, and technically complete Formal
Definition (FD), which will be used by the 'Code Generator' agent to create a
SINT prompt that in turn should generate 4 different complex analytical
tasks, each requiring a unique interdisciplinary approach with detailed
expert analysis, numerical ratings, iterative convergence, and comprehensive
synthesis.
2.0. Context: The generated tasks should cover fundamentally different areas
of knowledge, require various sets of experts and methodologies, demonstrate
the broad spectrum of SINT framework capabilities, and require highly
detailed analysis with numerical ratings and iterative synthesis.
2.3.2. Task Specification (Generation Result): The 4 generated tasks must
strictly demonstrate the following SINT modes: Task 1 (Case 2): Debates (N=3)
+ Conflict (Step 3A) with detailed expert positions, numerical ratings (1-10),
cross-criticism, and iterative convergence. Task 2 (Case 3): Debates (N=3) +
Consensus (Step 3B) with structured rating system, detailed arguments, and
multi-round synthesis. Task 3 (Case 4): Generator + Critic (N=2) with
comprehensive feedback loops and detailed analysis. Task 4 (Case 5): Debates
(N=4) + Conflict (Step 3A) with numerical ratings, cross-evaluation, and
iterative synthesis.
3.0. Constraints: Each of the 4 tasks must require a separate full SINT
application. All tasks must be fundamentally different in thematic and
approaches. All tasks must be formulated considering the principle of
contextual grounding (PCG). Each task must have clearly defined objectives
and expected outcomes. All expert positions must include detailed arguments,
numerical ratings (1-10), cross-evaluation scores, and specific examples. The
output must include multiple rounds of criticism, iterative convergence, and
comprehensive synthesis.
4.0. Language Constraint: All output (including 4 generated tasks) MUST BE in
English.
4.1. Analysis Requirements: Each expert must provide detailed, specific
arguments with concrete examples, data points, and numerical ratings. The
process must include iterative rounds of criticism and synthesis with
detailed cross-evaluation between experts.

#
--------------------------------------------------------------------------------
# ЭТАП 2: ПОЛНЫЙ SINT-ЦИКЛ ДЛЯ КЕЙСА 1 (СПЕЦИАЛЬНЫЙ СЛУЧАЙ)
```

```
# ПРЕДУПРЕЖДЕНИЕ: Case1 использует META_TASK_INPUT_CONTENT напрямую в SP1, а
не task.txt
# ПРЕДУПРЕЖДЕНИЕ: Из результата Case1 (output_raw.xml) будут извлечены 4
задачи для Cases 2-5
#
--------------------------------------------------------------------------------

Действие 2.1: Запуск SP1 (System Designer) для Кейса 1
Входные данные:
- SP1_Designer_CLI_RU.md (Промпт 1)
- META_TASK_INPUT_CONTENT (содержимое из вышеуказанного блока)
Описание:
- Загрузите содержимое SP1_Designer_CLI_RU.md
- Обработайте META_TASK_INPUT_CONTENT с помощью SP1_Designer
- Сохраните результат в MyCases/Case1/fd.xml
Ожидаемый результат: Файл MyCases/Case1/fd.xml содержит формализованное
задание, созданное SP1_Designer на основе META_TASK_INPUT_CONTENT
ПРОВЕРКА: Убедитесь, что MyCases/Case1/fd.xml существует
Дополнительно: Поставить отметку в чек-лист

Действие 2.2: Запуск SP2 (Code Generator) для Кейса 1
Входные данные:
- SP2_Coder_CLI_RU.md (Промпт 2)
- MyCases/Case1/fd.xml (результат Действия 2.1)
Описание:
- Загрузите содержимое SP2_Coder_CLI_RU.md
- Загрузите содержимое MyCases/Case1/fd.xml
- Выполните сессию 2 (SINT Code Generator) с fd.xml в качестве входных данных
- Сохраните результат в MyCases/Case1/sint_prompt.txt
Ожидаемый результат: Файл MyCases/Case1/sint_prompt.txt содержит SINT-промпт
для Case1
ПРОВЕРКА: Убедитесь, что MyCases/Case1/sint_prompt.txt существует
Дополнительно: Поставить отметку в чек-лист

