SENAR

Supervised Engineering & Normative AI Regulation

Версия 1.3 | 25.03.2026

Авторы: Андрей Юмашев, Вадим Соглаев

Основы SENAR

8 правил для AI-разработки

Версия: 1.3 | **Дата:** 2026-03-25 **Авторы:** Андрей Юмашев (Andrey Yumashev), Вадим Соглаев (Vadim Soglaev) **Сайт:** senar.tech

---

> *8 правил, 2 шлюза качества, 2 метрики и чеклист верификации — всё, что нужно для работы с AI-ассистентами без багов в продакшене. Никакого специального инструментария не требуется. Текстовый редактор и система контроля версий — достаточно для старта. По мере зрелости практики автоматизация помогает поддерживать дисциплину при масштабировании. Время чтения: 15 минут.*

Для кого это

Вы пишете код с AI-ассистентами — Copilot, Claude, Cursor или аналогами. Вы заметили: результат AI выглядит правильно, но иногда ломается в продакшене. Вам нужна лёгкая дисциплина, которая ловит проблемы до развёртывания.

Основы SENAR — это такая дисциплина. Работает с любым AI-инструментом, любым языком, любым процессом.

Для организаций, масштабирующих AI-разработку на несколько команд, есть полный SENAR Standard.

Ключевой термин

Супервайзер — человек, который направляет AI. Если работаете один — это вы. В команде — тот, кто ставит задачу и проверяет результат AI. Один человек на задачу.

---

8 правил

Правило 1. Задача до кода

Каждое изменение начинается с задачи:

- **Цель** — что нужно сделать (одно предложение).
- **Критерии приёмки** — нумерованные условия, определяющие «готово». Каждое проверяется независимо.
- **Негативный сценарий** — хотя бы одна ситуация, которая может пойти не так (ошибка, некорректный ввод, граничное условие).

Для задач, затрагивающих авторизацию, платежи, персональные данные или внешние API — включите хотя бы один критерий приёмки про валидацию входных данных и контроль доступа.

Не начинайте кодить без задачи. Исследования — свободно, но коммит результатов исследования требует задачи.

Зачем это нужно. Качество результата AI равно качеству входных данных. «Добавь логин» порождает правдоподобный код, который проваливается на граничных случаях. «POST /auth/login возвращает JWT; возвращает 401 при неверном пароле; возвращает 422 при отсутствующем email» даёт тестируемый код с первой попытки.

Правило 2. Границы области изменений

Каждая задача явно определяет:

- **Что менять** — файлы, модули или компоненты в рамках задачи.
- **Чего не трогать** — минимум: «не изменять файлы за пределами указанной области». Будьте конкретнее, когда задача рядом с чувствительными зонами.

Без границ AI-агенты уверенно рефакторят код за пределами задачи, добавляют лишние абстракции или правят работающие системы, которых никто не просил трогать. Расползание границ — самая частая ошибка AI.

Также полезно: указать шаблоны для подражания («следуй структуре `auth/router.py`») и ограничения («не изменяй схему базы данных»).

Правило 3. Верификация по критериям

Каждый критерий приёмки проверяется фактически — запустить тест, проверить вывод, измерить результат, просмотреть diff.

«Работает» — это не верификация. «Выглядит правильно» — это не верификация. Для каждого критерия нужно зафиксированное свидетельство: пройденный тест, вывод команды, измеренное значение, подтверждённое поведение.

Нет свидетельства по критерию? Этот критерий не верифицирован. Где автоматические тесты уместны — пишите их. Где нет (документация, конфигурация, инфраструктура) — подойдут другие свидетельства: вывод команды, скриншот, результат grep.

Никогда не принимайте результат AI на основании интуиции, внешнего вида кода или заявления AI о готовности. **Уверенность AI — это не доказательство.**

Правило 4. Тесты проверяют требования, а не реализацию

Тесты берутся из критериев приёмки. Они проверяют **что** система делает, а не **как**.

Тест, который ломается при рефакторинге реализации (без изменения поведения) — это тест реализации. Тесты реализации создают ложные падения при рефакторинге и ложную уверенность при изменении функциональности.

Хороший тест: «POST /auth/login с валидными учётными данными возвращает 200 и JWT». Плохой тест: «Метод `_hash_password` вызывается ровно один раз с bcrypt».

Следите за тестами, которые генерирует AI: он склонен создавать тесты, отражающие реализацию, а не проверяющие требование. Тест, который просто повторяет то, что делает код, ничего не проверяет.

Правило 5. Проверка на скрытые дефекты

Перед тем как объявить задачу «готовой», пройдите чеклист верификации (см. ниже). Три уровня по степени риска:

- **Стандартный уровень** (каждая задача): область изменений, удаления, фантомные импорты, качество тестов, подмена тестов, валидация входных данных, захардкоженные секреты, устаревшие паттерны, межфайловая согласованность.
- **Высокий уровень** (безопасность, авторизация, платежи, данные): стандартный плюс обход null-проверок, обход пустой конфигурации, доверие заголовкам, IDOR, шорткат return-True, покрытие авторизации, небезопасная десериализация.
- **Критический уровень** (продакшен-инциденты, регуляторика, сложные фичи): все пункты.

Задачи с пользовательским вводом, записью в базу, аутентификацией, файловым вводом-выводом или вызовами внешних API — минимум высокий уровень. При

сомнениях берите выше.

Зачем это нужно. AI-дефекты сложно заметить — они проходят автоматические проверки, но содержат тонкие логические или архитектурные ошибки и проблемы безопасности. Чеклист бьёт по конкретным паттернам: возврат True без валидации, обход проверок через сравнение с null, доверие HTTP-заголовкам, пустые значения конфигурации, которые молча отключают защиту.

Правило 6. Нулевая терпимость к незавершённой работе

Каждое утверждение о завершённости подкрепляется свидетельством:

- «Все тесты проходят» — покажи вывод тестового прогона.
- «Нет нарушений линтера» — покажи вывод линтера.
- «API возвращает 401» — покажи ответ.
- «Файл обновлён» — покажи diff.

«Готово» без верификации — это не готово. «Вероятно работает» — это не готово. Если у задачи 5 критериев приёмки, все 5 нуждаются в зафиксированном свидетельстве. Заявить о полном завершении при частичной готовности — хуже, чем честно сказать «не закончил».

Правило 7. Устраняй причины, а не симптомы

Нашли дефект — установите корневую причину до того, как чинить.

Быстрый фикс, подавляющий симптом, создаёт техдолг, маскирует смежные дефекты и гарантирует повторение. AI-агенты особенно склонны к таким фиксам — они добавят null-проверку там, где настоящая проблема в том, что данные вообще не должны быть null.

Прежде чем чинить, ответьте на вопрос: «Почему этот дефект существует?» Если ответ — «не знаю», продолжайте расследование до наложения патча. Минимум: определить точку отказа и условие, которое её вызвало. Для сложных дефектов (конкурентность, состояние, внешние зависимости) — воспроизведите проблему для проверки гипотезы.

Гигиена контекста: не отправляйте персональные данные, учётные данные или регулируемую информацию в облачные AI-инструменты без Соглашения об обработке данных (DPA). Полные требования — в Standard Rule 10.12.

Правило 8. Фиксация знаний

Документируйте следующее во время или сразу после завершения задачи:

- **Тупики** — любой подход, на который ушло больше 15 минут и который был отвергнут. Запишите: что пробовали, почему не сработало, что выбрали вместо этого. (15 минут — отправная точка, адаптируйте под свою предметную область. На основе эмпирических наблюдений: после 15 минут затраты на документирование растут, а подход крайне редко приносит результат.)
- **Решения** — неочевидные выборы с обоснованием (например, «выбрали argon2 вместо bcrypt, потому что bcrypt не импортируется в Python 3.14»).
- **Известные проблемы** — ограничения, обходные решения или техдолг, внесённый задачей.

Храните знания там, где будущие AI-сессии смогут их найти — документация проекта, база знаний, комментарии в коде, что угодно рабочее. Знания, которые существуют только у вас в голове, для AI не существуют.

Зачем это нужно. У AI нет памяти между сессиями. Без зафиксированных знаний одни и те же тупики исследуются повторно, одни и те же решения обсуждаются заново, одни и те же обходные пути переоткрываются — с теми же затратами каждый раз.

Шлюзы качества

Два шлюза качества, через которые проходит каждая задача. Как их обеспечить — вручную, через инструментарий, на ревью — решайте сами. Метод не важен; важно, чтобы проверка состоялась.

ПРИМЕЧАНИЕ: Здесь используется термин «шлюзы» в соответствии с SENAR Standard (раздел 8), где Quality Gates — это точки принудительного контроля, а Checkpoints/ Контрольные точки (раздел 3.12) — это действия по сохранению контекста внутри сессии. Эти понятия различны.

Шлюз качества «Старт»

(В полном стандарте SENAR — QG-0)

Перед тем как писать код, проверьте:

#	Проверка	Правило
1	Цель определена (одно предложение)	Правило 1
2	Критерии приёмки перечислены (нумерованные, проверяемые)	Правило 1

3	Включён хотя бы один негативный сценарий	Правило 1
4	Область изменений определена (что менять, чего не трогать)	Правило 2
5	Для задач, связанных с безопасностью: поверхность угроз определена, критерии по безопасности включены	Правило 1

Если чего-то не хватает — дополните, прежде чем направить AI на работу. Пункт 5 относится к задачам, затрагивающим авторизацию, пользовательский ввод, хранение данных, платежи или внешние API.

Шлюз качества «Готово»

(В полном стандарте SENAR — QG-2)

Перед тем как объявить задачу завершённой, проверьте:

#	**Проверка**	**Правило**
1	Все критерии приёмки имеют зафиксированное свидетельство верификации	Правила 3, 6
2	Для тестируемых критериев есть автоматические тесты и они проходят; для нетестируемых — другое зафиксированное свидетельство	Правила 3, 4
3	Чеклист верификации пройден на соответствующем уровне	Правило 5
4	Корневая причина установлена для всех дефектов, найденных в ходе задачи	Правило 7
5	Тупики, решения и известные проблемы задокументированы (если нет — укажите «нет»)	Правило 8

Если что-то не выполнено — задача остаётся в работе.

Метрики

Два числа для отслеживания. Не задавайте целевые показатели сразу — измеряйте минимум 3 рабочих цикла, чтобы установить базовую линию.

FPSR — доля успеха с первой попытки

`задачи_завершённые_без_переделок / всего_завершённых_задач * 100%`

Процент задач, завершённых корректно с первой попытки — без дефектов после объявления «готово», без повторного открытия, без переделок. Главный индикатор того, работают ли шлюзы «Старт» и «Готово».

Команды, внедряющие Основы SENAR, обычно видят 50-65% в начале с ростом до 80-90% по мере выработки привычки. Это ориентировочные диапазоны — установите свои.

Показатель тупиков (Dead End Rate)

`время_на_тупики / общее_время_задачи * 100%`

Проще: `количество_тупиков / всего_задач` — среднее число тупиков на задачу.

Измеряет долю усилий, потраченных на отвергнутые подходы. Отражает эффективность фиксации знаний (правило 8). Снижение показателя — значит, база знаний работает.

Выше 30% — повод разобраться. Ниже 10% в зрелом проекте — эффективное переиспользование знаний.

Чеклист верификации

Этот чеклист реализует правило 5. Выбирайте уровень по степени риска задачи.

Тип задачи	Уровень	Пунктов
Баг-фикс, конфиг, простой CRUD	Облегчённый	4 (пункты 1, 3, 5, 7)
Обычная фича, рефакторинг	Standard	10
Auth, платежи, PII, внешние API	High	18
Прод-инцидент, регуляторика, сложная фича	Critical	28

Стандартный уровень — каждая задача

> ***Облегчённый режим для мелких задач:*** *Для задач короче 15 минут (баг-фиксы, изменения конфигурации, простой CRUD) проверьте минимум пункты 1, 3, 5 и 7 —*

> *соответствие области, подмену тестов, валидацию ввода и секреты. Полный стандартный уровень применяйте для всего, что затрагивает авторизацию, платежи или пользовательские данные.*

#	Проверка	На что обращать внимание
1	**Область изменений**	AI модифицировал файлы за пределами определённой области задачи?
2	**Удаления**	AI молча удалил или заменил существующий работающий код?
3	**Фантомные зависимости**	Все подключаемые зависимости реально существуют и доступны? (пакеты в манифесте, API доступны, сервисы объявлены)
4	**Качество тестов**	Тесты проверяют поведение из критериев приёмки или просто отражают реализацию?
5	**Подмена тестов**	AI подправил существующие тесты, чтобы они проходили, вместо исправления кода?
6	**Валидация входных данных**	Все пользовательские данные валидируются по типу, длине и формату перед использованием в запросах, файловых операциях или командах.
7	**Захардкоженные секреты**	Нет API-ключей, паролей, токенов или учётных данных в исходном коде.
8	**Устаревшие паттерны**	Код использует deprecated API, удалённые функции или паттерны устаревших версий фреймворков?
9	**Межфайловая согласованность**	Если тип, интерфейс или контракт изменён — все потребители обновлены?
10	**Качество кода**	Функции длиннее 200 строк, дублирование логики, игнорирование существующих утилит, нечитаемая структура. AI-код часто работает, но архитектурно одноразовый.

Высокий уровень — безопасность, авторизация, платежи, данные

Все пункты стандартного уровня, плюс:

#	Проверка	На что обращать внимание

11	**Обход null-проверок**	Сравнения, где обе стороны могут быть null/None/nil — `None == None` истинно, что обходит проверки доступа.
12	**Обход пустой конфигурации**	Проверки безопасности пропускаются, когда конфигурационное значение — пустая строка (например, `if secret and ...` проваливается в открытое состояние).
13	**Доверие заголовкам**	HTTP-заголовки (X-Forwarded-For, X-Partner-ID) используются для решений безопасности без проверки от вышестоящего прокси.
14	**IDOR**	Доступ к ресурсам по ID без проверки авторизации запрашивающего пользователя на этот конкретный ресурс.
15	**Шорткат return-True**	Функции контроля доступа, которые возвращают True или предоставляют доступ без явной проверки принадлежности.
16	**Покрытие авторизации**	Каждый эндпоинт, обращающийся к пользовательским данным, имеет обязательную аутентификацию; авторизация проверяется на уровне ресурса.
17	**Небезопасная десериализация**	Десериализация недоверенных данных без валидации (pickle, yaml.load без SafeLoader, eval/exec на входных данных).
18	**SSRF**	Подделка серверных запросов (Server-Side Request Forgery): URL из пользовательского ввода валидируются по allowlist; нет нефильтрованных запросов к произвольным URL.

Критический уровень — продакшен-инциденты, регуляторика, сложные фичи

Все пункты стандартного и высокого уровней, плюс:

#	Проверка	На что обращать внимание
19	**Версии зависимостей**	Указанные версии пакетов реальны и опубликованы в официальном реестре?
20	**Захардкоженные значения**	Магические числа, URL, вшитые в код (секреты проверяются на стандартном уровне).
21	**Переусложнение**	Лишние абстракции, паттерны проектирования или обобщения сверх требований задачи.

22	**Дублирование**	Новый код, дублирующий функциональность, уже доступную в существующих утилитах проекта.
23	**Граничные случаи**	Счастливый путь работает — а что с null, пустыми значениями, границами, конкурентным и высоконагруженным вводом?
24	**Именование**	AI следует соглашениям об именовании проекта последовательно?
25	**Область коммита**	Коммит атомарен и сфокусирован на задаче, или содержит несвязанные изменения?
26	**Инъекция через форматирование строк**	Строковая интерполяция или форматирование с недоверенным вводом (Python `str.format`, JS template literals с eval, C printf).
27	**Недостижимый защитный код**	Ветки кода «на всякий случай», которые никогда не выполнятся, маскируя неполноту управления потоком.
28	**Проглоченные исключения**	Блоки catch/except, которые молча отбрасывают ошибки — `except Exception: pass`, пустые catch-блоки, логирование ошибок на уровне debug.

Что дальше

Основы SENAR — это отправная точка. Когда команда растёт или нужно больше структуры, есть чёткая линия развития:

	Базовая	Начальная	Командная	Корпоративная
Пар	1	1–3	3–10	10+
Правил	8	11	15	15
Шлюзов качества	2	2 (QG-0, QG-2)	5 (QG-0..QG-4)	5 + комплаенс
Метрик	2	4	10	10 + портфель
Ролей	1	3 (совмещённые)	5 (выделенные)	5 + портфель
Церемоний	0	3	7	7 + портфель

Инструментарий	Не требуется	Рекомендуется	Обязателен	Обязателен

> ***Примечание о «парах»:*** *Пара — это один Супервайзер, работающий с одним или несколькими AI-агентами. Как правило, это один разработчик со своим AI-инструментом для кодирования.*

Следующий шаг после Основ SENAR: SENAR Начальная (Foundation) — добавляет управление сессиями, 3 совмещённые роли, 4 метрики. Занимает 2–4 недели после того, как привычки Основ SENAR закрепились. Подробности в полном SENAR Standard.

> ***Примечание:*** *Управление сессиями (структурированный старт и завершение AI-рабочих сессий) вводится на Начальном уровне. Core фокусируется на дисциплине отдельной задачи.*

Соответствие стандарту: Если вы соблюдаете все 8 правил и стабильно проходите оба шлюза, вы практикуете Основы SENAR. Никакого официального подтверждения не требуется — соответствие Core оценивается самостоятельно.

Правила Базовой конфигурации напрямую соответствуют правилам Standard — ничего не приходится переучивать:

Правило Базовой конфигурации	Эквивалент в Standard
1. Задача до кода	Rule 1 (S10.1) + QG-0 (S8.1)
2. Границы области изменений	QG-0 criteria + Guide habit
3. Верификация по критериям	QG-2 (S8.3) + Rule 15 L2 (S10.15)
4. Тесты проверяют требования	QG-2 test criteria + QG-3 (S8.4)
5. Проверка на скрытые дефекты	Rule 15 (S10.15) + AI Review Checklist
6. Нулевая терпимость к незавершённой работе	QG-2 criteria (S8.3)
7. Устраняй причины, а не симптомы	QG-2 criterion (S8.3) + Guide failure modes
8. Фиксация знаний	Rule 4 (S10.4) + Rule 9 (S10.9)

ПРИМЕЧАНИЕ: Правила Standard 10.2 Длительность сессии (Session Duration), 10.3 Каденция контрольных точек (Checkpoint Cadence), 10.5 Периодический аудит (Periodic Audit), 10.6 Контроль версий (Version Control), 10.7 Лимит параллельных агентов (Parallel Agent Limit), 10.8 Калибровка сложности и стоимости (Complexity-Cost Calibration), 10.10

Трассировка требований (Requirement Traceability), 10.11 Документирование кода (Code Documentation as Context), 10.12 Гигиена контекста (Context Hygiene), 10.13 Управление моделями AI (AI Model Governance), 10.14 Управление изменениями скриптов (Script Change Management) — это новые правила для конфигураций Начальная/Командная/Корпоративная, не имеющие эквивалента в Базовой.

Краткая справочная карта

Перед началом задачи (шлюз качества «Старт»):

1. Сформулируйте цель (одно предложение).
2. Перечислите критерии приёмки (нумерованные, каждый проверяется независимо).
3. Добавьте хотя бы один негативный сценарий.
4. Определите область: что менять, чего НЕ трогать.

Пока AI работает:

5. Верифицируйте каждый критерий приёмки свидетельством (запустите, протестируйте, измерьте).
6. Документируйте каждый тупик, который занял больше 15 минут.

Перед объявлением готовности (шлюз качества «Готово»):

7. Пройдите чеклист верификации на соответствующем уровне.
8. Подтвердите: все КП верифицированы, тесты проходят, чеклист пройден, знания зафиксированы.

После задачи:

9. Зафиксируйте решения, известные проблемы и всё неочевидное.
10. Спросите себя: сможет ли новый разработчик (или AI) продолжить эту работу, не задавая вопросов? Если нет — зафиксируйте недостающее.

SENAR

Supervised Engineering & Normative AI Regulation

Version: 1.3 | **Date:** 25.03.2026 **Authors:** Андрей Юмашев (Andrey Yumashev), Вадим Соглаев (Vadim Soglaev) | **Website:** senar.tech

---

> ***ПРИМЕЧАНИЕ:*** *Приведённое ниже описание носит информационный характер. Нормативные требования определены в Разделах 4–13.*

Начните здесь: Минимально жизнеспособный SENAR (MVS)

Вы практикуете SENAR, когда выполняются шесть условий:

1. **Каждая реализация с участием AI имеет Задачу** с целью и критериями приёмки — до начала работы.
2. **Результат AI верифицирован** Супервайзером по критериям приёмки — никогда не принят формально.
3. **Задачи не могут стартовать** без цели, критериев приёмки и связи с требованием (Контекстный шлюз).
4. **Задачи не могут быть закрыты**, если не пройден CI, тесты и проверка типов (Шлюз реализации).
5. **Пропускная способность и Lead Time измеряются** — вы знаете, сколько задач за сессию и сколько времени они занимают.
6. **Тупиковые подходы документируются** — когда подход терпит неудачу, вы фиксируете причину, чтобы никто не повторил ошибку.

Вот и всё. Выполняйте эти шесть пунктов — и вы практикуете ядро SENAR. Для структурированного входа см. Основы SENAR (8 правил, 2 шлюза, 2 метрики). Всё остальное в данном стандарте опирается на этот фундамент.

Что такое SENAR

SENAR — это методология разработки программного обеспечения, в которой AI-агенты являются основными производителями инженерных артефактов, а люди выступают в роли Супервайзеров — направляя, верифицируя и управляя результатами работы AI.

SENAR отвечает на фундаментальный сдвиг в программной инженерии: когда AI производит код, ценность человека смещается от **производства** (написания кода) к **экспертизе** (проектирование контекста, верификация корректности, архитектурные решения).

Существующие методологии (Scrum, SAFe, Kanban) создавались для команд людей, координирующихся между собой. SENAR создан для пар Супервайзер+AI, производящих верифицированное, прослеживаемое программное обеспечение.

Ценности SENAR

1. **Контекст важнее кода** — качество вывода AI определяется качеством входного контекста
2. **Верификация важнее скорости** — ограничивающий фактор — корректность, а не скорость
3. **Знания важнее опыта** — что не задокументировано, не существует для AI
4. **Принуждение важнее договорённости** — стандарты качества обеспечиваются автоматизированными шлюзами, а не совещаниями
5. **Суждение важнее нажатий клавиш** — внимание человека направлено на решения и верификацию, а не на набор кода

Основы SENAR

Основы SENAR (SENAR Core) содержат 8 базовых правил для любой команды. Данный Стандарт расширяет Основы организационными процессами, метриками и управлением. Командам, впервые внедряющим SENAR, РЕКОМЕНДУЕТСЯ начинать с Основ SENAR. Команды, практикующие Основы SENAR, не обязаны соответствовать данному Стандарту, но им рекомендуется переход на Начальную конфигурацию (Foundation) по мере готовности.

Набор документов

Документ	Назначение	Аудитория
Основы SENAR	8 базовых правил — входной уровень внедрения	Любая команда, индивидуальные

		Супервайзеры
Стандарт SENAR (данный документ)	Нормативные требования (ОБЯЗАН/ РЕКОМЕНДУЕТСЯ/ДОПУСКАЕТСЯ)	Организации, аудиторы
Гид SENAR	Философия, паттерны взаимодействия, обучение	Супервайзеры, внедряющие
Справочник SENAR	Глоссарий, коэффициенты масштабирования, экономическая модель, соответствие, инструментарий	Менеджеры, комплаенс

Нормативный язык

Согласно RFC 2119: **ОБЯЗАН** (SHALL) = обязательно, **РЕКОМЕНДУЕТСЯ** (SHOULD) = рекомендовано, **ДОПУСКАЕТСЯ** (MAY) = необязательно. В настоящем документе нормативные ключевые слова указаны на русском языке заглавными буквами: ОБЯЗАН (ОБЯЗАНА, ОБЯЗАНЫ, ОБЯЗАНО), НЕ ДОЛЖЕН (НЕ ДОЛЖНА, НЕ ДОЛЖНЫ), РЕКОМЕНДУЕТСЯ, НЕ РЕКОМЕНДУЕТСЯ, ДОПУСКАЕТСЯ.

Нормативные требования содержатся в Разделах 4–13 данного Стандарта. Гид и Справочник носят информационный характер.

Инструментарий

SENAR не предписывает конкретный инструмент, но на практике зависит от инструментов. Автоматизированные шлюзы качества, сбор метрик и управление знаниями требуют инструментальной поддержки. Подробные требования к возможностям инструментов см. в Справочнике SENAR (Требования к инструментарию).

Зависимость от возможностей AI

Ряд положений SENAR зависит от возможностей AI: они опираются на поведенческие особенности AI (типы галлюцинаций, ограничения контекстного окна, качество следования инструкциям), которые меняются со сменой поколений моделей. Организациям РЕКОМЕНДУЕТСЯ пересматривать такие положения при существенном изменении используемой модели AI (см. Section 10.13). Гид SENAR явно маркирует зависящие от возможностей положения.

Эмпирическая база и ограничения

Количественные рекомендации SENAR (базовые значения метрик, рекомендации по длительности сессий, оценки стоимости) получены на основе единственной эталонной реализации: 552 задачи, $989 затрат на AI, 38 сессий по 6 микросервисам. Это кейс-стади, а не контролируемый эксперимент. Эти числа следует рассматривать как иллюстративные ориентиры, а не универсальные целевые показатели — именно поэтому Section 9 требует сначала установить собственные базовые значения и лишь затем определять цели.

Ограничения: одна организация, самоотчётные метрики, дизайн «до/после» без контрольной группы, одно семейство моделей AI. Для подтверждения обобщаемости необходима независимая репликация в различных организациях, предметных областях и на разных моделях AI.

Интеллектуальное наследие

SENAR опирается на устоявшиеся основы программной инженерии: инженерию требований (IEEE 29148), модели стоимости качества (Boehm, 1981), зрелость процессов (CMMI/SEI), метрики потока (DORA, Accelerate), практики качества бережливого производства (First Pass Yield, Right First Time) и исследования совместной работы человека и AI. Вклад SENAR — конкретное применение и кодификация этих принципов для AI-нативной разработки, где AI-агенты являются основными производителями кода, а инженеры-люди выступают Супервайзерами. Методология не претендует на изобретение инженерии качества — она адаптирует её к модели производства, которой не существовало на момент создания предшествующих фреймворков.

Версионирование

Изменения нормативных требований публикуются как нумерованные версии Стандарта (1.0, 1.1, 2.0). Материалы Гида и Справочника могут обновляться между версиями Стандарта. Журнал изменений ведётся на senar.tech.

---

1. Область применения

1.1 Назначение

SENAR определяет методологию разработки программного обеспечения, в которой AI-агенты являются основными производителями инженерных артефактов, а инженеры-люди выступают в роли Супервайзеров — направляя, верифицируя и управляя процессом, основанным на AI.

1.2 Целевая аудитория

- **Организации**, переходящие к AI-нативной разработке программного обеспечения;
- **Супервайзеры** — инженеры, направляющие AI-агентов;
- **Менеджеры**, ответственные за поставку, качество и стоимость;
- **Поставщики инструментов**, создающие AI-нативные платформы разработки;
- **Аудиторы**, оценивающие практики качества AI-нативных команд.

1.3 Применимость

Данный стандарт применяется в случаях, когда:

a) AI-агенты генерируют существенную долю производственных артефактов (код, тесты, конфигурация, документация); b) инженеры-люди направляют, проверяют и утверждают результаты, сгенерированные AI; c) требуется прослеживаемость от требований до поставленных артефактов; d) обеспечение качества реализуется через автоматизированные механизмы.

SENAR не привязан к конкретному инструменту. Он применим к любой AI-платформе для разработки — автономным агентам, IDE-ассистентам, терминальным инструментам или собственным конвейерам.

1.4 За пределами области применения

a) Обучение, дообучение или оценка моделей AI; b) традиционная разработка, где люди пишут основную часть кода; c) управление организационными изменениями; d) конкретные реализации инструментов.

1.5 Связь с другими стандартами

SENAR развивает и адаптирует концепции устоявшихся методологий:

- **SAFe 6.0** — SENAR переосмысливает концепции SAFe для AI-нативных команд. Сравнительные примечания по SAFe приводятся по всему документу.
- **ISO 9001:2015** — шлюзы качества SENAR покрывают отдельные требования ISO 9001. Для полного соответствия организациям следует провести GAP-анализ.
- **ISO/IEC 25010:2023** — метрики SENAR согласованы с моделью качества программного обеспечения.
- **Scrum / Kanban** — SENAR заимствует итеративную поставку и измерение потока, заменяя ориентированные на людей церемонии альтернативами для работы с AI.

SENAR может быть дополнен доменными профилями для регулируемых отраслей (медицинские изделия, финансовые услуги, аэрокосмическая промышленность), добавляющими контроли, необходимые по отраслевым стандартам.

---

2. Нормативные ссылки

2.1 Нормативные ссылки

- **RFC 2119** — Ключевые слова для обозначения уровней требований в RFC
- **RFC 8174** — Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.

2.2 Информационные ссылки

Перечисленные стандарты упоминаются для контекста и согласованности, но не являются нормативными в рамках настоящего Стандарта.

- **ISO 9001:2015** — Системы менеджмента качества — Требования
- **ISO/IEC 25010:2023** — Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения — Модель качества продукта
- **SAFe 6.0** — Scaled Agile Framework (Scaled Agile, Inc.)
- **Scrum Guide** — The Scrum Guide (Schwaber & Sutherland, 2020)
- **Kanban Method** — Kanban: Successful Evolutionary Change for Your Technology Business (Anderson, 2010)
- **DORA Metrics** — Accelerate: The Science of Lean Software and DevOps (Forsgren, Humble & Kim, 2018)
- **ISO/IEC 12207:2017** — Системная и программная инженерия — Процессы жизненного цикла программного обеспечения
- **IEEE 29148:2018** — ISO/IEC/IEEE 29148:2018, Системная и программная инженерия — Процессы жизненного цикла — Инженерия требований.
- **CMMI** — CMMI Institute, "CMMI for Development, Version 2.0", 2018.

---

3. Термины и определения

Полный глоссарий см. в Справочнике SENAR.

Нотация конфигураций: Когда нормативные требования различаются в зависимости от конфигурации, в стандарте используется нотация `[Team+: ОБЯЗАТЕЛЬНО]` — требование является РЕКОМЕНДУЕТСЯ на Командном уровне, но ОБЯЗАТЕЛЬНО на Корпоративном уровне. Определения конфигураций см. в Section 11. Для начального уровня внедрения см. Основы SENAR.

3.1 AI-агент

Программная система на основе большой языковой модели, генерирующая инженерные артефакты под управлением человека. AI-агент поставляется Провайдером модели AI (3.25) и работает на конкретной версии модели.

AI-агенты — не стабильные детерминированные инструменты. Версии моделей различаются по возможностям, профилям галлюцинаций и качеству следования инструкциям. Смена версии модели рассматривается как изменение конфигурации (нормативные требования см. в Section 10.13).

3.2 Супервайзер

Инженер-человек, который направляет AI-агентов, верифицирует результаты, принимает архитектурные решения и обеспечивает соблюдение шлюзов качества. Основной режим работы — AI-направляемая разработка, с ручным кодированием как обоснованным исключением (Section 4.1).

3.3 Пара Супервайзер+AI

Фундаментальная производственная единица: один Супервайзер, работающий с одним или несколькими AI-агентами.

3.4 Контекст

Информация, предоставляемая AI-агенту для получения корректного результата: цель, критерии приёмки, ограничения, знания и связи прослеживаемости.

3.5 Задача

Атомарная единица отслеживаемой работы. Содержит цель, критерии приёмки и связь с требованием.

3.6 Исследование

Изучение вопроса без полной формальности Задачи. Исследованиям РЕКОМЕНДУЕТСЯ быть ограниченными по времени (Section 6.1). Если исследование приводит к реализации, создаётся Задача.

3.7 Сессия

Ограниченный по времени период супервизируемой работы с AI, с определёнными началом и окончанием.

3.8 Инкремент

Пакет работ с определённой областью, целями и запланированным бюджетом.

3.9 Шлюз качества

Автоматизированная точка контроля, блокирующая продвижение работы при невыполнении критериев.

3.10 Запись знаний

Задокументированное решение, паттерн, известная проблема или тупиковый подход, хранящиеся в поисковой базе знаний.

3.11 Тупиковый подход

Задокументированный неудачный подход с указанием причины отказа. Тупиковым считается любое исследование, занявшее более 15 минут и не давшее пригодного

результата. При достижении этого порога Супервайзер останавливается, фиксирует подход и причину неудачи и переходит к альтернативному пути. Нормативные требования к обработке тупиковых подходов см. в Section 10.4.

3.12 Контрольная точка

Сохранение контекста в ходе Сессии для предотвращения потери работы.

3.13 Обход шлюза

Задокументированное исключение, позволяющее провести работу через Шлюз качества. Требует обоснования, признания рисков и плана устранения.

3.14 Федерация

Механизм координации нескольких пар Супервайзер+AI в рамках одного или нескольких проектов: отслеживание зависимостей, общие знания, межпроектные оповещения. Требования к федерации при управлении несколькими проектами см. в Section 5.7.

3.15 Время цикла

Время от начала Задачи до её завершения (`started_at → completed_at`). Отличается от Lead Time тем, что не включает время ожидания в очереди (сравните с Lead Time: `created_at → completed_at`).

3.16 История

Промежуточная группировка Задач, представляющая поставку, видимую заинтересованным сторонам.

3.17 Требование

Задокументированное, верифицируемое утверждение о потребности, возможности или ограничении системы. SENAR определяет три уровня требований: Бизнес-требование (3.18), Системное требование (3.19) и Требование к задаче (3.20). Цель Задачи и критерии приёмки образуют требования на уровне Т3.

3.18 Бизнес-требование (БТ)

Потребность заинтересованных сторон, выраженная в бизнес-терминах. Источник всех нижестоящих требований. Обычно соответствует цели Инкремента или Эпика.

3.19 Системное требование (СТ)

Возможность или ограничение на уровне системы, выведенное из одного или нескольких Бизнес-требований. Описывается через поведение системы, а не через реализацию. Обычно соответствует цели Истории.

3.20 Требование к задаче (ТЗ)

Требование на уровне реализации, декомпозированное из Бизнес- или Системного требования. Соответствует цели и критериям приёмки Задачи. Самый низкий уровень формального управления требованиями; тест-кейсы — артефакты верификации, производные от ТЗ, а не отдельный уровень требований.

3.21 Иерархия требований

Цепочка декомпозиции от бизнес-потребности к единице реализации: БТ → СТ → ТЗ. Не для каждой работы нужны все уровни; глубину определяет Контекстный архитектор исходя из сложности и регуляторного контекста (см. Section 8.2).

3.22 Тестовая модель (ТМ)

Артефакт верификации, производный от Требований к задаче. Определяет способ проверки каждого ТЗ: тест-кейсы, тестовые данные, ожидаемые результаты и метод верификации (автоматический тест, ручная демонстрация, измерение). Тестовая модель — не уровень требований, а мост между требованиями и верификацией. В AI-нативной разработке AI обычно генерирует тесты из ТЗ; Супервайзер проверяет, что сгенерированные тесты действительно покрывают заявленные критерии приёмки.

Уровень формализации Тестовой модели зависит от конфигурации — см. Section 11 и QG-2 (Section 8.3) для нормативных требований.

3.23 Документация кода

Документация уровня модулей, API и архитектуры, служащая постоянным контекстом для AI-агентов. В AI-нативной разработке документация кода имеет двойную аудиторию: Супервайзеров-людей и AI-агентов. Самодостаточная, машиночитаемая документация снижает затраты на контекст для каждой Задачи и повышает качество вывода AI.

3.24 Прослеживаемость

Возможность проследить каждый инженерный артефакт до породившего его требования через цепочку связанных ссылок. Двунаправленная: каждое ТЗ прослеживается вверх до БТ; каждое БТ декомпозируется хотя бы в одно ТЗ. Полная цепочка прослеживаемости: БТ → СТ → ТЗ → ТМ → Код.

3.25 Провайдер модели AI

Внешний сервис, предоставляющий возможности инференса AI-моделей (облачные API, локальные серверы моделей). Провайдеры моделей AI — де-факто поставщики: модель AI является основным производственным инструментом, эквивалентным компилятору. Возможности, ограничения и ценообразование модели меняются по решению провайдера, вне контроля организации.

3.26 Версия модели AI

Конкретный релиз модели AI, идентифицируемый обозначением провайдера. Смена версии может повлиять на качество вывода, стоимость и поведение модели. Версии различаются по возможностям, профилям галлюцинаций, стоимости и качеству следования инструкциям. Базовые значения метрик (FPSR, стоимость/задача) зависят от версии — требования к рекалибровке см. в Section 10.13.

3.27 Расползание области

Незапланированные изменения в реализации Задачи, выходящие за рамки заявленной цели и критериев приёмки. В AI-нативной разработке расползание области проявляется, когда AI-агент правит код за пределами определённых границ, добавляет незапрошенные функции или рефакторит код, не покрытый Задачей.