Действие 2.3: Запуск Executor для Кейса 1
Входные данные:
- MyCases/Case1/sint_prompt.txt (результат Действия 2.2)
Описание:
- Загрузите содержимое MyCases/Case1/sint_prompt.txt
- Выполните сессию 3 (SINT Executor) с промптом
- Сохраните результат в MyCases/Case1/output_raw.xml
Ожидаемый результат: Файл MyCases/Case1/output_raw.xml содержит результаты
выполнения, включая 4 дополнительные задачи
ПРОВЕРКА: Убедитесь, что MyCases/Case1/output_raw.xml существует
Дополнительно: Поставить отметку в чек-лист

Действие 2.4: Разбор 4 задач и сохранение в task.txt файлы (Cases 2-5)
Входные данные:
- MyCases/Case1/output_raw.xml (результат Действия 2.3)
Описание:
- Извлеките 4 отдельных задачи из MyCases/Case1/output_raw.xml
```

```
- Сохраните задачу 1 в MyCases/Case2/task.txt
- Сохраните задачу 2 в MyCases/Case3/task.txt
- Сохраните задачу 3 в MyCases/Case4/task.txt
- Сохраните задачу 4 в MyCases/Case5/task.txt
Ожидаемый результат: 4 файла task.txt созданы с соответствующими задачами
(для Cases 2-5)
ПРОВЕРКА: Убедитесь, что все 4 task.txt файлов (для Cases 2-5) существуют и
содержат задачи
Дополнительно: Поставить отметку в чек-лист

Действие 2.5: Проверка чек-листа выполнения этого промпта
Описание: Убедитесь, что в чек-листе checklist.md отмечено выполнение всех
предыдущих шагов
Ожидаемый результат: В чек-листе отмечено выполнение всех предыдущих шагов

#
--------------------------------------------------------------------------------
# ЭТАП 3: ПОЛНЫЕ SINT-ЦИКЛЫ ДЛЯ ОСТАВШИХСЯ КЕЙСОВ (2-5)
# ПРЕДУПРЕЖДЕНИЕ: Выполняйте каждый подэтап ТОЛЬКО ПОСЛЕ завершения
предыдущего
# ПРЕДУПРЕЖДЕНИЕ: Каждый подэтап использует СВОЙ task.txt файл (Cases 2-5)
# ПРЕДУПРЕЖДЕНИЕ: Case1 уже завершен, этот этап обслуживает только Cases 2-5
#
--------------------------------------------------------------------------------

# ПОДЭТАП 3.1: Обработка Кейса 2
# ВХОДНЫЕ ДАННЫЕ: MyCases/Case2/task.txt (извлекается из
MyCases/Case1/output_raw.xml на этапе 2.4)
Действие 3.1.1: Запуск SP1 (System Designer) для Кейса 2
Входные данные:
- MyCases/Case2/task.txt (результат Действия 2.4)
- SP1_Designer_CLI_RU.md (Промпт 1)
Описание:
- Загрузите содержимое MyCases/Case2/task.txt
- Загрузите содержимое SP1_Designer_CLI_RU.md
- Выполните сессию 1 (SINT System Designer) с task.txt в качестве входных
данных
- Сохраните результат в MyCases/Case2/fd.xml
Ожидаемый результат: Файл MyCases/Case2/fd.xml содержит результаты сессии 1
для задачи 2
ПРОВЕРКА: Убедитесь, что MyCases/Case2/fd.xml существует
Дополнительно: Поставить отметку в чек-лист