3.28 Галлюцинация (AI)

Вывод AI, который выглядит правдоподобно, но фактически некорректен: ссылки на несуществующие API, методы, флаги CLI или пакеты; выдуманные пути к файлам; уверенные, но ошибочные утверждения о поведении системы. В контексте зависимостей галлюцинированный пакет — пакет, которого нет в официальном реестре или который ведёт к неожиданному мейнтейнеру.

3.29 WSJF (Weighted Shortest Job First)

Метод приоритизации: отношение Стоимости задержки к Размеру работы. Применяется при Планировании инкремента (Section 7.1) для упорядочивания пула задач. Заимствован из SAFe без изменений.

3.30 Поток создания ценности

Сквозной поток от запроса заинтересованной стороны до поставленного, верифицированного программного обеспечения. В Корпоративной конфигурации (Enterprise) Инкременты группируются по потокам создания ценности с едиными бюджетами (Section 11.3).

3.31 Состязательное ревью

Независимая проверка результатов AI агентом, не имеющим доступа к контексту сессии или ходу рассуждений генерирующего агента. См. Section 10.15, L3.

3.32 Диспетчеризация агента

Делегирование Задачи или подзадачи отдельному экземпляру AI-агента, обычно работающему в изолированной среде. См. Section 5.6.

3.33 Профиль агента

Именованный набор скриптов, разрешений и контекста, определяющий возможности и границы AI-агента для конкретной функции. См. Section 5.2.

3.34 Структурированный протокол инструментов

Протокол структурированного взаимодействия AI-агентов с сервисами платформы: самоописывающиеся схемы инструментов, атомарные операции и журналирование

аудита. См. Section 5.5.

NOTE: Примеры подходящих протоколов: Model Context Protocol (MCP), вызов функций OpenAI, пользовательские API.

3.35 Операционный скрипт

Структурированная инструкция на естественном языке: как AI-агент выполняет конкретное действие. Содержит триггер, предусловия, алгоритм, постусловия и выходные данные. См. Section 5.3.

3.36 Adversarial Detection Rate (ADR)

Число критических находок состязательного ревью на одну задачу, прошедшую ревью L3. Формула: `adversarial_critical_findings / L3_reviewed_tasks`. См. Section 9.2.

3.37 Стандарты кода

Документ с обязательными правилами качества кода, загружаемый в контекст AI-агента. Охватывает безопасность, архитектуру, базы данных, API, конкурентность и паттерны тестирования. См. Section 10.15, L2.

3.38 Скрытый дефект

AI-сгенерированный код, который выглядит корректным — проходит автоматические проверки, валидацию типов и самопроверку — но содержит скрытые дефекты (обход защиты, логические ошибки, архитектурные нарушения), обнаруживаемые только при независимом состязательном ревью. Скрытые дефекты возникают, когда AI генерирует код по паттернам, а не на основе семантического понимания. См. Section 10.15.

3.39 FPSR (First-Pass Success Rate)

Процент Задач, прошедших верификацию с первой попытки — без возврата на доработку. Определение и требования к измерению см. в Section 9.1.

3.40 Обзор качества

Проверка в конце Инкремента: метрики, открытые дефекты, покрытие базы знаний, полнота прослеживаемости. Нормативные требования см. в Section 7.4.

3.41 Менеджер потока

Роль Супервайзера: межкомандная координация, отслеживание зависимостей и эффективность потока в рамках Федерации. Определение и обязанности см. в Section 4.4.

3.42 Контекстный архитектор

Роль Супервайзера: проектирование и поддержание архитектуры знаний, стратегий загрузки контекста и стандартов документации для эффективной AI-направляемой работы. Определение и обязанности см. в Section 4.2.

3.43 Инженер знаний

Роль Супервайзера: фиксация, структурирование и поддержание Записей знаний, тупиковых подходов и повторно используемых паттернов организации. Определение и обязанности см. в Section 4.3.

3.44 Инженер верификации

Роль Супервайзера: проектирование Тестовой модели, проведение состязательных ревью, контроль соблюдения Шлюзов качества. Определение и обязанности см. в Section 4.5.

---

4. Роли

SENAR определяет пять наборов обязанностей. Один человек МОЖЕТ совмещать несколько наборов. Все обязанности ОБЯЗАНЫ быть закрыты, но не обязательно выделенными позициями.

Карьерные траектории, руководства по переходу и коэффициенты масштабирования см. в Гиде SENAR и Справочнике.

4.1 Супервайзер (ОБЯЗАТЕЛЬНО — все конфигурации)

Направляет AI-агентов, верифицирует результаты, принимает архитектурные решения, контролирует соблюдение Шлюзов качества.

Обязанности: a) формулировать Задачи с целями и критериями приёмки до начала работы; b) предоставлять AI-агенту структурированный контекст; c) направлять AI в ходе выполнения, корректируя курс; d) проверять все результаты AI по критериям приёмки; e) принимать архитектурные решения и решения о компромиссах; f) контролировать прохождение Шлюзов качества при закрытии Задач; g) фиксировать записи знаний в ходе работы; h) поддерживать дисциплину Сессий (контрольные точки, длительность).

Супервайзер отдаёт приоритет AI-генерации, но МОЖЕТ писать код вручную, когда стоимость подготовки контекста превышает стоимость ручного вмешательства. Ручные вмешательства РЕКОМЕНДУЕТСЯ делать прослеживаемыми.

4.2 Контекстный архитектор (ОБЯЗАТЕЛЬНО — все конфигурации)

Проектирует требования как структурированный вход для AI, управляет иерархией требований (БТ → СТ → ТЗ) и обеспечивает прослеживаемость. Качество закладывается на входе: дефект в Бизнес-требовании каскадирует на все нижестоящие Системные требования, Требования к задачам и в конечном счёте — на код. Контекстный архитектор — главный хранитель качества входных данных.

Начальная (совмещение с Супервайзером — Section 4.8, Section 11.1): На Начальном уровне Супервайзер берёт на себя обязанности Контекстного архитектора. Минимальные обязанности Начального уровня: a) поддерживать файл контекста проекта (например, CLAUDE.md) в актуальном состоянии; b) перед началом задачи

проверять качество контекста — входные данные QG-0 должны быть достаточны и хорошо структурированы; c) проектировать начальную декомпозицию задач (минимум БТ → ТЗ).

Командная (ОБЯЗАТЕЛЬНО — выделенная роль): a) поддерживать иерархию требований (БТ → СТ → ТЗ) с двунаправленной прослеживаемостью; b) управлять жизненным циклом требований: черновик → утверждено → верифицировано → устарело; c) обеспечивать свойства качества требований: верифицируемость (ОБЯЗАТЕЛЬНО), согласованность, достаточность, неизбыточность, прослеживаемость [Командная+: ОБЯЗАТЕЛЬНО]; d) приоритизировать работу с использованием WSJF (Section 3.29); e) периодически проверять прослеживаемость: выявлять осиротевшие требования (без Задач) и осиротевшие Задачи (без требований); f) способствовать повторному использованию требований, сохраняя верифицированные требования как записи Базы знаний; g) проектировать Профили агентов и управлять Операционными скриптами (Section 5); h) проверять изменения скриптов как изменения производственного процесса (Section 10.14); i) поддерживать инвентарь инструментов по каждому Профилю агента (Section 5.4).

4.3 Инженер знаний (ОБЯЗАТЕЛЬНО Командная+)

Фиксирует, курирует и поддерживает организационную базу знаний.

NOTE: В Начальной конфигурации (Foundation) обязанности Инженера знаний совмещены с ролью Инженера верификации — их закрывает один человек (Section 4.8, Section 11.1). Обозначение «Командная+» означает выделенную роль; на Начальном уровне обязанности совмещаются. Выделенный Инженер знаний необходим начиная с Командной конфигурации (Team).

Обязанности: a) фиксировать тупиковые подходы, решения, паттерны, известные проблемы; b) поддерживать записи в поисковом и категоризированном виде; c) проверять актуальность знаний, депрецировать устаревшие записи; d) выявлять пробелы в знаниях.

4.4 Менеджер потока (ОБЯЗАТЕЛЬНО Командная+)

Управляет ритмом Сессий, учётом затрат и метриками потока.

NOTE: В Начальной конфигурации обязанности Менеджера потока берёт на себя Супервайзер (Section 4.8, Section 11.1). Обозначение «Командная+» означает выделенную роль; на Начальном уровне обязанности совмещаются.

Обязанности: a) мониторить длительность Сессий и каденцию контрольных точек; b) отслеживать стоимость на Задачу и Инкремент относительно бюджета; c) мониторить метрики потока (Section 9); d) фасилитировать Планирование инкремента и Ретроспективу; e) координировать зависимости между Парами (Федерация); f) планировать Обзоры качества.

На Корпоративном масштабе расширяется до координации нескольких Пар (Координатор федерации).

4.5 Инженер верификации (ОБЯЗАТЕЛЬНО Командная+)

Аудирует результаты, сгенерированные AI, на корректность, безопасность и соответствие архитектуре.

NOTE: В Начальной конфигурации обязанности Инженера верификации совмещены с ролью Инженера знаний — их закрывает один человек (Section 4.8, Section 11.1). Обозначение «Командная+» означает выделенную роль; на Начальном уровне обязанности совмещаются.

Обязанности: a) проводить Обзоры качества; b) аудировать результаты AI по Чеклисту проверки AI-вывода (см. Гид SENAR); c) проверять AI-сгенерированные тесты на покрытие и качество ассертов; d) отслеживать Defect Escape Rate и выявлять системные проблемы.

4.6 Роли Корпоративного уровня (только Корпоративная конфигурация (Enterprise))

На Корпоративном масштабе (10+ Пар) возникают три дополнительных набора обязанностей:

Портфельный менеджер: координирует несколько Инкрементов в потоках создания ценности. Управляет единым бюджетом, межпоточными зависимостями и стратегической приоритизацией. Следит за тем, чтобы инвестиции в AI соответствовали целям организации.

Главный Супервайзер: устанавливает архитектурные стандарты для всех Пар. Проверяет и утверждает архитектурные исключения. Определяет общеорганизационные паттерны взаимодействия с AI и шаблоны контекста. Примечание: Главный Супервайзер определяет общеорганизационные стандарты; Контекстный архитектор (Section 4.2) реализует их в конкретных проектах.

Координатор федерации: отвечает за межпроектное отслеживание зависимостей, общую базу знаний и межкомандную прослеживаемость требований. Фасилитирует церемонии Синхронизации федерации (Section 7.5) на уровне портфеля.

Это наборы обязанностей, а не должности. Один человек МОЖЕТ охватывать несколько ролей Корпоративного уровня.

4.7 Сводка

Обязанность	Командная	Корпоративная
Супервайзер	ОБЯЗАТЕЛЬНО (каждая Пара)	ОБЯЗАТЕЛЬНО (каждая Пара)
Контекстный архитектор	ОБЯЗАТЕЛЬНО (выделенный)	ОБЯЗАТЕЛЬНО (выделенный)
Инженер знаний	ОБЯЗАТЕЛЬНО (выделенный)	ОБЯЗАТЕЛЬНО (выделенный)
Менеджер потока	ОБЯЗАТЕЛЬНО (выделенный)	ОБЯЗАТЕЛЬНО (выделенный)
Инженер верификации	ОБЯЗАТЕЛЬНО (выделенный)	ОБЯЗАТЕЛЬНО (выделенный)
Портфельный менеджер	—	ОБЯЗАТЕЛЬНО
Главный Супервайзер	—	ОБЯЗАТЕЛЬНО
Координатор федерации	—	ОБЯЗАТЕЛЬНО

NOTE: Для начального уровня внедрения (1–2 Пары) см. Основы SENAR. Роль Супервайзера в Основах вбирает все обязанности.

4.8 Комбинации ролей по размеру команды

Размер команды	Паттерн совмещения
1 Пара (Базовая)	Супервайзер неформально закрывает все обязанности
1–3 Пары (Начальная)	Супервайзер + Контекстный архитектор (совмещены), Инженер знаний + Инженер верификации (совмещены). 2 человека закрывают все роли.
3–5 Пар (Командная)	Контекстный архитектор + Менеджер потока (совмещены), Инженер знаний (выделенный или совместный), Инженер верификации (выделенный), Супервайзеры (каждая пара). Минимум 3 различных носителя ролей.

5–10 Пар (Командная)	Все 5 ролей выделены. Контекстный архитектор может быть совмещён с Менеджером потока при 5–6 парах.
10+ Пар (Корпоративная)	Все роли выделены + роли Корпоративного уровня (Портфельный менеджер, Главный Супервайзер, Координатор федерации).

NOTE: Комбинации ролей — рекомендация уровня РЕКОМЕНДУЕТСЯ. Организации МОГУТ выбирать другие комбинации исходя из навыков команды и предметной области. Главное — чтобы все обязанности были закрыты, а не чтобы использовались конкретные комбинации.

---

5. Инструментирование агентов

5.1 Обзор

AI-нативная разработка требует явного управления поведением, возможностями и ограничениями AI-агента. Данный раздел определяет, как организации ОБЯЗАНЫ конфигурировать и версионировать средства контроля поведения AI-агента.

Инструментирование агентов действует на трёх уровнях:

Уровень	Артефакт	Назначение	Аналогия
Поведенческий контракт	Файл правил проекта	Определить границы, запреты, конвенции	Должностная инструкция
Операционные скрипты	Процедурные инструкции	Алгоритмические шаги для конкретных действий	Рабочая инструкция на станок
Программный интерфейс	API / Инструменты / Хуки	Инструменты взаимодействия агента с платформой	Панель управления станком

Каждый уровень ОБЯЗАН быть под контролем версий и подчиняться управлению изменениями (см. Section 10.14).

5.2 Профили агентов

Профиль агента — конкретный набор скриптов, разрешений и контекста для AI-агента, выполняющего определённую функцию.

Организации ОБЯЗАНЫ определить как минимум следующие профили:

Профиль	Скрипты	Доступ	Назначение
Генератор	Реализация, коммит, отладка	Чтение/запись кода и задач	Основная разработка
Ревьюер	Ревью, аудит безопасности	Только чтение кода и задач	Независимая верификация

Планировщик	Планирование, ретроспектива	Запись эпиков/ историй, чтение всего	Архитектура, декомпозиция
Документатор	Документирование	Запись в базу знаний	Документация и фиксация знаний
Верификатор	Обзор качества, тестирование	Чтение всего, запись находок	Аудит качества

Разделение ответственности: профиль Ревьюера НЕ ДОЛЖЕН иметь права записи в проверяемые артефакты. Это — основа состязательного ревью: один и тот же агент не может одновременно генерировать и утверждать собственный вывод.

Организации МОГУТ определять дополнительные профили. Один физический AI-агент МОЖЕТ переключаться между профилями в рамках сессии при условии, что каждое переключение профиля журналируется.

NOTE: В Начальной конфигурации (Foundation) (1–3 Пары) организации МОГУТ использовать менее пяти профилей. Требование 5 профилей является РЕКОМЕНДУЕТСЯ для Начальной, ОБЯЗАТЕЛЬНО для Командная+. Команды Начальной конфигурации обычно объединяют Генератор и Ревьюер в меньшее число профилей, соответствующих их совмещённой ролевой структуре (Section 4.8, Section 11.1).

Командная конфигурация (Team): организации ОБЯЗАНЫ определить все пять профилей с принудительными границами разрешений. Корпоративная конфигурация: организации ОБЯЗАНЫ определить все пять профилей с принудительными границами разрешений и аудиторским следом переключений профилей.

5.3 Операционные скрипты

Операционный скрипт — структурированная инструкция на естественном языке, определяющая, как AI-агент выполняет конкретное действие. Скрипты — основной механизм кодификации организационного процесса в поведение агента.

5.3.1 Структура скрипта

Каждому скрипту РЕКОМЕНДУЕТСЯ содержать:

- **Триггер**: когда скрипт вызывается (команда, событие, условие)
- **Предусловия**: что должно быть истинным перед выполнением
- **Алгоритм**: пронумерованные шаги с точками принятия решений
- **Постусловия**: что должно быть истинным после выполнения

- **Выходные данные**: что скрипт производит (артефакты, изменения состояния, отчёты)

5.3.2 Управление скриптами

Скрипты определяют поведение агента — изменение скрипта равнозначно изменению производственного процесса. Операционное правило управления изменениями скриптов см. в Section 10.14.

ОБЯЗАТЕЛЬНО (все конфигурации):

- операционные скрипты хранятся в системе контроля версий
- изменения скриптов проверяются перед развёртыванием (как изменения кода)
- решение об изменении скрипта фиксируется в базе знаний с обоснованием

ОБЯЗАТЕЛЬНО (конфигурации Командная+):

- изменения скриптов тестируются на изолированном проекте перед распространением на все проекты
- ведётся реестр активных скриптов с идентификаторами версий
- обеспечивается возможность отката: любое изменение скрипта может быть возвращено к предыдущей версии
- ведётся аудиторский след изменений скриптов

РЕКОМЕНДУЕТСЯ:

- скрипты имеют критерии приёмки (что скрипт делает, не делает, граничные случаи)
- эффективность скриптов отслеживается через метрики (например, FPSR задач, выполненных под управлением данного скрипта)

5.4 Программный интерфейс

Программный интерфейс определяет инструменты, API и автоматизированные действия, доступные AI-агенту.

5.4.1 Инвентарь инструментов

Организации ОБЯЗАНЫ поддерживать инвентарь инструментов, доступных каждому Профилю агента:

- какие API-эндпоинты доступны
- какие операции с файловой системой разрешены

- какие внешние сервисы достижимы
- какие действия требуют подтверждения человеком

5.4.2 Принцип наименьших привилегий

Каждый Профиль агента ОБЯЗАН иметь доступ только к инструментам, необходимым для его функции:

- Ревьюер НЕ ДОЛЖЕН иметь доступа на запись в продакшен-код
- Генератор НЕ ДОЛЖЕН иметь доступа к учётным данным развёртывания
- Верификатор НЕ ДОЛЖЕН иметь доступа к изменению результатов шлюзов качества

5.4.3 Автоматизированные действия (хуки)

Организации МОГУТ определять автоматизированные действия, запускаемые событиями:

- после сессии: сбор метрик, синхронизация знаний
- после завершения задачи: валидация шлюза качества, уведомление
- перед коммитом: проверка линтером, сканирование безопасности

Хуки ОБЯЗАНЫ быть под контролем версий наряду со скриптами.

5.5 Границы безопасности

5.5.1 Защита от инъекций промптов

Организации ОБЯЗАНЫ обеспечить, чтобы AI-агенты не обрабатывали ненадёжный пользовательский контент как инструкции. Когда агент читает внешние данные (пользовательский ввод, содержимое файлов, ответы API), они ОБЯЗАНЫ рассматриваться как данные, а не как команды. Операционным скриптам РЕКОМЕНДУЕТСЯ включать явные границы инструкций.