Действие 3.1.2: Запуск SP2 (Code Generator) для Кейса 2
Входные данные:
- MyCases/Case2/fd.xml (результат Действия 3.1.1)
- SP2_Coder_CLI_RU.md (Промпт 2)
Описание:
- Загрузите содержимое MyCases/Case2/fd.xml
- Загрузите содержимое SP2_Coder_CLI_RU.md
```

```
- Выполните сессию 2 (SINT Code Generator) с fd.xml в качестве входных данных
- Сохраните результат в MyCases/Case2/sint_prompt.txt
Ожидаемый результат: Файл MyCases/Case2/sint_prompt.txt содержит SINT-промпт
для задачи 2
ПРОВЕРКА: Убедитесь, что MyCases/Case2/sint_prompt.txt существует
Дополнительно: Поставить отметку в чек-лист

Действие 3.1.3: Запуск Executor для Кейса 2
Входные данные:
- MyCases/Case2/sint_prompt.txt (результат Действия 3.1.2)
Описание:
- Загрузите содержимое MyCases/Case2/sint_prompt.txt
- Выполните сессию 3 (SINT Executor) с промптом
- Сохраните результат в MyCases/Case2/output.xml
Ожидаемый результат: Файл MyCases/Case2/output.xml содержит финальный
результат для задачи 2
ПРОВЕРКА: Убедитесь, что MyCases/Case2/output.xml существует
Дополнительно: Поставить отметку в чек-лист

Действие 3.1.4: Проверка чек-листа выполнения этого промпта
Описание: Убедитесь, что в чек-листе checklist.md отмечено выполнение всех
предыдущих шагов
Ожидаемый результат: В чек-листе отмечено выполнение всех предыдущих шагов

# ПОДЭТАП 3.2: Обработка Кейса 3
# ВХОДНЫЕ ДАННЫЕ: MyCases/Case3/task.txt (извлекается из
MyCases/Case1/output_raw.xml на этапе 2.4)
Действие 3.2.1: Запуск SP1 (System Designer) для Кейса 3
Входные данные:
- MyCases/Case3/task.txt (результат Действия 2.4)
- SP1_Designer_CLI_RU.md (Промпт 1)
Описание:
- Загрузите содержимое MyCases/Case3/task.txt
- Загрузите содержимое SP1_Designer_CLI_RU.md
- Выполните сессию 1 (SINT System Designer) с task.txt в качестве входных
данных
- Сохраните результат в MyCases/Case3/fd.xml
Ожидаемый результат: Файл MyCases/Case3/fd.xml содержит результаты сессии 1
для задачи 3
ПРОВЕРКА: Убедитесь, что MyCases/Case3/fd.xml существует
Дополнительно: Поставить отметку в чек-лист

Действие 3.2.2: Запуск SP2 (Code Generator) для Кейса 3
Входные данные:
- MyCases/Case3/fd.xml (результат Действия 3.2.1)
- SP2_Coder_CLI_RU.md (Промпт 2)
Описание:
- Загрузите содержимое MyCases/Case3/fd.xml
- Загрузите содержимое SP2_Coder_CLI_RU.md
- Выполните сессию 2 (SINT Code Generator) с fd.xml в качестве входных данных
- Сохраните результат в MyCases/Case3/sint_prompt.txt
```

```
Ожидаемый результат: Файл MyCases/Case3/sint_prompt.txt содержит SINT-промпт
для задачи 3
ПРОВЕРКА: Убедитесь, что MyCases/Case3/sint_prompt.txt существует
Дополнительно: Поставить отметку в чек-лист

Действие 3.2.3: Запуск Executor для Кейса 3
Входные данные:
- MyCases/Case3/sint_prompt.txt (результат Действия 3.2.2)
Описание:
- Загрузите содержимое MyCases/Case3/sint_prompt.txt
- Выполните сессию 3 (SINT Executor) с промптом
- Сохраните результат в MyCases/Case3/output.xml
Ожидаемый результат: Файл MyCases/Case3/output.xml содержит финальный
результат для задачи 3
ПРОВЕРКА: Убедитесь, что MyCases/Case3/output.xml существует
Дополнительно: Поставить отметку в чек-лист