Практические меры:

a) промпты агентов ОБЯЗАНЫ чётко разграничивать системные инструкции и пользовательские данные; b) внешние данные, загружаемые агентами (ответы API, содержимое файлов, записи базы данных), НЕ ДОЛЖНЫ интерпретироваться как команды агента или модификации поведенческого контракта агента; c) организациям РЕКОМЕНДУЕТСЯ тестировать конфигурации агентов на известные паттерны инъекций промптов в рамках Обзоров качества (Section 7.4).

5.6 Структурированный протокол инструментов

Организации ОБЯЗАНЫ предоставлять возможности платформы через структурированный самоописывающийся протокол инструментов, а не через CLI-команды или прямой доступ к базе данных. Выбранному протоколу РЕКОМЕНДУЕТСЯ обеспечивать:

a) самоописывающиеся схемы — агент получает определения инструментов с типами параметров и описаниями, что снижает число галлюцинированных аргументов; b) атомарные операции — каждый вызов инструмента представляет собой единую транзакцию, исключая режим отказа «цепочки команд»; c) структурированный ввод/вывод — исключение ошибок парсинга текстового CLI-вывода; d) журналирование аудита — каждый вызов инструмента записывается с параметрами и результатами.

NOTE: Примеры подходящих протоколов: Model Context Protocol (MCP), вызов функций OpenAI, пользовательские REST/gRPC API инструментов.

CLI как резерв: CLI-командам РЕКОМЕНДУЕТСЯ оставаться доступными, когда структурированный протокол инструментов недоступен. Организации ОБЯЗАНЫ документировать, какие операции имеют CLI-резерв, а какие работают только через протокол.

Прямой доступ к базе данных: AI-агенты НЕ ДОЛЖНЫ использовать прямой доступ к базе данных для операций записи. Операции чтения МОГУТ использовать прямой доступ для отладки, но продуктивные рабочие процессы ОБЯЗАНЫ использовать абстракции протокола или CLI.

5.7 Диспетчеризация агентов и изоляция выполнения

Диспетчеризация агентов — делегирование Задачи или подзадачи отдельному экземпляру AI-агента — несёт специфические риски:

a) делегированный агент работает с ограниченным контекстом (нет истории сессии, ограниченные знания); b) несколько делегированных агентов могут одновременно модифицировать одни и те же файлы; c) Супервайзер не может наблюдать за ходом рассуждений делегированного агента в реальном времени.

Организации, использующие диспетчеризацию агентов, ОБЯЗАНЫ:

1. **Изолировать**: делегированным агентам РЕКОМЕНДУЕТСЯ работать в изолированных рабочих копиях для предотвращения конфликтов файлов; NOTE: Примеры механизмов изоляции: рабочие деревья системы контроля

версий, отдельные клоны репозитория, контейнеризованные среды сборки, эфемерные облачные пространства.

2. **Ограничить область**: промпты диспетчеризации ОБЯЗАНЫ включать явные границы — какие файлы модифицировать, какие оставить без изменений;
3. **Проверить**: все результаты делегированных агентов ОБЯЗАНЫ проходить Состязательное ревью L3 (Section 10.15) — диспетчеризация агентов является сценарием с наивысшим риском скрытых дефектов. Для Начальной конфигурации допускается замена на Ревью L2 с чеклистом высокого уровня, если независимый доступ к агенту недоступен (Section 10.15);
4. **Контекст**: промптам диспетчеризации РЕКОМЕНДУЕТСЯ включать релевантные Стандарты кода, архитектурные ограничения и известные проблемы из базы знаний;
5. **Лимитировать**: организации ОБЯЗАНЫ определить максимальное число параллельных диспетчеризаций на Супервайзера (применяется Section 10.7).

Делегированные агенты НЕ ДОЛЖНЫ:

- модифицировать файлы за пределами определённой области без явного одобрения;
- коммитить напрямую в общие ветки;
- иметь доступ к изолированным рабочим копиям или состоянию сессии других агентов.

Паттерн: Супервайзер делегирует → агент работает в изолированной копии → агент возвращает результат → Супервайзер проверяет → Супервайзер сливает в основную ветку.

5.8 Федерация — масштабирование SENAR на несколько проектов

Когда организация управляет несколькими проектами («федерация»), практики SENAR масштабируются с дополнительными требованиями к координации.

5.8.1 Независимость проектов

Каждый проект в федерации ОБЯЗАН поддерживать собственные:

- трекер задач и историю сессий
- базу знаний (паттерны, известные проблемы, тупиковые подходы)
- базовые значения и целевые показатели метрик
- конфигурацию агентов (профили, скрипты, разрешения)

5.8.2 Межпроектная координация

Федерации РЕКОМЕНДУЕТСЯ назначить координационный проект (аналог Release Train Engineer в SAFe), ответственный за:

- межпроектное отслеживание зависимостей
- маршрутизацию знаний в федерации (паттерн, обнаруженный в Проекте A, затрагивающий Проект B)
- агрегированные метрики (пропускная способность, стоимость, качество на уровне федерации)
- общие Стандарты кода и критерии ревью

5.8.3 Маршрутизация знаний

Записи знаний ОБЯЗАНЫ быть типизированы по области:

- **Проектные**: паттерны, известные проблемы и решения, релевантные только одному проекту — хранятся в базе знаний этого проекта;
- **Межпроектные**: паттерны, затрагивающие несколько проектов (изменения контрактов API, обновления общих библиотек) — хранятся в координационном проекте и маршрутизируются в затронутые проекты;
- **Глобальные**: методологические инсайты — хранятся в координационном проекте, доступны всем.

Организации ОБЯЗАНЫ определить правила маршрутизации: какие типы знаний распространяются автоматически, какие — с ручным повышением.

Межпроектные записи знаний ОБЯЗАНЫ требовать одобрения Супервайзера принимающего проекта перед включением в активный контекст этого проекта. Глобальные записи знаний ОБЯЗАНЫ требовать одобрения Супервайзера координационного проекта. Записи знаний, касающиеся тем безопасности (аутентификация, авторизация, шифрование, секреты, CORS, CSRF, разрешения), ОБЯЗАНЫ помечаться для проверки человеком независимо от правил маршрутизации.

5.8.4 Метрики федерации

Метрики уровня федерации агрегируют метрики проектов:

Метрика	Вычисление на уровне федерации
Пропускная способность	Сумма пропускных способностей проектов (задачи/сессия по всем проектам)
FPSR	Взвешенное среднее по количеству задач проекта

DER	Взвешенное среднее по количеству задач проекта
ADR	Взвешенное среднее по количеству задач агента проекта
Стоимость	Сумма стоимостей проектов

Организации НЕ ДОЛЖНЫ сравнивать «сырые» значения метрик между проектами с разными стеками, размерами команд или уровнями зрелости. Для сравнения РЕКОМЕНДУЕТСЯ использовать нормализованные метрики (например, стоимость на стори-поинт с учётом сложности).

5.9 Портируемость

Стандарт не привязан к конкретному AI-агенту, инструменту или платформе.

Операционные скрипты РЕКОМЕНДУЕТСЯ писать на структурированном естественном языке, интерпретируемом любым AI-агентом достаточной мощности:

- чёткие алгоритмические шаги (не платформоспецифичные вызовы SDK)
- явные входные и выходные данные на каждом шаге
- условная логика, выраженная на естественном языке
- никаких допущений о конкретных реализациях инструментов

Программный интерфейс МОЖЕТ быть платформоспецифичным (структурированные протоколы инструментов, API вызова функций и т.д.), но Поведенческий контракт и Операционные скрипты ОБЯЗАНЫ быть портируемыми между реализациями AI-агентов.

При миграции между AI-платформами организации ОБЯЗАНЫ:

- убедиться, что все Операционные скрипты корректно выполняются на новой платформе
- рекалибровать базовые значения метрик (Section 10.13: Управление моделью AI)
- обновить уровень Программного интерфейса, сохранив уровни Контракта и Скриптов

---

6. Единицы работы

6.1 Исследование

Ограниченное по времени изучение вопроса без формальных атрибутов Задачи.

a) исследованиям РЕКОМЕНДУЕТСЯ быть ограниченными по времени (лимит определяется организацией); b) если исследование приводит к реализации, Задача ОБЯЗАНА быть создана до коммита изменений; c) исследования РЕКОМЕНДУЕТСЯ журналировать с кратким описанием.

6.2 Задача

Атомарная единица отслеживаемой работы.

Обязательные атрибуты

Атрибут	Описание
Цель	Что должно быть достигнуто
Критерии приёмки	Верифицируемые условия, определяющие «готово»
Связь с требованием	Ссылка на родительскую Историю, Бизнес-требование, Системное требование или Требование к задаче (ОБЯЗАТЕЛЬНО)
Тип работы	Функциональная категория (разработка, архитектура, QA, документация)

Состояния жизненного цикла

Переход	Шлюз
planning → active	QG-0
active → done	QG-2
done → merged (Командная+)	QG-3

merged → released (Командная+)	QG-4
active → blocked	Нет
blocked → active	Нет

На уровне Командной конфигурации (Team) и выше состояние `done` ведёт к ревью слияния (QG-3) и одобрению релиза (QG-4).

6.3 История

Группировка Задач, представляющая поставку, видимую заинтересованным сторонам. Связывает Задачи с требованиями.

Обязательные атрибуты

Атрибут	Описание
Заголовок	Описание поставки одним предложением
Критерии приёмки	Верифицируемые условия уровня Истории, определяющие «готово» (независимо от индивидуальных КП Задач)
Связанные задачи	Ссылки на все Задачи, составляющие данную Историю
Связь с требованием	Ссылка на родительское Бизнес-требование (БТ) или Системное требование (СТ) (ОБЯЗАТЕЛЬНО)

Критерии завершения

История считается завершённой, когда все её Задачи достигли состояния `done` И критерии приёмки уровня Истории верифицированы Супервайзером.

6.4 Сессия

Ограниченный по времени период супервизируемой работы с AI.

a) организации ОБЯЗАНЫ установить и задокументировать максимальную длительность Сессии; b) контрольные точки ОБЯЗАНЫ выполняться с интервалами, задокументированными организацией; c) Сессия ОБЯЗАНА начинаться с загрузки контекста и завершаться фиксацией метрик и передачей контекста; d) церемонии МОГУТ быть автоматизированными командами инструментария; e) дисциплина

контроля версий: коммиты ОБЯЗАНЫ быть атомарными; обнаружение секретов ОБЯЗАНО быть автоматизировано; изменения AI ОБЯЗАНЫ проверяться на расползание области.

6.5 Инкремент

Пакет работ с определённой областью, целями, запланированным бюджетом и реестром рисков.

a) каждый Инкремент ОБЯЗАН иметь 3–5 измеримых целей; b) пул задач РЕКОМЕНДУЕТСЯ приоритизировать (рекомендуется WSJF); c) каждый Инкремент завершается Обзором качества и Ретроспективой.

---

7. Церемонии

Церемонии — точки принятия стратегических решений людьми. За обеспечение качества отвечают Шлюзы (Section 8).

7.1 Планирование инкремента (ОБЯЗАТЕЛЬНО)

Определение целей, пула задач, бюджета и рисков для предстоящего Инкремента.

Участники	**Контекстный архитектор (ведёт), Менеджер потока, Супервайзеры**
Частота	Один раз за Инкремент

Результат: цели, приоритизированные задачи, запланированный бюджет, реестр рисков.

7.2 Начало сессии (ОБЯЗАТЕЛЬНО)

Загрузка контекста, выбор задач, проверка окружения.

Участники	**Супервайзер**
Накладные расходы	Минимальные (МОГУТ быть автоматизированы)

7.3 Окончание сессии (ОБЯЗАТЕЛЬНО)

Фиксация метрик, запись передачи контекста, фиксация знаний.

Участники	**Супервайзер**
Накладные расходы	Минимальные (МОГУТ быть автоматизированы)

ОБЯЗАТЕЛЬНЫЕ выходные данные: резюме, состояния задач, передача (следующие шаги, предупреждения), метрики, тупиковые подходы (если были). **РЕКОМЕНДУЕМЫЕ выходные данные:** записи знаний о решениях/паттернах/известных проблемах.

7.4 Обзор качества (ОБЯЗАТЕЛЬНО)

Периодический комплексный аудит кодовой базы и Базы знаний.

Участники	Инженер верификации (ведёт)
Частота	С каденцией, задокументированной организацией

Охват: безопасность, качество кода, здоровье тестов, дрейф конфигурации, актуальность знаний, соответствие архитектуре, здоровье зависимостей, AI-специфичные проблемы (дублирование, накопленное расползание области, мусорные TODO).

7.5 Синхронизация федерации (РЕКОМЕНДУЕТСЯ; ОБЯЗАТЕЛЬНО при Командная+)

Координация нескольких пар Супервайзер+AI.

Участники	Менеджер потока (ведёт), Супервайзеры
Частота	Регулярная каденция (рекомендуется ежедневно или каждые 2–3 дня)

Повестка: статус по каждой Паре, обновления зависимостей, изменения общих компонентов, проблемы интеграции.

7.6 Обзор поставки (РЕКОМЕНДУЕТСЯ)

Демонстрация работающего ПО заинтересованным сторонам, сбор обратной связи, получение подтверждения приёмки.

Участники	Контекстный архитектор (ведёт), Супервайзер, Заинтересованная сторона
Частота	По вехам поставки

7.7 Ретроспектива инкремента (ОБЯЗАТЕЛЬНО)

Количественный обзор: план vs факт по стоимости, тренд пропускной способности, FPSR, DER, ADR, темп фиксации знаний.

Участники	Менеджер потока (ведёт)
Частота	По окончании каждого Инкремента

Результат: конкретные, измеримые, ограниченные по срокам действия по улучшению, закреплённые за ответственными.

Организациям РЕКОМЕНДУЕТСЯ выделять время на инновации, обучение и эксперименты с AI-инструментарием.

7.8 Сводка

Церемония	Обязательность	Частота
Планирование инкремента	ОБЯЗАТЕЛЬНО	За Инкремент
Начало сессии	ОБЯЗАТЕЛЬНО	За Сессию
Окончание сессии	ОБЯЗАТЕЛЬНО	За Сессию
Обзор качества	ОБЯЗАТЕЛЬНО	Периодически (определяется организацией)
Синхронизация федерации	ОБЯЗАТЕЛЬНО (Командная+)	Регулярная каденция
Обзор поставки	РЕКОМЕНДУЕТСЯ	По вехам
Ретроспектива инкремента	ОБЯЗАТЕЛЬНО	За Инкремент

---

8. Шлюзы качества

Автоматизированные точки контроля. Реализуются как код, а не чеклисты. Чеклист проверки AI-вывода см. в Гиде SENAR.

8.1 QG-0: Контекстный шлюз (начало Задачи)

Качество закладывается на входе. QG-0 гарантирует, что у Задачи есть достаточный и хорошо структурированный контекст до начала AI-направляемой работы.

Критерии ОБЯЗАТЕЛЬНЫ: цель определена, критерии приёмки верифицируемы, связь с требованием или историей установлена (ОБЯЗАТЕЛЬНО), тип работы назначен.

Критерии ОБЯЗАТЕЛЬНЫ (Командная+): критерии приёмки являются независимо верифицируемыми (каждый может быть протестирован или измерен отдельно); в критерии приёмки включён хотя бы один негативный сценарий (ошибочный случай, некорректный ввод или граничное условие).

Критерии ОБЯЗАТЕЛЬНЫ (все конфигурации, задачи безопасности): задачи, затрагивающие аутентификацию, пользовательский ввод, хранение данных, платежи или внешние API, ОБЯЗАНЫ декларировать поверхность угроз и включать хотя бы один критерий приёмки по безопасности. См. Основы SENAR (Стартовый шлюз, критерий 5) и Section 8.7.

РЕКОМЕНДУЕТСЯ: план, оценка сложности, выявленные релевантные знания.

NOTE: Начальная конфигурация наследует требования Стартового шлюза из Базовой (границы области, негативный сценарий), хотя выше они перечислены как критерии ОБЯЗАТЕЛЬНО уровня «Командная+». На Начальном уровне QG-0 = проверки Стартового шлюза Базовой + структура QG-0 Стандарта. Критерии «Командная+» (независимо верифицируемые КП, негативный сценарий) уже практикуются на Базовом уровне; Начальная формализует их как критерии шлюза.

8.2 QG-1: Шлюз требований (уровень Истории/ Инкремента)

Гарантирует, что требования определены, утверждены, декомпозированы и обладают достаточным качеством до начала реализации.

Критерии ОБЯЗАТЕЛЬНЫ [Командная+]: Бизнес-требование (БТ) существует и утверждено; декомпозиция до Требований к задаче (ТЗ) выполнена на нужную глубину; все ТЗ имеют верифицируемые критерии приёмки; нет осиротевших требований (без Задач реализации).

Критерии РЕКОМЕНДУЕТСЯ (Командная+: ОБЯЗАТЕЛЬНО): требования удовлетворяют согласованности (нет противоречий между ТЗ в рамках Истории); требования удовлетворяют достаточности (покрыты нормальные, граничные и ошибочные сценарии); нефункциональные требования специфицированы, где применимо (производительность, безопасность, доступность).

Глубина декомпозиции

Контекстный архитектор ОБЯЗАН определять глубину декомпозиции [Командная+] с учётом сложности и регуляторного контекста:

Контекст	Требуемая глубина	Пример
Стандартная функция, Командная конфигурация (Team)	БТ → СТ → ТЗ (3 уровня)	БТ: «Поддержка OAuth» → СТ: «Поток провайдера OAuth» → ТЗ: КП Задачи
Сложная/регулируемая, Корпоративная конфигурация (Enterprise)	БТ → СТ → ТЗ → ТМ (формальная Тестовая модель)	Полная цепочка прослеживаемости для аудиторского соответствия

NOTE: Для простых функций (БТ → ТЗ, 2 уровня) см. Основы SENAR.

Свойства качества требований

Требования, подаваемые на QG-1, ОБЯЗАНЫ быть верифицируемыми (каждое можно протестировать, измерить или продемонстрировать). Требованиям РЕКОМЕНДУЕТСЯ (Командная+: ОБЯЗАТЕЛЬНО) удовлетворять:

a) **Согласованность** — отсутствие противоречий с существующими требованиями того же или более высокого уровня; b) **Достаточность** — набор ТЗ полностью покрывает родительское требование (СТ или БТ); c) **Неизбыточность** — отсутствие дублирующих требований между Историями; d) **Прослеживаемость** — каждое ТЗ прослеживается вверх до БТ; каждое БТ декомпозируется хотя бы в одно ТЗ.

Управление изменениями требований

NOTE: Управление изменениями требований применяется в конфигурации Командная+.