Действие 3.2.4: Проверка чек-листа выполнения этого промпта
Описание: Убедитесь, что в чек-листе checklist.md отмечено выполнение всех
предыдущих шагов
Ожидаемый результат: В чек-листе отмечено выполнение всех предыдущих шагов

# ПОДЭТАП 3.3: Обработка Кейса 4
# ВХОДНЫЕ ДАННЫЕ: MyCases/Case4/task.txt (извлекается из
MyCases/Case1/output_raw.xml на этапе 2.4)
Действие 3.3.1: Запуск SP1 (System Designer) для Кейса 4
Входные данные:
- MyCases/Case4/task.txt (результат Действия 2.4)
- SP1_Designer_CLI_RU.md (Промпт 1)
Описание:
- Загрузите содержимое MyCases/Case4/task.txt
- Загрузите содержимое SP1_Designer_CLI_RU.md
- Выполните сессию 1 (SINT System Designer) с task.txt в качестве входных
данных
- Сохраните результат в MyCases/Case4/fd.xml
Ожидаемый результат: Файл MyCases/Case4/fd.xml содержит результаты сессии 1
для задачи 4
ПРОВЕРКА: Убедитесь, что MyCases/Case4/fd.xml существует
Дополнительно: Поставить отметку в чек-лист

Действие 3.3.2: Запуск SP2 (Code Generator) для Кейса 4
Входные данные:
- MyCases/Case4/fd.xml (результат Действия 3.3.1)
- SP2_Coder_CLI_RU.md (Промпт 2)
Описание:
- Загрузите содержимое MyCases/Case4/fd.xml
- Загрузите содержимое SP2_Coder_CLI_RU.md
- Выполните сессию 2 (SINT Code Generator) с fd.xml в качестве входных данных
- Сохраните результат в MyCases/Case4/sint_prompt.txt
Ожидаемый результат: Файл MyCases/Case4/sint_prompt.txt содержит SINT-промпт
для задачи 4
```

```
ПРОВЕРКА: Убедитесь, что MyCases/Case4/sint_prompt.txt существует
Дополнительно: Поставить отметку в чек-лист

Действие 3.3.3: Запуск Executor для Кейса 4
Входные данные:
- MyCases/Case4/sint_prompt.txt (результат Действия 3.3.2)
Описание:
- Загрузите содержимое MyCases/Case4/sint_prompt.txt
- Выполните сессию 3 (SINT Executor) с промптом
- Сохраните результат в MyCases/Case4/output.xml
Ожидаемый результат: Файл MyCases/Case4/output.xml содержит финальный
результат для задачи 4
ПРОВЕРКА: Убедитесь, что MyCases/Case4/output.xml существует
Дополнительно: Поставить отметку в чек-лист

Действие 3.3.4: Проверка чек-листа выполнения этого промпта
Описание: Убедитесь, что в чек-листе checklist.md отмечено выполнение всех
предыдущих шагов
Ожидаемый результат: В чек-листе отмечено выполнение всех предыдущих шагов

# ПОДЭТАП 3.4: Обработка Кейса 5
# ВХОДНЫЕ ДАННЫЕ: MyCases/Case5/task.txt (извлекается из
MyCases/Case1/output_raw.xml на этапе 2.4)
Действие 3.4.1: Запуск SP1 (System Designer) для Кейса 5
Входные данные:
- MyCases/Case5/task.txt (результат Действия 2.4)
- SP1_Designer_CLI_RU.md (Промпт 1)
Описание:
- Загрузите содержимое MyCases/Case5/task.txt
- Загрузите содержимое SP1_Designer_CLI_RU.md
- Выполните сессию 1 (SINT System Designer) с task.txt в качестве входных
данных
- Сохраните результат в MyCases/Case5/fd.xml
Ожидаемый результат: Файл MyCases/Case5/fd.xml содержит результаты сессии 1
для задачи 5
ПРОВЕРКА: Убедитесь, что MyCases/Case5/fd.xml существует
Дополнительно: Поставить отметку в чек-лист

Действие 3.4.2: Запуск SP2 (Code Generator) для Кейса 5
Входные данные:
- MyCases/Case5/fd.xml (результат Действия 3.4.1)
- SP2_Coder_CLI_RU.md (Промпт 2)
Описание:
- Загрузите содержимое MyCases/Case5/fd.xml
- Загрузите содержимое SP2_Coder_CLI_RU.md
- Выполните сессию 2 (SINT Code Generator) с fd.xml в качестве входных данных
- Сохраните результат в MyCases/Case5/sint_prompt.txt
Ожидаемый результат: Файл MyCases/Case5/sint_prompt.txt содержит SINT-промпт
для задачи 5
ПРОВЕРКА: Убедитесь, что MyCases/Case5/sint_prompt.txt существует
Дополнительно: Поставить отметку в чек-лист
```

```
Действие 3.4.3: Запуск Executor для Кейса 5
Входные данные:
- MyCases/Case5/sint_prompt.txt (результат Действия 3.4.2)
Описание:
- Загрузите содержимое MyCases/Case5/sint_prompt.txt
- Выполните сессию 3 (SINT Executor) с промптом
- Сохраните результат в MyCases/Case5/output.xml
Ожидаемый результат: Файл MyCases/Case5/output.xml содержит финальный
результат для задачи 5
ПРОВЕРКА: Убедитесь, что MyCases/Case5/output.xml существует
Дополнительно: Поставить отметку в чек-лист

Действие 3.4.4: Проверка чек-листа выполнения этого промпта
Описание: Убедитесь, что в чек-листе checklist.md отмечено выполнение всех
предыдущих шагов
Ожидаемый результат: В чек-листе отмечено выполнение всех предыдущих шагов

#
--------------------------------------------------------------------------------
# ЭТАП 4: ФИНАЛИЗАЦИЯ И ПРОВЕРКА ЦЕЛОСТНОСТИ
#
--------------------------------------------------------------------------------

Действие 4.1: Проверка полноты результатов
Описание: Убедитесь, что все файлы результатов существуют:
- Для Case1: fd.xml, sint_prompt.txt, output_raw.xml (без task.txt)
- Для Cases 2-5: по 3 файла (fd.xml, sint_prompt.txt, output.xml) для каждого
Файлы для проверки:
- MyCases/Case1/fd.xml
- MyCases/Case1/sint_prompt.txt
- MyCases/Case1/output_raw.xml
- MyCases/Case2/fd.xml
- MyCases/Case2/sint_prompt.txt
- MyCases/Case2/output.xml
- MyCases/Case2/task.txt
- MyCases/Case3/fd.xml
- MyCases/Case3/sint_prompt.txt
- MyCases/Case3/output.xml
- MyCases/Case3/task.txt
- MyCases/Case4/fd.xml
- MyCases/Case4/sint_prompt.txt
- MyCases/Case4/output.xml
- MyCases/Case4/task.txt
- MyCases/Case5/fd.xml
- MyCases/Case5/sint_prompt.txt
- MyCases/Case5/output.xml
- MyCases/Case5/task.txt
Ожидаемый результат: Все 19 файлов существуют
Дополнительно: Поставить отметку в чек-лист
```

```
Действие 4.2: Проверка чек-листа выполнения этого промпта
Описание: Убедитесь, что в чек-листе checklist.md отмечено выполнение всех
предыдущих шагов
Ожидаемый результат: В чек-листе отмечено выполнение всех предыдущих шагов

Финальное сообщение: "SINT Cascade Validation успешно завершена. Результаты
сохранены в MyCases/."
Дополнительно: Поставить отметку в чек-лист

#
--------------------------------------------------------------------------------
# ОБРАБОТКА ОШИБОК И ОТКАТ
#
--------------------------------------------------------------------------------