Когда утверждённое БТ или СТ изменяется после начала реализации:

a) изменение ОБЯЗАНО быть задокументировано с обоснованием и указанием лица, утвердившего его; b) анализ воздействия ОБЯЗАН выявить все нижестоящие требования и Задачи, затронутые изменением; c) затронутые Задачи, уже находящиеся в состоянии `done`, ОБЯЗАНЫ быть помечены для повторной верификации — их одобрение QG-2 аннулируется; d) изменённое требование ОБЯЗАНО повторно пройти QG-1 до начала новой реализации; e) Супервайзеры затронутых Задач ОБЯЗАНЫ быть уведомлены об изменении.

Масштабирование: на Командном уровне анализ воздействия ОБЯЗАН поддерживаться инструментарием (связи «родитель/потомок» требований). На Корпоративном уровне изменения требований ОБЯЗАНЫ быть под контролем версий и подчиняться ревью изменений (см. Section 11.3, Требования-как-код).

8.3 QG-2: Шлюз реализации (Задача завершена)

Критерии ОБЯЗАТЕЛЬНЫ: CI проходит, тесты проходят, статический анализ проходит (включая проверку типов, где применимо), нет новых нарушений линтера, критерии приёмки верифицированы Супервайзером, уязвимости безопасности не обнаружены инструментами сканирования.

РЕКОМЕНДУЕТСЯ: покрытие соответствует порогу, нет дублирования, документация обновлена, запись знаний создана при наличии решения/тупикового подхода. Командная+: сгенерированные тесты проверяются на соответствие Тестовой модели (ТМ) — AI-сгенерированные тесты ОБЯЗАНЫ покрывать заявленные критерии приёмки, а не просто достигать покрытия.

РЕКОМЕНДУЕТСЯ (Командная+: ОБЯЗАТЕЛЬНО): манифесты зависимостей, изменённые AI (package.json, requirements.txt, go.mod, Cargo.toml и т.п.), проверяются на галлюцинированные пакеты — несуществующие в официальном реестре или ведущие к неожиданному мейнтейнеру, с опечатками в именах (близкие по написанию к легитимным) и путаницу зависимостей (коллизии внутренних и публичных пространств имён).

8.4 QG-3: Шлюз верификации (слияние — конфигурации Командная+)

Критерии ОБЯЗАТЕЛЬНЫ: приёмочные тесты проходят, сканирование безопасности чистое, регрессий нет, код проверен (по уровню риска — см. 8.7).

РЕКОМЕНДУЕТСЯ: производительность в рамках SLA, соответствие требованиям доступности, межсервисная интеграция верифицирована.

Минимальные критерии проверки AI-вывода

На QG-3 ОБЯЗАНЫ выполняться следующие AI-специфичные проверки:

a) сгенерированный код не вызывает несуществующие API, методы или флаги CLI (проверка на галлюцинации); b) все импортируемые пакеты существуют в манифесте зависимостей проекта (валидация зависимостей); c) сгенерированный код следует архитектурным паттернам и конвенциям проекта (соответствие паттернам); d) сгенерированные тесты покрывают заявленные критерии приёмки, а не просто достигают покрытия (валидность тестов); e) в сгенерированном коде отсутствуют захардкоженные учётные данные, API-ключи или секреты (проверка секретов).

Эти критерии заменяют зависимость от информационного Чеклиста проверки AI-вывода в Гиде SENAR нормативными минимальными требованиями.

8.5 QG-4: Шлюз приёмки (релиз)

Критерии ОБЯЗАТЕЛЬНЫ: проведён Обзор поставки ИЛИ зафиксировано одобрение заинтересованной стороны; QG-3 по-прежнему проходит; стейджинг-окружение (среда, эквивалентная продакшену) верифицировано; приёмка заинтересованной стороной зафиксирована.

8.6 Правила применения

a) шлюзы ОБЯЗАНЫ быть автоматизированы везде, где возможно; критерии, требующие человеческого суждения, требуют явного зафиксированного одобрения; b) каждое прохождение шлюза ОБЯЗАНО порождать аудиторскую запись; c) шлюзы НЕ ДОЛЖНЫ обходиться без задокументированного Обхода шлюза (обоснование + риск + план устранения + одобрение старшего); d) организациям РЕКОМЕНДУЕТСЯ отслеживать частоту Обходов шлюза.

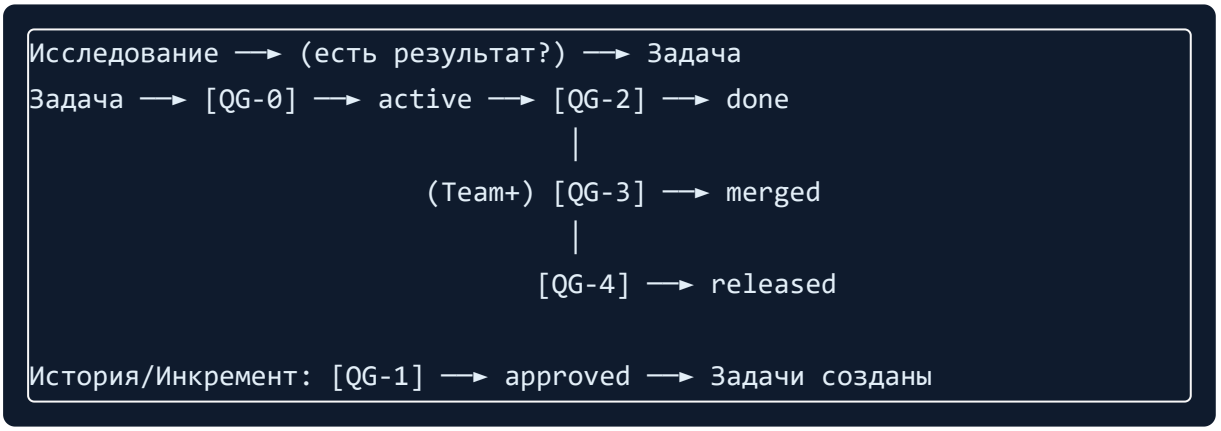
8.7 Проверка на основе рисков

Уровень риска	Примеры	Проверка

Высокий	Безопасность, аутентификация, платежи, миграция данных, архитектура	ОБЯЗАТЕЛЬНО: рецензирование + проверка безопасности (все конфигурации)
Стандартный	Функциональность, UI, бизнес-логика	Автоматизация QG-2 + заявление Супервайзера о верификации
Низкий	Документация, конфигурация, тривиальные исправления	Достаточно автоматизации QG-2

Проверка безопасности для изменений высокого риска: для изменений, затрагивающих аутентификацию, обработку платежей, персональные данные или криптографию, проверка безопасности ОБЯЗАНА выполняться на ВСЕХ уровнях конфигурации, а не только Корпоративная. Это обязательное (ОБЯЗАТЕЛЬНО) требование вне зависимости от размера команды. Организации, разрабатывающие системы, критичные с точки зрения безопасности, ОБЯЗАНЫ внедрять контроли безопасности (QG-3 с Минимальными критериями проверки AI-вывода, проверка кода по уровню риска) вне зависимости от конфигурации.

8.8 Конвейер шлюзов



8.9 Сводка

Шлюз	Применяется	Базовая	Начальная	Командная	Корпоративная
QG-0	Начало Задачи	ДА (Стартовый шлюз)	ДА	ДА	ДА
QG-1	История/ Инкремент	—	—	ДА	ДА

QG-2	Задача завершена	ДА (Шлюз завершения)	ДА	ДА	ДА
QG-3	Слияние	—	—	ДА	ДА
QG-4	Релиз	—	—	ДА	ДА

NOTE: Основы SENAR определяют QG-0 (Контекстный шлюз) и QG-2 (Шлюз реализации) как два шлюза качества. Начальная конфигурация наследует оба шлюза из Базовой (Section 11.1).

8.10 Перекрёстная ссылка по требованиям безопасности

Сводное представление всех требований безопасности по всему Стандарту.
Используйте эту таблицу при аудите покрытия безопасности или при внедрении новой конфигурации.

Требование	Где определено	Конфигурация
Минимальные критерии проверки AI-вывода (проверка секретов, проверка галлюцинаций)	QG-3 (8.4)	Командная+
Проверка изменений высокого риска — требуется одобрение человека (аутентификация, платежи, данные, криптография)	8.7 + Section 10.15 L3	Все конфигурации
Декларация поверхности безопасности при начале задачи	QG-0 (8.1)	Все конфигурации
Верификация покрытия аутентификации	Чеклист проверки AI-вывода, п. 15 (Основы SENAR)	Высокий уровень
Валидация входных данных	Чеклист проверки AI-вывода, п. 6 (Основы SENAR)	Стандартный уровень
Обнаружение секретов в сессиях	Section 6.4	Все конфигурации
Проверка диспетчеризации агентов	Section 5.6 + Section 10.15	Командная+

NOTE: «Все конфигурации» означает, что требование применяется вне зависимости от размера команды — включая команды Начальной конфигурации. Нормативное утверждение о применимости проверки безопасности см. в 8.7.

---

9. Метрики

Организации ОБЯЗАНЫ установить базовые значения, проведя измерения в течение минимум 3 Инкрементов, прежде чем определять целевые показатели. Целевые показатели индивидуальны для организации, а не универсальны.

9.1 Обязательные (ОБЯЗАТЕЛЬНО)

#	Метрика	Формула	Что измеряет
1	**Пропускная способность**	tasks_done / sessions	Производительность поставки
2	**Lead Time**	completed_at - created_at	Скорость поставки
3	**First-Pass Success Rate**	single_cycle_tasks (завершены без доработки) / total × 100%	Качество контекста
4	**Defect Escape Rate**	post_done_defects / done × 100%	Эффективность шлюзов

Дефект после завершения — это баг, явно привязанный к завершённой задаче, которая его породила. Связь ОБЯЗАНА фиксироваться в трекере задач.

9.2 Рекомендуемые (РЕКОМЕНДУЕТСЯ; ОБЯЗАТЕЛЬНО в Командная+)

#	Метрика	Формула	Что измеряет
5	**Knowledge Capture Rate**	entries / tasks	Организационная память
6	**Cost Predictability**	actual_cost / planned_cost × 100%	Точность оценок
7	**Cost per Task**	total_cost / tasks (по сложности)	Эффективность
8	**Manual Intervention**	manual_tasks / total × 100%	Приверженность AI-first

	Rate		
9	**Cycle Time**	completed_at - started_at	Скорость выполнения (vs Lead Time, включающий время ожидания)
10	**Adversarial Detection Rate**	adversarial_critical_findings / L3_reviewed_tasks	Плотность скрытых дефектов

Manual Intervention Rate требует самоотчёта Супервайзера — флаг на задаче, указывающий, что код был написан вручную. Организациям РЕКОМЕНДУЕТСЯ определить, что считать «ручным вмешательством»: любой продакшен-код, написанный вручную, или только задачи, полностью выполненные без AI.

Adversarial Detection Rate (ADR) показывает, сколько критических находок приходится на одну задачу, прошедшую состязательное ревью L3 (Section 10.15). Целевой показатель: организациям РЕКОМЕНДУЕТСЯ стремиться к ADR < 0.5 (менее одной CRITICAL-находки на две проверенные задачи). ADR, равный 0, может означать как отличное качество вывода AI, так и поверхностное ревью — организациям РЕКОМЕНДУЕТСЯ уметь различать эти два случая.

Для вычисления ADR «L3_reviewed_tasks» означает задачи, прошедшие Состязательное ревью L3. Задачи, не получившие ревью L3 (например, задачи низкого риска, где L3 было пропущено согласно Section 10.15), исключаются из знаменателя. Это обеспечивает, что ADR отражает эффективность ревью, а не охват ревью.

Калибровка целевого показателя Knowledge Capture Rate: по мере накопления базы знаний темп появления новых записей закономерно снижается. Организации с устоявшимися базами РЕКОМЕНДУЕТСЯ устанавливать целевые показатели KCR с учётом этого эффекта. Показатель 1.0 (одна запись на задачу) уместен для новых проектов. Для зрелых проектов (>500 задач) показатель 0.33 (одна запись на три задачи) лучше соответствует реальному темпу обнаружения новых знаний. Организации ОБЯЗАНЫ документировать обоснование своего целевого показателя KCR.

9.3 Сбор

Метрики 1–5 и 9 ОБЯЗАНЫ автоматически формироваться из данных трекера задач и журнала сессий. Метрики 6–7 требуют интеграции с системой учёта затрат. Метрика 8 требует самоотчёта Супервайзера. Cost Predictability требует оценки Менеджера потока на Ретроспективе инкремента.

Метрика 10 (ADR) ОБЯЗАНА собираться из находок состязательного ревью, зафиксированных при ревью L3 (Section 10.15). Организации ОБЯЗАНЫ вести запись

находок ревью по задачам, классифицированных по уровню серьёзности, для вычисления ADR.

При вычислении метрик ОБЯЗАНЫ учитываться все задачи — без отбрасывания данных за определённые периоды и без фильтрации исторических записей. Организации НЕ ДОЛЖНЫ исключать задачи из расчёта метрик на основании даты создания, статуса миграции или версии инструментария. Обоснование: фильтры по периодам усложняют систему и затрудняют поддержку, а также маскируют проблемы качества данных. По мере накопления новых данных влияние старых записей уменьшается, и метрики постепенно приближаются к актуальным значениям. Если ранние данные заведомо ненадёжны (например, ручные записи до автоматизации), это ОБЯЗАНО быть задокументировано как известное ограничение, а не отфильтровано.

Cost Predictability (метрика 6) требует фиксации запланированной стоимости до начала реализации. На практике оценка стоимости задач с участием AI ненадёжна: время выполнения недетерминировано, ценообразование моделей варьируется. В Командной конфигурации (Team) Cost Predictability ОБЯЗАН отслеживаться, но в первые 3 Инкремента базовые значения могут быть предварительными. В Корпоративной конфигурации (Enterprise) Cost Predictability ОБЯЗАН отслеживаться с установленными базовыми значениями. Cost per Task (метрика 7) является более практичной альтернативой для управления стоимостью на ранних этапах внедрения.

Работа с несколькими агентами и параллельными сессиями: когда несколько AI-агентов работают одновременно, метрики на уровне сессии (пропускная способность, длительность, стоимость) отражают результативность отдельного агента, а не совокупный выход команды. Организации с мультиагентными конфигурациями РЕКОМЕНДУЕТСЯ дополнительно отслеживать агрегированные метрики на уровне Инкремента. Токены и стоимость в мультиагентных сценариях РЕКОМЕНДУЕТСЯ учитывать по каждому агенту, а сводные итоги — быть доступны для отчётности на уровне Инкремента.

---

10. Операционные правила

10.1 Задача до реализации (ОБЯЗАТЕЛЬНО)

Реализация НЕ ДОЛЖНА начинаться без Задачи. Исследования освобождены; если они порождают код, Задача ОБЯЗАНА быть создана до коммита.

10.2 Длительность сессии (ОБЯЗАТЕЛЬНО)

Организации ОБЯЗАНЫ установить, задокументировать и контролировать максимальную длительность Сессии. Обоснование РЕКОМЕНДУЕТСЯ задокументировать. Максимум РЕКОМЕНДУЕТСЯ основывать на эмпирических данные сессий. Ориентир: сессии длиннее 180 минут дают убывающую отдачу в большинстве рабочих процессов с AI.

10.3 Каденция контрольных точек (ОБЯЗАТЕЛЬНО)

Контрольные точки ОБЯЗАНЫ выполняться с интервалами, не превышающими задокументированный организацией лимит. Интервал РЕКОМЕНДУЕТСЯ устанавливать не превышающим допустимый объём потери контекста при внезапном прерывании сессии.

10.4 Документирование тупиковых подходов (ОБЯЗАТЕЛЬНО)

Неудачные подходы ОБЯЗАНЫ фиксироваться как записи знаний типа «Тупиковый подход» (подход, причина, альтернатива). Тупиковым считается исследование, занявшее более 15 минут и не давшее пригодного результата. При достижении этого порога Супервайзер ОБЯЗАН остановиться, зафиксировать подход и причину неудачи и перейти к альтернативному пути.

10.5 Периодический аудит (ОБЯЗАТЕЛЬНО)

Обзоры качества ОБЯЗАНЫ проводиться с каденцией, определённой организацией, и пересматриваться на каждой Ретроспективе. Каденцию РЕКОМЕНДУЕТСЯ устанавливать не реже одного раза за 3 Инкремента. Для Начальной конфигурации рекомендуемая каденция — ежемесячно (Section 11.1), что покрывает данное требование при Инкрементах длительностью до 2 недель.

10.6 Контроль версий (ОБЯЗАТЕЛЬНО)

a) коммиты ОБЯЗАНЫ быть атомарными; b) обнаружение секретов ОБЯЗАНО быть автоматизировано; c) изменения AI ОБЯЗАНЫ проверяться на расползание области.

10.7 Лимит параллельных агентов (ОБЯЗАТЕЛЬНО)

Супервайзеры НЕ ДОЛЖНЫ превышать установленный организацией лимит параллельных агентов. Типичный стартовый лимит — 3 одновременных агента на Супервайзера; организациям РЕКОМЕНДУЕТСЯ калибровать его с учётом сложности задач и ресурсов Супервайзера.

10.8 Калибровка сложность-стоимость (ОБЯЗАТЕЛЬНО)

Организации ОБЯЗАНЫ поддерживать базовые значения стоимости по уровням сложности задач.

10.9 Фиксация знаний (ОБЯЗАТЕЛЬНО)

Организации ОБЯЗАНЫ установить и отслеживать целевой показатель Knowledge Capture Rate.

10.10 Прослеживаемость требований (ОБЯЗАТЕЛЬНО Командная+)

Организации ОБЯЗАНЫ поддерживать прослеживаемость от Бизнес-требований через Системные требования до Требований к задачам. Каждая Задача ОБЯЗАНА ссылаться хотя бы на одно требование или Историю. Требования ОБЯЗАНЫ храниться в версионируемой, поисковой системе.

10.11 Документация кода как контекст (ОБЯЗАТЕЛЬНО)

Документация кода (docstrings, doc-комментарии, описания API, архитектурная документация) — не артефакт «постфактум», а активный контекст для AI-агентов. Хорошо документированный код снижает объём контекста, который Супервайзер предоставляет на каждую Задачу, и позволяет AI генерировать корректный вывод с меньшим количеством явных инструкций.

Организации ОБЯЗАНЫ поддерживать документацию кода: (a) назначение модуля/пакета, (b) контракты публичного API с типами параметров и возвращаемых значений, (c) архитектурные границы с зависимостями и точками интеграции. Документация ОБЯЗАНА быть машиночитаемой (структурированные комментарии, форматы описания API и т.п.). Документацию РЕКОМЕНДУЕТСЯ обновлять в рамках той же Задачи, которая изменяет код (критерий QG-2: «документация обновлена при изменении API»).