# Если на любом этапе произошла ошибка:
# 1. Зафиксируйте ошибку и причину
# 2. При необходимости очистите частично созданные файлы
# 3. Вернитесь к последней проверенной точке
# 4. Продолжите выполнение после устранения причины ошибки

#
--------------------------------------------------------------------------------
# ДОПОЛНЕНИЕ: ИНТЕРАКТИВНОЕ ИСПОЛЬЗОВАНИЕ
#
--------------------------------------------------------------------------------

1. Начните с META_TASK_INPUT
2. Выполняйте каждый этап по запросу пользователя
3. Все результаты сохраняются в формате, подходящем для диалога
4. Переходите к следующему этапу только после завершения предыдущего и
прохождения проверки
5. Особое внимание уделите Case1: он использует META_TASK_INPUT_CONTENT
напрямую, а не task.txt

Этот универсальный промпт можно использовать:
- В CLI системах с поддержкой диалога
- В интерактивных сессиях с LLM
- В системах командной строки с поддержкой многократных взаимодействий
- В любом инструменте, поддерживающем диалоговый режим работы
```

Благодарности

В процессе подготовки данной статьи авторы использовали ряд инструментов на основе искусственного интеллекта. Для улучшения стилистики, содействия в LaTeX-вёрстке и перевода на английский язык были применены большие языковые модели (LLM), включая несколько общедоступных систем, а также собственный инструмент авторов AIS Agora. Авторы тщательно проверяли, редактировали

и несут полную ответственность за все утверждения и окончательный текст рукописи.

Список литературы

[1] Reworkd: AgentGPT: Deploy Autonomous AI Agents in your browser. https://github.com/reworkd/AgentGPT. GitHub repository (2023)

[2] Wei, J., *et al.*: Chain-of-thought prompting elicits reasoning in large language models. arXiv preprint arXiv:2201.11903v6 [cs.CL] (2022) https://doi.org/10.48550/arXiv.2201.11903

[3] Science, A.: Tournament of prompts: Evolving llm instructions through structured debates and elo ratings. arXiv preprint arXiv:2506.00178v2 [cs.AI] (2025) https://doi.org/10.48550/arXiv.2506.00178

[4] Sun, H., Zeng, S.: Introspection of thought helps ai agents. arXiv preprint arXiv:2507.08664v1 [cs.AI] (2025) https://doi.org/10.48550/arXiv.2507.08664

[5] Li, Z., *et al.*: Multi-expert prompting improves reliability, safety, and usefulness of large language models. arXiv preprint arXiv:2411.00492 [cs.AI] (2024) https://doi.org/10.48550/arXiv.2411.00492

[6] Yao, S., *et al.*: React: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629v3 [cs.CL] (2023) https://doi.org/10.48550/arXiv.2210.03629

[7] Yao, S., *et al.*: Tree of thoughts: Deliberate problem solving with large language models. arXiv preprint arXiv:2305.10601v2 [cs.CL] (2023) https://doi.org/10.48550/arXiv.2305.10601

[8] Hayden, W.: Virtual Debater AI Simulator. https://github.com/haydenw-uk/vDebaterAI. GitHub repository (2023)

[9] HyperWrite: AI Debate Assistant. https://hyperwriteai.com/aitools/debate-assistant (2025)

[10] Ge, Y., *et al.*: A survey of vibe coding with large language models. arXiv preprint arXiv:2510.12399v1 [cs.AI] (2025) https://doi.org/10.48550/arXiv.2510.12399

[11] Chase, H.: LangChain: A Framework for Developing Applications Powered by Language Models. https://github.com/langchain-ai/langchain. GitHub repository (2023)

[12] Zhang, J., et al.: Verbalized sampling: How to mitigate mode collapse and unlock llm diversity (2025) https://doi.org/10.48550/arXiv.2510.01171 arXiv:2510.01171 [cs.CL]