В AI-нативной разработке у документации двойная аудитория: Супервайзеры и AI-агенты. Документация, полезная только людям (например, «уточните у Ивана»), бесполезна для AI. Документация ОБЯЗАНА быть самодостаточной и машиночитаемой.

10.12 Гигиена контекста (ОБЯЗАТЕЛЬНО)

Супервайзеры НЕ ДОЛЖНЫ включать в контекст AI-агента учётные данные, API-ключи, персональные данные (PII) или регулируемые данные (финансовые, медицинские) — за исключением случаев, когда у провайдера модели AI есть действующее соглашение об обработке данных и классификация данных это допускает. Организации ОБЯЗАНЫ вести политику классификации контекста, определяющую запрещённые категории данных. Заполнители или синтетические данные РЕКОМЕНДУЕТСЯ использовать, когда помощь AI нужна для задач, связанных с чувствительными доменами.

10.13 Управление моделью AI (ОБЯЗАТЕЛЬНО)

Провайдеры моделей AI — внешние поставщики. Модель AI — основной производственный инструмент, эквивалентный компилятору. Организации ОБЯЗАНЫ:

a) фиксировать идентификатор и версию модели AI, использованной для каждой Сессии; b) рекалибровать базовые значения метрик (FPSR, стоимость на задачу, пропускная способность) при существенном изменении активной версии модели — смена поколения модели является существенным изменением; c) в Командная+: оценивать новые версии модели до внедрения — сравнивать FPSR и стоимость на задачу на репрезентативном наборе задач; d) в Корпоративной: поддерживать формальный процесс утверждения модели — новые модели ОБЯЗАНЫ тестироваться на

критерии приёмки для предметной области организации до продуктивного использования; e) документировать известные ограничения модели и паттерны галлюцинаций как записи знаний.

NOTE: Положения SENAR, опирающиеся на поведение AI (эвристики обнаружения галлюцинаций в Гиде, рекомендации по длительности сессий, лимиты параллельных агентов), зависят от возможностей модели. Организациям РЕКОМЕНДУЕТСЯ пересматривать их при существенной смене поколений модели.

10.14 Управление изменениями скриптов (ОБЯЗАТЕЛЬНО)

Изменения операционных скриптов ОБЯЗАНЫ рассматриваться как изменения производственного процесса:

- под контролем версий (Section 10.6)
- проверяются перед развёртыванием
- решение фиксируется в базе знаний (применяется Section 10.9)

Конфигурации Командная+ ОБЯЗАНЫ дополнительно:

- тестировать изменения скриптов на изолированном окружении перед распространением
- поддерживать реестр версий активных скриптов по проектам
- обеспечивать возможность отката для любого изменения скрипта

Полное определение Операционных скриптов и их управление см. в Section 5.3.

10.15 Верификация качества AI-вывода (ОБЯЗАТЕЛЬНО)

AI-сгенерированный код ОБЯЗАН проходить верификацию качества перед коммитом. Определены три уровня верификации:

Уровень	Метод	Конфигурации
L1: Автоматизированный	Статический анализ — размер файла, размер функции, метрики сложности, проверки паттернов безопасности, линтер	Все (ОБЯЗАТЕЛЬНО)
L2: Заявление о верификации	Лицо или агент, выполнивший работу, пишет структурированное заявление о верификации, подтверждающее, что было	Все (ОБЯЗАТЕЛЬНО)

	проверено по критериям приёмки и Стандартам кода	
L3: Состязательное ревью	Независимый агент (другая модель или без контекста) проверяет без доступа к ходу рассуждений генерирующего агента	ОБЯЗАТЕЛЬНО (Командная+)

Требования к Состязательному ревью L3: a) проверяющий агент НЕ ДОЛЖЕН иметь доступа к контексту сессии или ходу рассуждений генерирующего агента; b) хотя бы одному проверяющему РЕКОМЕНДУЕТСЯ быть «холодным» ревьюером — агентом с нулевым предварительным контекстом задачи; c) находки ОБЯЗАНЫ классифицироваться по серьёзности (CRITICAL, HIGH, MEDIUM) и фиксироваться; d) находки CRITICAL ОБЯЗАНЫ блокировать коммит до устранения; e) организации ОБЯЗАНЫ отслеживать Adversarial Detection Rate (ADR) — см. Section 9.

Ревью L3 ОБЯЗАНО применяться на основе уровня риска (см. Section 8.7):

- **Высокий риск** (безопасность, аутентификация, платежи, миграция данных, архитектура): для изменений высокого риска (Section 8.7) ревью L3 ОБЯЗАНО применяться вне зависимости от уровня конфигурации; хотя бы один проверяющий ОБЯЗАН быть Супервайзером-человеком, а не AI-агентом; ОБЯЗАНА выполняться проверка безопасности (см. Section 8.7);
- **Стандартный риск** (функциональность, UI, бизнес-логика): L3 ОБЯЗАНО применяться (Командная+);
- **Низкий риск** (документация, конфигурация, тривиальные исправления): L3 МОЖЕТ быть пропущено с задокументированным обоснованием.

NOTE: Заявление L2 покрывает дефекты качества (логические ошибки, нарушения стиля, покрытие критериев приёмки), но НЕ является контролем безопасности — для этого минимальный эффективный уровень верификации — L3. Заявление L2 — структурированная запись о том, что было проверено; это не ревью независимой стороной. Исполнитель (человек или агент) фиксирует, какие критерии приёмки верифицированы и каким способом.

NOTE: Скрытые дефекты в AI-сгенерированном коде особенно трудно обнаружить — они проходят автоматические проверки, но содержат тонкие логические, архитектурные или проблемы безопасности. Типичные паттерны: возврат True без валидации, обход через сравнение с null, доверие HTTP-заголовкам без проверки, пустые конфигурационные значения, молча отключающие защиту, и недостижимый «защитный» код, маскирующий неполное понимание потока управления.

Диспетчеризация агентов (делегирование подагенту) — сценарий с наивысшим риском скрытых дефектов: делегированный агент работает с ограниченным контекстом.

Организации, использующие диспетчеризацию, ОБЯЗАНЫ применять ревью L3 ко всем результатам делегированных агентов.

10.16 Сводка

#	Правило	Командная	Корпоративная
1	Задача до реализации	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
2	Длительность сессии	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
3	Каденция контрольных точек	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
4	Документирование тупиковых подходов (порог >15 мин)	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
5	Периодический аудит	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
6	Контроль версий	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
7	Лимит параллельных агентов	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
8	Калибровка сложность-стоимость	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
9	Фиксация знаний	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
10	Прослеживаемость требований	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
11	Документация кода как контекст	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
12	Гигиена контекста	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
13	Управление моделью AI	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
14	Управление изменениями скриптов	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
15	Верификация качества AI-вывода	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО

NOTE: Для начального уровня внедрения см. Основы SENAR, определяющие 8 базовых правил.

---

11. Конфигурации

NOTE: Для индивидуального внедрения см. Основы SENAR — 8 правил, 2 шлюза качества, 2 метрики. Стандарт определяет три организационные конфигурации: Начальная (Foundation), Командная (Team) и Корпоративная (Enterprise).

11.1 SENAR Начальная (Foundation) (1–3 Пары)

Мост между Базовой и Командной. Для небольших команд, которым нужно больше структуры, чем даёт Базовая, но которые не готовы к полному процессу Командной конфигурации.

Категория	Требования
База	Все правила Основ SENAR применяются
Обязанности	3 совмещённых набора обязанностей, закрытых 2 людьми (Section 4.8): Супервайзер + Контекстный архитектор (совмещены), Инженер знаний + Инженер верификации (совмещены). Обязанности Менеджера потока берёт на себя Супервайзер.
Церемонии	3: Начало сессии, Окончание сессии, Обзор качества (ежемесячно). NOTE: Начальная проводит Инкременты (Section 6.5), но опускает формальные церемонии Планирования инкремента и Ретроспективы. Планирование неявно включено в Начало сессии; ретроспективные выводы фиксируются в Обзоре качества.
Шлюзы качества	2: QG-0 (Контекстный шлюз) + QG-2 (Шлюз реализации)
Метрики	4: Пропускная способность, Lead Time, FPSR, Defect Escape Rate
Правила	Базовая 8 + Правило 2 (Standard 10.2, Длительность сессии), Правило 4 (Standard 10.4, Документирование тупиковых подходов), Правило 9 (Standard 10.9, Фиксация знаний) = 11 правил
База знаний	Общая, доступная всем членам команды и AI-сессиям

> ***Примечание:*** *На Начальном уровне мониторинг длительности сессий (Правило 10.2) является обязанностью Супервайзера, поскольку Менеджер потока не является отдельной ролью в данной конфигурации (Section 4.8).*

> ***Примечание:*** *Правила Базовой конфигурации и Стандарта используют различную нумерацию. Правило Базовой конфигурации 8 (Фиксация знаний) расширяется в Правила Стандарта 10.4 (Документирование тупиковых подходов) и 10.9 (Фиксация знаний). Начальная добавляет их как явные отслеживаемые требования наряду с Правилом Стандарта 10.2 (Длительность сессии).*

> ***Примечание:*** *Section 6.4 (обнаружение секретов в сессиях) неявно требует элементов Правила 10.6 (Контроль версий). Организациям, внедряющим Начальную конфигурацию, РЕКОМЕНДУЕТСЯ также внедрить Правило 10.6 для обеспечения атомарных коммитов, автоматизированного обнаружения секретов и проверки расползания области AI.*

Путь внедрения (по неделям)

- **Неделя 1 — привычки Базовой конфигурации + Начало/Окончание сессии.** Вся команда читает Основы SENAR и начинает ежедневно применять 8 правил. Вводятся Начало сессии (загрузка контекста + объявление цели) и Окончание сессии (резюме результатов + запись знаний) как командные церемонии. Назначаются 3 совмещённых набора обязанностей — роли могут пересекаться.
- **Неделя 2 — документирование тупиковых подходов + фиксация знаний.** Активируется Правило 4 (Документирование тупиковых подходов): каждый заблокированный путь получает запись знаний до смены направления. Активируется Правило 9 (Фиксация знаний): каждое значимое решение и неочевидная находка заносятся в общую БЗ. Через две недели у команды — работающая база знаний, а не пустая.
- **Неделя 3 — Обзор качества (первый ежемесячный).** Первый Обзор качества: обзор журнала сессий, проверка задокументированных тупиковых подходов, выявление повторяющихся проблем. Это ретроспектива с результатом — конкретные корректировки порогов, длительности сессий или глубины декомпозиции задач.
- **Неделя 4 — базовые значения метрик.** Начинается измерение 4 метрик: Пропускная способность (задач за неделю), Lead Time (от создания задачи до завершения), FPSR (доля успеха с первой попытки на QG-2), Defect Escape Rate (дефекты, обнаруженные после завершения). Фиксируются базовые значения — пока без целей, только калибровка. Пороги адаптируются к предметной области.

Начальная vs Базовая — что отличается

Базовая — индивидуальная дисциплина: один Супервайзер, два шлюза качества, две метрики — всё на личном уровне. Начальная добавляет **командную координацию**:

- Общая база знаний, видимая всем членам команды и AI-сессиям (не только локальный контекст автора)
- Начало/Окончание сессии как **командные церемонии** — точки синхронизации, а не просто индивидуальные привычки
- Ежемесячный Обзор качества — коллективный обзор здоровья процесса, а не индивидуальная рефлексия
- 2 дополнительные метрики (FPSR + DER), которые обретают смысл только при наличии общего определения «готово» у нескольких людей

Когнитивная нагрузка невелика — Начальная рассчитана на постепенное внедрение поверх уже сложившихся привычек Базовой конфигурации.

FAQ для команд Начальной конфигурации

Когда переходить на Командную? Когда выполнено любое условие: команда доросла до 3+ Пар и координационная нагрузка стала заметной; необходимо межпарное отслеживание зависимостей или федерация знаний; нужен QG-1 (Шлюз требований) или QG-3 (Шлюз верификации); требуются выделенные (непересекающиеся) обязанности.

Можно ли пропустить Начальную и сразу перейти на Командную? Да. Если у команды уже есть процессная дисциплина — структурированная декомпозиция задач, культура code review, ретроспективы — начинайте с Командной. Начальная — мост, а не обязательная остановка.

Что, если у нас ровно 3 пары? Оцените уровень опыта. Если все опытны в структурированной AI-разработке — переходите на Командную, накладные расходы управляемы. Если опыт в команде разный или команда только начинает работать с AI-процессами — оставайтесь на Начальной 4–8 недель для формирования привычек, прежде чем добавлять федерацию и все 5 шлюзов.

11.2 SENAR Командная (Team) (3–10 Пар)

Категория	Требования
Обязанности	Все 5 выделенных (ОБЯЗАТЕЛЬНО)
Церемонии	Все 7 (согласно Section 7)
Шлюзы качества	Все 5: QG-0 — QG-4 (ОБЯЗАТЕЛЬНО)
Метрики	Все 10 (ОБЯЗАТЕЛЬНО)

Правила	Все 15 (ОБЯЗАТЕЛЬНО)
Федерация	Межпарное отслеживание зависимостей, общая БЗ (ОБЯЗАТЕЛЬНО)
Проверка на основе рисков	QG-3 дифференцирован по риску (Section 8.7)
Проверка безопасности	ОБЯЗАТЕЛЬНО для изменений высокого риска (аутентификация, платежи, данные, криптография) — см. Section 8.7

11.3 SENAR Корпоративная (Enterprise) (10+ Пар)

Все требования Командной, ПЛЮС:

Категория	Дополнительно
Управление портфелем	Инкременты группируются по потокам создания ценности с единым бюджетом
Дополнительные обязанности	Портфельный менеджер, Главный Супервайзер, Координатор федерации
Соответствие	Аудиторские следы шлюзов для требований ISO/регуляторов
Управление	Ревью Обходов шлюзов, архитектурные исключения, контроль бюджета
Требования-как-код	Требования хранятся в VCS, CI валидирует прослеживаемость, ревью изменений обязательно (ОБЯЗАТЕЛЬНО)

Коэффициенты масштабирования и руководство для Корпоративной конфигурации см. в Справочнике SENAR.

NOTE: Корпоративная конфигурация описывает целевое состояние для масштабного внедрения. Организациям такого масштаба следует проверять коэффициенты и адаптировать механизмы координации под свой контекст.

11.4 Сравнение

Элемент	Базовая	Начальная	Командная	Корпоративная
Пары	1	1–3	3–10	10+
Правила	8	11	15	15

Шлюзы качества	2	2 (QG-0, QG-2)	5 (QG-0..QG-4)	5 + соответствие
Метрики	2	4	10	10 + портфель
Обязанности	1 (Супервайзер)	3 (совмещены)	5 (выделены)	5 + портфель
Церемонии	0	3	7	7 + портфель
База знаний	Рекомендуется	Обязательна	Обязательна + федерация	Обязательна + федерация
Проверка безопасности	Чеклист	Чеклист	ОБЯЗАТЕЛЬНО (высокий риск)	ОБЯЗАТЕЛЬНО + формальный аудит
Инструментарий	Нет	Рекомендуется	Обязателен	Обязателен

11.5 Миграция

```
Основы SENAR ──► Foundation ──► Team ──► Enterprise
 + 3 роли (совмещены)   + выделенные роли     + управление портфелем
 + 3 церемонии          + федерация            + потоки создания ценности
 + 2 доп. метрики       + все шлюзы            + соответствие
 + управление сессиями  + все метрики/правила  + управление
```

Каждый шаг инкрементален — пилотируйте на части Пар. Типичные сроки:

- Базовая → Начальная: 2–4 недели после усвоения командой привычек Базовой конфигурации
- Начальная → Командная: 2–3 месяца, когда команда достигает 3+ Пар или нуждается в федерации
- Командная → Корпоративная: когда организация имеет 10+ Пар и нуждается в портфельном управлении

---

12. Модель зрелости

12.1 Уровни

Уровень 1: Стихийный

AI используется от случая к случаю. Нет задач, шлюзов, метрик, задокументированных знаний.

Уровень 2: Супервизируемый (соответствует Основам SENAR)

Задачи создаются до работы. Вывод AI верифицируется. QG-0 и QG-2 работают. 2 метрики отслеживаются (FPSR + Пропускная способность). Тупиковые подходы документируются (порог >15 мин). Сессии имеют начало/окончание с передачей контекста.

Ключевой индикатор: каждая реализация с участием AI имеет Задачу; вывод верифицируется.

Уровень 3: Измеримый (соответствует Командной (Team))

Все 5 наборов обязанностей закрыты. Все 5 шлюзов автоматизированы. Все 10 метрик с базовыми значениями и целями (включая ADR, отслеживаемый по задачам). Фиксация знаний структурирована. Федерация активна. Architectural Decision Records (ADR) ведутся. Решения принимаются на основе данных.

Ключевой индикатор: базовые значения установлены; метрики пересматриваются на каждой Ретроспективе.

> ***Примечание:*** *Уровни 4 и 5 — перспективные цели, основанные на отраслевых паттернах (CMMI, DORA). Ни одна реализация SENAR не была независимо валидирована на этих уровнях. Они приводятся как ориентиры для организаций, стремящихся к непрерывному совершенствованию.*

Уровень 4: Управляемый (перспективный)

Предсказуемость стоимости стабильна. FPSR растёт. Defect Escape Rate ниже порога. База знаний активно повышает качество контекста AI. Процессные улучшения ставятся

как эксперименты и измеряются.

Уровень 5: Оптимизирующий (перспективный)

Организация вырабатывает собственные паттерны. Потоки создания ценности оптимизированы. Процессные эксперименты — норма. База знаний — конкурентное преимущество.

NOTE: Уровни 4–5 являются перспективными. Подробности см. в примечании перед Уровнем 4.

12.2 Измерения оценки

Организации МОГУТ находиться на разных уровнях по различным измерениям:

Измерение	L1	L2	L3
Дисциплина задач	Отсутствует	Задачи до работы	Полный жизненный цикл + прослеживаемость
Шлюзы качества	Отсутствуют	QG-0 + QG-2	Все 5 автоматизированы
Метрики	Отсутствуют	2 (Базовая) (Core)	4 обязательных + 6 рекомендуемых
Знания	Не документируются	Тупиковые подходы фиксируются	Структурированы + целевой показатель
Верификация	Отсутствует	Супервайзер самоверифицирует	Независимая верификация, состязательное ревью и отслеживание ADR
Стоимость	Неизвестна	Отслеживается	Базовые значения установлены и прогнозируются
Требования	Не документируются	Цель задачи + КП	Полная иерархия (БТ→СТ→ТЗ) + свойства качества

12.3 Прогрессия

Организациям РЕКОМЕНДУЕТСЯ двигаться последовательно — каждый уровень строится на предыдущем. Организации МОГУТ в первую очередь подтягивать наиболее слабые измерения.

---

13. Соответствие

13.1 Заявление о соответствии

Для заявления о соответствии SENAR организация ОБЯЗАНА:

a) определить применимую конфигурацию: Начальная, Командная или Корпоративная (Section 11); b) реализовать все требования уровня ОБЯЗАТЕЛЬНО применимой конфигурации; c) задокументировать все нереализованные требования уровня РЕКОМЕНДУЕТСЯ с обоснованием; d) хранить свидетельства реализации (аудиторские записи, данные метрик, записи трекера задач).

Заявление о соответствии ОБЯЗАНО использовать следующий формат: «[Организация] соответствует SENAR v[версия], конфигурация [Configuration], [самодекларация | независимая оценка коллегами | независимый аудит], по состоянию на [дата].»

13.2 Уровни соответствия

Уровень	Описание	Свидетельства
Самодекларация	Организация самостоятельно оценивает себя по требованиям SENAR	Запись внутренней оценки
Независимая оценка коллегами	Оценка проводится практиком SENAR из другой организации	Отчёт коллегиальной оценки
Независимый аудит	Оценка квалифицированным аудитором с опытом оценки процессов программной инженерии	Формальный аудиторский отчёт

Организации МОГУТ заявлять любой уровень соответствия. Более высокие уровни дают более весомые свидетельства для заинтересованных сторон, клиентов и регуляторов.

13.3 Основы SENAR

Команды, практикующие Основы SENAR, не обязаны соответствовать данному Стандарту, но им рекомендуется переход на Начальную конфигурацию (Foundation) по мере готовности. Соответствие Основам SENAR заявляется отдельно по документу Основы SENAR.

13.4 Частичное соответствие

Организации МОГУТ заявлять о частичном соответствии, указывая, каким разделам они соответствуют (например, «Соответствие SENAR Командная (Team) для Разделов 6–9»). Заявления о частичном соответствии ОБЯЗАНЫ явно перечислять включённые разделы.

Заявления о частичном соответствии ОБЯЗАНЫ включать как минимум Разделы 6 (Единицы работы), 8 (Шлюзы качества), 9 (Метрики) и 10 (Операционные правила) — они составляют неделимое ядро практики SENAR. Заявление, охватывающее только определительные разделы (Разделы 1–4), не является валидным заявлением о частичном соответствии.

13.5 Обработка несоответствий

Когда обязательное требование невозможно выполнить, организация ОБЯЗАНА задокументировать несоответствие: обоснование, признание рисков, план устранения и одобрение лицом, старшим по отношению к инициатору. Записи о несоответствиях отличаются от операционных Обходов шлюзов (Section 3.13) — они действуют на уровне процесса, а не задачи.

13.6 Поддержание соответствия

Организациям РЕКОМЕНДУЕТСЯ повторно оценивать соответствие на каждой Ретроспективе инкремента (Section 7.7). Заявления о соответствии РЕКОМЕНДУЕТСЯ пересматривать после существенных изменений процессов, смены поколения модели AI (Section 10.13) или перехода между уровнями конфигураций.

Быстрый старт SENAR: 5 минут до надёжной AI-разработки

Вы используете AI для написания кода. SENAR делает результат надёжным.

Никаких совещаний. Никаких обязательных сертификаций. Эти привычки — **SENAR Core**: 8 правил, 2 шлюза качества, 2 метрики — всё, что нужно для старта. Затраты на внедрение: менее 1 часа, около 5 минут за сессию после этого.

---

6 привычек

ПЕРЕД ТЕМ как дать AI задание

1. Запишите ЧТО и КОГДА ГОТОВО

Перед началом запишите две вещи:

- **Цель** — что должно быть сделано (одно предложение)
- **Критерии приёмки** — как вы поймёте, что готово (нумерованный список, каждый пункт проверяется независимо)

Плохо: «Добавить функциональность логина» Хорошо: «Реализовать вход по email/паролю. КП: 1. POST /auth/login возвращает JWT при валидных данных. 2. Возвращает 401 при неверном пароле. 3. Возвращает 422 при отсутствии поля email. 4. Токен истекает через 24 часа.»

Почему это важно: Качество вывода AI равно качеству входных данных. Размытая цель порождает правдоподобный код, который падает в продакшене. Точная цель с чёткими критериями приёмки даёт тестируемый и корректный код с первой попытки.

2. Установите границы

Скажите AI, что НЕ ТРОГАТЬ:

- «Изменяй ТОЛЬКО директорию users/»
- «НЕ модифицируй схему базы данных»
- «Следуй паттернам из auth/router.py»

Без границ AI уверенно отрефакторит половину кодовой базы, чтобы «улучшить» то, о чём вы не просили.

ПОКА AI работает

3. Проверяйте по критериям, а не по ощущениям

Не просто бегло взгляните на код, думая «выглядит нормально». Проверьте каждый критерий приёмки:

- КП 1: POST /auth/login возвращает JWT? → Проверьте тест или запустите его.
- КП 2: Возвращает 401 при неверном пароле? → Проверьте тест.
- КП 3: Нет email → 422? → Проверьте тест.

Если для критерия нет теста — критерий не проверен.

4. Документируйте тупиковые подходы

Когда подход не сработал, запишите одно предложение — почему:

- «Попробовал bcrypt для хеширования паролей — импорт не работает на Python 3.14, переключился на argon2»
- «SQLAlchemy async session: нельзя использовать lazy loading, нужен selectinload»

Это занимает 10 секунд и экономит часы — вам на следующей неделе, вашему коллеге, любому AI, который читает вашу базу знаний.

ПОСЛЕ того как готово

5. Запустите тесты

Если тесты проходят И все критерии приёмки выполнены → готово. Если нет → не готово. Без исключений, без «наверное, работает».

6. Зафиксируйте полученные знания

Если вы обнаружили что-то неочевидное в ходе задачи, запишите:

- Решение, которое вы приняли, и почему
- Паттерн, который хорошо сработал
- Подводный камень, который вас удивил

> ***Примечание — 6 привычек vs. 8 правил Базовой конфигурации:*** *Эти 6 привычек охватывают суть 8 правил SENAR Core. Полный Core добавляет две явные практики: многоуровневый чеклист верификации для латентных дефектов (Правило 5) и анализ первопричин перед устранением симптомов (Правило 7). Если*

> *вы работаете в регулируемом контексте или в условиях высоких рисков, прочитайте полный документ Core, прежде чем полагаться только на этот Быстрый старт.*

До и После: реальные данные

Эти цифры получены на одном проекте (6 микросервисов, 552 задачи, $989 затрат на AI, 38 сессий). Они иллюстративны, не универсальны — ваши цифры будут другими. SENAR требует устанавливать собственные базовые показатели, прежде чем ставить целевые.

Без структурированного процесса (сессии 1–3, ad hoc):

- Задачи без критериев приёмки → AI выдавал код, который «выглядел правильно», но не проходил крайние случаи
- Тупиковые подходы не документировались → одни и те же ошибочные подходы повторялись между сессиями
- Без дисциплины сессий → марафоны по 200+ минут с заметным снижением эффективности
- Дефекты обнаруживались после «готово» → ориентировочная стоимость переделки ~$105 на пропущенный дефект (оценка на основе данных проекта: средняя стоимость переработки дефектов, обнаруженных после завершения задачи)

С внедрёнными привычками SENAR (сессии 4–38):

- У каждой задачи есть цель + КП перед стартом → First-Pass Success Rate улучшился до 85%+ (задачи корректны с первой попытки)
- Тупиковые подходы документируются → повторные ошибки исключены для задокументированных случаев
- Сессии ограничены 120 минутами с контрольными точками → стабильная производительность, без «вылетов» контекста
- QG-0 блокирует работу без задачи → ноль задач «а для чего это было?»

Оговорка: Это опыт одной команды (N=1), а не контролируемый эксперимент. Улучшение смешивает внедрение методологии с естественным обучением команды. Нужны независимые воспроизведения — именно поэтому мы опубликовали стандарт, чтобы другие тоже могли измерять.

Стоимость SENAR: ~5 минут накладных расходов за сессию + 1–3 минуты на задачу.
Чего вы избегаете: переделки из-за размытых требований, повторение тупиковых подходов, марафонные сессии с падающей продуктивностью.

Вот и всё

Эти привычки напрямую соответствуют правилам **SENAR Core** — самодостаточного подмножества стандарта SENAR. Всё остальное в методологии построено на этом фундаменте.

Что вы получаете:

- Меньше сюрпризов «у меня работает»
- AI чаще выдаёт корректный код с первой попытки (мы это измеряем — это называется First-Pass Success Rate)
- Знания накапливаются, а не испаряются между сессиями
- Вы можете передать работу другому разработчику (или себе будущему) без потери контекста

Что это стоит: ~5 минут накладных расходов за сессию + 1–3 минуты на задачу для целей, критериев и верификации.

Следующие шаги

Вы хотите...	Читайте
Прочитать формальный документ SENAR Core (8 правил, 2 шлюза, 2 метрики)	**SENAR Core**
Посмотреть полный пример задачи от начала до конца	Гид: Разбор примера
Настроить SENAR с вашим AI-инструментом	Гид: Интеграция с инструментами (Claude Code, Cursor, Copilot)
Внедрить SENAR в существующую кодовую базу	Гид: Внедрение в Legacy
Узнать об уровнях требований	Гид: Инженерия требований
Понять философию	Гид: Философия
Перейти с Базовой (Core) на полный стандарт	Гид: Переход — от Базовой (Core) к Standard
Масштабировать на команду	Стандарт, раздел 11: Конфигурации
Оценить текущую практику	Стандарт, раздел 12: Модель зрелости

Гид SENAR: Философия

Ценности SENAR и шесть столпов, на которых строится методология.

Ценности SENAR

1. **Контекст важнее кода** — Качество вывода AI определяется качеством входного контекста. Инвестируйте в требования, а не в скорость кодирования.
2. **Верификация важнее скорости** — AI генерирует с машинной скоростью. Ограничивающий фактор — корректность, а не скорость.
3. **Знания важнее опыта** — У AI нет памяти между сессиями. Что не задокументировано — не существует для AI.
4. **Принуждение важнее договорённости** — Шлюзы качества как автоматизированный код, а не совещания, которые можно пропустить.
5. **Суждение важнее нажатий клавиш** — Внимание человека направлено на решения (что строить, корректно ли это), а не на набор кода.

Столп 1: Контекст прежде всего (качество на входе)

Качество вывода AI — прямая функция качества входного контекста.

Принцип каскада: Качество закладывается на входе, а не проверяется на выходе. Дефект в бизнес-требовании расползается на все системные требования, требования к задачам и в итоге на код. Когда дефектное требование доходит до AI, оно порождает правдоподобный код, который проходит автоматические проверки, но решает не ту задачу. Никакие тесты на выходе (QG-2, QG-3) не поймают требование, которое было ошибочным с самого начала.

Поэтому SENAR вкладывается в качество требований (QG-0, QG-1) до начала реализации — не как бюрократию, а как самую окупаемую инвестицию в качество.

Требования и есть контекст. Хорошо определённое бизнес-требование, разбитое на чёткие системные требования и требования к задачам, — главный вход для качества AI-

генерируемого кода. Иерархия требований (BR → SR → TR) — не бумажная работа, а структурированный контекст, от которого зависит, выдаст ли AI корректный результат.

Компоненты контекста:

- Цель задачи и критерии приёмки (= требования к задаче)
- Связь требований с родительской историей/БТ/СТ (= прослеживаемость)
- Архитектурные ограничения и конвенции
- Релевантные знания (решения, тупиковые подходы, подводные камни)
- Примеры и антипаттерны
- Границы области изменений (что НЕ менять)

Типичные ошибки:

- *Вайб-промптинг* — размытые инструкции без структуры. Допустимо для тривиальных задач, опасно для сложной работы.
- *Свалка контекста* — заваливание AI неструктурированной информацией. Критические ограничения тонут в потоке.
- *Неявное допущение* — ожидание, что AI знает конвенции без документации.
- *Неоднозначные требования* — то, что человек уточнил бы в разговоре, AI интерпретирует буквально или галлюцинирует ответ. В отличие от живой команды, AI не спрашивает «вы имели в виду X или Y?» — он молча выбирает один вариант.

Столп 2: AI-First, но не AI-Only

Основной режим работы Супервайзера — AI-направляемая разработка. Ручное кодирование — обоснованное исключение, а не запрет.

Когда ручное вмешательство оправдано:

- Микрофиксы (1–3 строки), подготовка контекста для которых дороже самого исправления
- AI застрял в цикле на неверном подходе
- Инфраструктурная конфигурация, где AI галлюцинирует
- Срочные хотфиксы
- AI сделал 90% верно — проще поправить 3 строки, чем заново объяснять

Типичные ошибки:

- *Теневое кодирование* — написание кода и его сокрытие от системы прослеживаемости

- *Микроменеджмент* — переписывание >30% вывода вместо улучшения контекста
- *Штамповка* — принятие результата без проверки

Столп 3: Принуждение вместо церемоний

Качество обеспечивается автоматизированными шлюзами, а не совещаниями.

AI-агенты не ходят на совещания, не чувствуют ответственности и не учатся на ретроспективах. Единственный надёжный механизм качества — автоматизированное принуждение.

Назначение	Механизм
Решить, что строить	Церемония
Проверить качество кода	Шлюз
Обзор со заинтересованной стороной	Церемония
Проверить выполнение требования	Шлюз

Столп 4: Сохранение знаний

Что не задокументировано — не существует для AI.

Документация кода — контекст, а не формальность. В AI-нативной разработке документация кода служит двум целям: помогает Супервайзерам-людям понять систему и сокращает объём контекста для каждой задачи. Хорошо задокументированный модуль (назначение, контракты публичного API, архитектурные границы) избавляет Супервайзера от необходимости заново объяснять всё это в каждой задаче. AI читает документацию, понимает роль модуля и генерирует код, который вписывается в архитектуру.

Незадокументированный код вынуждает Супервайзера компенсировать: длиннее описания задач, больше явных ограничений, больше границ области. Это дорогой контекст, которому место в самом коде. Документация вроде «обрабатывает OAuth для Google и GitHub, хранит токены зашифрованными в таблице сессий, обновляет автоматически» — экономит 5 строк контекста задачи при каждой работе с авторизацией.

Правило 9.11 (Документация кода как контекст) делает это обязательным требованием (ОБЯЗАТЕЛЬНО): документация кода, достаточная для того, чтобы AI понял назначение модуля, контракты API и границы без чтения всей реализации.

Что фиксировать: решения (с обоснованием), паттерны, подводные камни, тупиковые подходы (наибольшая ценность повторного использования), наблюдения.

Доставка знаний в AI: статические файлы контекста, поисковые API, MCP-интеграции или RAG. Выбирайте осознанно.

Жизненный цикл знаний: `current` → `needs_review` (помечено как потенциально устаревшее) → `deprecated`. Устаревшие записи активно вредят качеству — они дают AI неверный контекст.

Столп 5: Паттерны взаимодействия

Управление AI — это диалог, а не одноразовая команда.

Паттерн	Когда использовать
Планирование-затем-исполнение	Сложные задачи: запросить у AI план → проверить → утвердить → выполнить
Итеративное уточнение	Большинство задач: сгенерировать → проверить → скорректировать → улучшить
На основе примера	UI/паттерны: «сделай как здесь, но с изменением X»
Негативный пример	Известные ловушки: «не делай X, потому что Y»
Контрольная точка и проверка	Многошаговые задачи: AI делает шаг 1 → проверка → шаг 2
Ограждение области	Контроль объёма: «измени ТОЛЬКО файлы X, Y. НЕ трогай A, B»
Откат и повтор	Неверное направление: откат через git, перезапуск с другим контекстом
Исследование	Неизвестная территория: исследовать прежде, чем выбрать подход

Управление контекстным окном

- Контекст AI деградирует по мере роста беседы (насыщение)

- Прогрессивное раскрытие: предоставляйте информацию по мере необходимости, а не сразу всю
- Стратегические контрольные точки очищают контекст и начинают с чистого листа
- Для больших кодовых баз: адресное чтение файлов вместо массовых дампов
- Передачи (handoff) сериализуют ключевой контекст для новых сессий
- Если AI начинает путать — дело не в модели, контекст переполнен

Управление галлюцинациями

Распространённые типы: несуществующие API/методы/флаги CLI, несуществующие файлы, правдоподобный код с тонкими ошибками на крайних случаях, уверенные, но ошибочные утверждения.

Эвристики обнаружения:

- Чрезмерная уверенность в новаторских решениях (красный флаг)
- «Подозрительно идеальный» код, обрабатывающий все крайние случаи
- Ссылки на пути/API, которые вы не узнаёте
- Всегда запускайте код — не доверяйте утверждениям AI о том, что он делает

Столп 6: Эмпирическая калибровка

Каждое правило и целевое значение должно калиброваться по вашим данным.

SENAR даёт формулы, а не целевые показатели. Организации устанавливают базовые значения, измеряя их на протяжении 3+ инкрементов, а затем задают цели на основе собственной реальности.

Типичные ошибки:

- *Карго-культ* — копирование целевых показателей другой организации
- *Преждевременная оптимизация* — агрессивные цели до понимания базовых показателей
- *Зацикленность на метриках* — оптимизация ради метрики, а не ради результата

Гид SENAR: Чеклист ревью AI-вывода

> ***ПРИМЕЧАНИЕ — Соотношение с SENAR Core***
>
> *SENAR Core содержит обновлённый **Чеклист верификации** (26 пунктов, 3 уровня: **Standard / High / Critical**), который заменяет данный чеклист для команд, использующих Core.*
>
> ***Соответствие уровней:** Tier 1 (этот гид) ≈ Standard (Базовая) (Core) | Tier 2 ≈ High | Tier 3 ≈ Critical*
>
> *Данный чеклист Гида сохраняется для команд, осваивающих SENAR через Standard (командная точка входа) и ещё не перешедших на Core. Если ваша команда использует SENAR Core — применяйте Чеклист верификации из Core.*

При ревью AI-сгенерированного вывода проверяйте эти специфичные для AI проблемы.

Чеклист

#	Проверка	На что обращать внимание
1	**Область изменений**	AI изменил файлы за пределами области задачи? (самая частая ошибка AI)
2	**Удаления**	AI молча удалил или подменил существующий рабочий код?
3	**Фантомные импорты**	Все импортированные пакеты указаны в файле зависимостей?
4	**Версии зависимостей**	Указанные версии реально существуют и опубликованы?
5	**Захардкоженные значения**	Магические числа, URL-адреса, учётные данные, API-ключи в коде?
6	**Переусложнение**	Ненужные абстракции, паттерны или обобщения?
7	**Дублирование**	Новый код дублирует существующие утилиты?

8	**Качество тестов**	Тесты проверяют поведение или просто зеркалят реализацию?
9	**Подмена тестов**	AI изменил тесты, чтобы они прошли, вместо исправления кода?
10	**Безопасность**	Открытый CORS, захардкоженные токены, SQL без параметризации?
11	**Крайние случаи**	Основной сценарий работает — а что с null, пустым значением, граничными условиями, конкурентным доступом?
12	**Именование**	AI следует конвенциям именования проекта?
13	**Область коммита**	Коммит атомарный и сфокусированный, или свалка всего подряд?
14	**Null-guard перед сравнением**	`None == None` — это `True` в Python (JS: `null === null` — true; в большинстве языков есть аналогичные ловушки null-равенства) — проверки доступа обходятся, когда обе стороны null?
15	**Обход через пустой конфиг**	Проверка безопасности пропускается, когда значение конфига — пустая строка? (`if secret and ...` проваливается в открытое состояние)
16	**Доверие заголовкам**	X-Forwarded-For, X-Partner-ID, Content-Length используются для безопасности без валидации прокси?
17	**IDOR**	Ресурс доступен по ID без проверки прав пользователя на этот ресурс? Аутентификация != авторизация.
18	**Шорткат return True**	Функция контроля доступа возвращает True / предоставляет доступ без явной проверки владения?
19	**Инъекция через форматирование строк**	`str.format(**untrusted_dict)` — Python format поддерживает доступ к атрибутам, что позволяет инъекцию (JS: шаблонные литералы с `eval()` ; C/C++: форматные строки `printf` ; любой язык: строковая интерполяция с непроверенными данными)
20	**God-функции/ файлы**	Функции >50 строк, файлы >400 строк — главный признак нерефакторенного AI-вывода
21	**Недостижимый защитный код**	`return False` после исчерпывающей обработки исключений — AI добавил «на всякий случай», но код никогда не выполнится

22	**Проглоченные исключения**	Блоки `catch/except`, которые отбрасывают ошибки (`except Exception: pass`, `catch(e) {}`, или логирование на уровне debug) — скрывают реальные сбои. Проверяйте возврат null/None/nil, маскирующий ошибочные состояния
23	**Небезопасная десериализация**	pickle.loads, yaml.load без SafeLoader, eval/exec на непроверенных данных, JSON prototype pollution? (Java: `ObjectInputStream`; JS: `eval(JSON)`; любой язык: десериализация непроверенных данных без валидации)

Уровни приоритета

Уровень 1 — Проверять всегда (каждая задача): Пункты 1–3 (область изменений, удаления, фантомные импорты) и 8–9 (качество тестов, подмена тестов). Они отлавливают самые частые и самые опасные дефекты AI. Проверка 5 пунктов занимает менее 2 минут.

Уровень 2 — Задачи, связанные с безопасностью (авторизация, платежи, данные, API): Пункты 10 (безопасность), 14–18 (null-guard, пустой конфиг, доверие заголовкам, IDOR, return True). Они отлавливают латентные дефекты — AI-вывод, который выглядит корректно, но ломается при враждебных условиях. Выявлены через состязательный аудит продакшен-кода, сгенерированного AI.

Уровень 3 — Глубокое ревью (сложные задачи, вывод агентов): Все оставшиеся пункты (4–7, 11–13, 19–23). Применяйте при ревью сложных фич, рефакторингов или любого вывода от диспетчеризованных агентов (Rule 15 L3).

Использование

Распечатайте этот список. Используйте его на QG-2 (шлюз реализации) для каждой задачи. Со временем часть проверок станет автоматической привычкой — но держите список на виду для новых Супервайзеров.

Гид SENAR: Сквозной разбор

Пути внедрения

У SENAR два пути внедрения. Начните с Базовой (Core); переходите на Standard, когда потребности перерастут его.

Базовый путь (индивидуально или малая команда)

1. **Примите 8 правил** — Задача перед кодом, Границы области, Проверка по критериям, Тесты проверяют требования, Проверка на латентные дефекты, Нулевая толерантность к незавершённой работе, Устранение причин, а не симптомов, Фиксация знаний.
2. **Внедрите 2 шлюза** — Стартовый шлюз (цель + КП + негативный сценарий + область до реализации) и Финальный шлюз (все КП проверены, чеклист пройден, знания зафиксированы).
3. **Измеряйте 2 метрики** — FPSR (First-Pass Success Rate) и DER (Dead End Rate). Установите базовые значения за 3+ цикла, прежде чем задавать целевые.

Это всё. Никаких ролей, церемоний, управления сессиями. Разбор ниже демонстрирует этот путь.

Путь Standard (команда или организация)

1. **Начните с Базовой** — примите все 8 правил, оба шлюза, обе метрики.
2. **Добавьте Начальную конфигурацию (Foundation)** — 1–3 пары; добавьте дисциплину сессий (Старт/Конец сессии, ежемесячный Обзор качества); Супервайзер совмещает роли Архитектора контекста, Инженера знаний и Инженера верификации. 11 правил, 4 метрики (добавляются Throughput и Lead Time), QG-0 + QG-2.
3. **Добавьте Командную конфигурацию** — введите выделенные роли (Архитектор контекста, Менеджер потока, Инженер верификации), полные церемонии (Планирование инкремента, Обзор качества, Ретроспектива).
4. **Добавьте Корпоративную конфигурацию** — федеративную координацию, кросс-проектные метрики, портфельное управление, аудиторские следы соответствия.

Раздел «Командный разбор» ниже демонстрирует путь Standard в Командной конфигурации.

Базовый разбор (одиночный Супервайзер)

Один человек, один AI-агент, одна сессия. Создаём эндпоинт REST API.

Старт сессии (2 мин): Открываем терминал. Загружаем предыдущую передачу контекста: «Модуль авторизации готов. Далее: эндпоинт профиля пользователя.» Выбираем задачу `impl-user-profile`.

Проверка стартового шлюза: Цель: «GET /users/me возвращает профиль текущего пользователя.» КП: «1. Возвращает 200 с данными пользователя. 2. Возвращает 401 без токена. 3. Включает информацию о компании.» Связь с историей: `user-management`. → Проходит.

Исполнение: Направляем AI с ограждением области:

> *«Реализуй GET /users/me. КП: [выше]. Следуй паттернам из auth/router.py. Измени ТОЛЬКО директорию users/.»*

(Этот пример использует Python/FastAPI. Тот же паттерн применим к Java/Spring Boot @RequestMapping, Go/Gin router.GET, TypeScript/NestJS @Get().)

AI генерирует роутер + тест. Ревью по чеклисту: область ✓, импорты ✓, нет захардкоженных значений ✓, тесты проверяют поведение, а не реализацию ✓. Одна проблема: AI не протестировал случай 401.

> *«Добавь тест для неаутентифицированного запроса → 401.»*

AI добавляет тест. Повторное ревью: чисто.

Шлюз реализации: CI зелёный, 6/6 тестов проходят, муру чисто, КП проверены. → Готово. (Для других стеков: Java — javac + SpotBugs, Go — go vet + staticcheck, TypeScript — tsc --strict.)

Фиксация тупиковых подходов: Нет для этой задачи — всё было прямолинейно.

Конец сессии (3 мин): 4 задачи выполнено, 75 минут. Передача контекста: «Профиль, настройки, загрузка аватара готовы. Далее: смена пароля. Внимание: загрузка аватара требует валидации размера файла.»

Суммарные накладные расходы: 5 минут (старт + конец). Никаких церемоний, ролей, совещаний. Только дисциплина.

> ***Раздел ниже демонстрирует SENAR на командном уровне.*** *Если вы используете только Базовую (Core) — можете остановиться здесь: Базовый разбор выше является вашим полным рабочим процессом.*

Командный разбор (3 пары)

Полный цикл SENAR для Командной конфигурации. 3 пары Супервайзер+AI, веб-приложение.

Планирование инкремента

Архитектор контекста ведёт:

1. Обзор пула задач: 32 задачи-кандидата в 6 историях
2. Применяет WSJF: Аутентификация получает наивысший приоритет (блокирует другую работу, умеренный размер)
3. Назначает: Пара A → авторизация (6 задач), Пара B → CRUD заказов (8 задач), Пара C → CI/CD (5 задач)
4. Риски: «Выбор библиотеки авторизации может потребовать исследования» → создаёт задачу типа Exploration
5. Бюджет: 19 задач отобрано для этого инкремента

Сессия Супервайзера

Старт сессии (2 мин): Загрузка передачи контекста → «Модель авторизации готова. Далее: эндпоинт логина.» Выбрать задачу `impl-login-endpoint`. Dev-окружение зелёное.

QG-0 (автоматически): Цель ✓, КП ✓ (4 критерия), связь с требованием ✓, тип работы ✓ → проходит.

Исполнение (Планирование-затем-исполнение + Ограждение области):

> *«Реализуй POST /auth/login. КП: [критерии]. Следуй паттернам из auth/models.py. Используй PyJWT. Конфиг в settings.py. Измени ТОЛЬКО директорию auth/.»*

(Этот пример использует Python/FastAPI с PyJWT. Тот же паттерн ограждения области применим к любому стеку: Java/Spring Security с jjwt, Go с golang-jwt, TypeScript/NestJS с @nestjs/jwt.)

AI генерирует роутер, сервис, тесты. Супервайзер проверяет по чеклисту:

- Область: ✓ только файлы auth/
- Фантомные импорты: ⚠ AI импортировал `python-jose`, но в проекте используется `PyJWT`
- Захардкоженные значения: ✓ секрет токена из env
- Крайние случаи: ⚠ нет теста на просроченный токен

Итеративное уточнение:

> *«Две проблемы: 1) Используй PyJWT, не python-jose. 2) Добавь тест на просроченный токен.»*

AI исправляет. Повторное ревью: чисто.

QG-2 (автоматически): CI ✓, 14/14 тестов ✓, mypy ✓, lint ✓, КП проверены ✓, безопасность ✓ → проходит.

Фиксация знаний: «Подводный камень: AI по умолчанию использует python-jose для JWT, даже когда PyJWT указан в requirements. Всегда указывайте библиотеку в контексте.»

3 задачи выполнено, 110 минут.

Конец сессии (5 мин): Метрики сохранены, передача контекста написана, 1 запись в базу знаний.

Обзор качества

Процедура Обзора качества:

1. Выберите 3–5 недавно завершённых задач (случайно или с учётом рисков)
2. Для каждой задачи проверьте: (а) КП имеют свидетельства, (б) чеклист применён на правильном уровне, (в) знания зафиксированы
3. Зафиксируйте находки как наблюдения в базе знаний
4. Если FPSR падает ниже целевого значения, обсудите корневые причины на следующей Синхронизации команды (Team Sync)

После 6 сессий Инженер верификации проводит аудит:

Находка	Действие
Дублирующаяся утилита валидации (есть в shared/)	Задача: рефакторинг
3 TODO-комментария, оставленных AI	Задача: устранить
Нулевое покрытие тестами удаления заказов	Задача: добавить тесты
Непоследовательный формат ошибок между модулями	Задача: стандартизировать

Знание: «Паттерн: AI недостаточно тестирует операции удаления — добавить в стандартный шаблон КП.»

Ретроспектива инкремента

Метрика	Значение
Пропускная способность	7,3 задачи/сессия
Lead Time (медиана)	35 минут
FPSR	74%
DER	5,3%
KCR	0,42
Предсказуемость затрат	110%
MIR	12%

Действия: улучшить шаблон контекста (библиотека JWT), добавить «тестировать удаление» в шаблон КП, пересмотреть QG-2 для тестов конкурентности.

Гид SENAR: Переход от традиционных ролей и от Базовой (Core) к Standard

Переход с SENAR Core на SENAR Standard

Когда переходить

SENAR Core достаточен для индивидуальных разработчиков и малых команд (1–2 человека). Рассмотрите переход на полный Standard, когда:

- **Команда растёт больше 3 человек** — нужны роли (Архитектор контекста, Менеджер потока, Инженер верификации), чтобы координировать работу между Супервайзерами.
- **Несколько проектов** — кросс-проектные зависимости требуют федеративной координации, а не переписки в чатах.
- **Регуляторные требования** — нужны аудиторские следы, формальная прослеживаемость требований и документированные свидетельства качества.
- **Информационные силосы** — знания не попадают в базу знаний, и разные Супервайзеры выдают непоследовательные результаты.
- **DER выходит на плато** — Dead End Rate перестаёт улучшаться, потому что знания фиксируются, но не переиспользуются систематически по всей команде.

Что Standard добавляет к Базовой (Core)

Измерение	SENAR Core	SENAR Standard
Правила	8 правил	15 правил (добавляет управление сессиями, жизненный цикл знаний, документацию кода, федерацию)
Шлюзы	Стартовый шлюз + Финальный шлюз	5 шлюзов: QG-0 (Контекст) + QG-1 (Требование) + QG-2 (Реализация) + QG-3 (Верификация) + QG-4 (Развёртывание)

Метрики	FPSR + DER	10 метрик (добавляет Throughput, Lead Time, KCR, MIR, Cost per Task, Cost Predictability и другие)
Роли	Супервайзер (неявно)	Супервайзер, Архитектор контекста, Менеджер потока, Инженер верификации, Инженер знаний
Церемонии	Нет	Старт/конец сессии, Планирование инкремента, Обзор качества, Ретроспектива инкремента
Конфигурации	Не применимо	Начальная, Командная, Корпоративная — прогрессивные уровни внедрения (Базовая служит точкой входа)
Управление сессиями	Не регламентировано	Лимиты длительности, каденция контрольных точек, дисциплина передачи
Мульти-команда	Не покрыто	Federation Sync, кросс-проектные зависимости, портфельные метрики
Модель зрелости	Не применимо	6 измерений, 5 уровней зрелости по каждому

Шаги миграции

Если вы уже используете SENAR Core, миграция на Standard инкрементальна — ничего не нужно переучивать.

Шаг 1: Сопоставьте практику Базовой конфигурации со Standard (День 1) Ваши 8 правил Базовой конфигурации напрямую соответствуют правилам Standard. Стартовый шлюз соответствует QG-0. Финальный шлюз соответствует QG-2. FPSR и DER остаются вашими основными метриками. SENAR Core — это точка входа в Standard: вы готовы к Начальной конфигурации (Foundation).

Шаг 2: Добавьте дисциплину сессий — Начальная конфигурация (Неделя 1) Введите формальные Старт сессии (загрузка контекста, выбор задач, проверка окружения) и Конец сессии (написание передачи контекста, фиксация метрик, документирование знаний). Ограничьте сессии 4–6 часами. Добавьте ежемесячный Обзор качества. Это и есть Начальная: 11 правил, 3 совмещённые роли (Супервайзер покрывает Архитектора контекста + Инженера знаний + Инженера верификации), 4 метрики (добавляются Throughput и Lead Time). Подходит для 1–3 пар.

Шаг 3: Добавьте командные церемонии (Неделя 2–4) Когда у вас 3+ Супервайзера:

- **Планирование инкремента** — ведёт Архитектор контекста: обзор бэклога, приоритизация по WSJF, назначение задач парам.
- **Обзор качества** — Инженер верификации проводит аудит: выборочная проверка недавней работы на архитектурную согласованность, качество тестов, актуальность знаний.
- **Ретроспектива инкремента** — ведёт Менеджер потока: обзор метрик, выявление улучшений процесса, установка целевых показателей.

Шаг 4: Добавьте организационные метрики (Месяц 2+) Помимо FPSR и DER, начните отслеживать: Throughput (задачи/сессия), Lead Time, Knowledge Capture Rate (KCR), Cost per Task, Cost Predictability. Установите базовые значения за 3 инкремента перед установкой целевых.

Шаг 5: Добавьте управление (при необходимости) Корпоративная конфигурация (10+ пар): федеративная координация между проектами, QG-3 (независимая верификация), QG-4 (шлюз развёртывания), прослеживаемость требований, аудиторские следы.

Соответствие правил Базовой конфигурации правилам Standard

Правило Базовой конфигурации	Эквивалент в Standard
1. Задача перед кодом	Rule 1 (S10.1) + QG-0 (S8.1)
2. Границы области	QG-0 РЕКОМЕНДУЕТСЯ-критерии + привычка из гида
3. Проверка по критериям	QG-2 (S8.3) + Rule 15 L2 (S10.15)
4. Тесты проверяют требования	QG-2 критерии тестов + QG-3 валидность тестов (S8.4)
5. Проверка на латентные дефекты	Rule 15 (S10.15) + Чеклист ревью AI (Гид)
6. Нулевая толерантность к незавершённой работе	QG-2 принуждение (S8.6)
7. Устранение причин, а не симптомов	Rule 5 Периодический аудит (S10.5) + Режимы сбоев из гида
8. Фиксация знаний	Rule 4 Документация тупиков (S10.4) + Rule 9 Фиксация знаний (S10.9)

Сопоставление ролей

Текущая роль	Ответственность в SENAR	Переносимые навыки	Навыки для развития
Старший разработчик	Супервайзер	Код-ревью, архитектура, доменные знания, отладка	Проектирование контекста, взаимодействие с AI, делегирование вместо исполнения
Тимлид	Старший Супервайзер	Архитектура, компромиссы, менторство	AI-first мышление, меньше ручного кодирования
QA-инженер	Инженер верификации	Верификация, мышление о крайних случаях, критерии приёмки	Паттерны AI-вывода, обнаружение галлюцинаций
DevOps-инженер	Супервайзер (инфраструктура)	Автоматизация, CI/CD, инфраструктура	AI-направляемая конфигурация, контекст для инфраструктуры
Владелец продукта	Архитектор контекста	Требования, приоритизация, управление стейкхолдерами	Структурирование требований для потребления AI, прослеживаемость
Скрам-мастер	Менеджер потока	Фасилитация, оптимизация процессов	Управление на основе метрик, отслеживание затрат

Карьерный путь Супервайзера

Уровень	Фокус	Ключевая компетенция
Младший Супервайзер	Отдельные задачи, тщательная проверка	Следование паттернам, тщательное ревью
Супервайзер	Множество задач, архитектурные решения	Проектирование контекста, суждение о компромиссах

Старший Супервайзер	Кросс-проектное влияние, менторство	Архитектурное направление, ревью чужой работы
Ведущий Супервайзер	Оптимизация процессов, методология	Проектирование шлюзов качества, оценка зрелости

Типичные сложности перехода

«Я быстрее наберу сам» — Верно для тривиальных задач. Для средних и сложных задач проектирование контекста + AI-генерация быстрее ручного кодирования и даёт тесты, документацию и прослеживаемый результат. Измерьте.

«Я не доверяю AI-результату» — Здоровый инстинкт. Именно для этого существуют QG-2 и чеклист ревью. Верификация — теперь ваш ключевой навык, а не доверие.

«Моя ценность была в написании кода» — Ваша ценность была в решении задач. Вы по-прежнему их решаете — но теперь через направление и суждение, а не набор текста. И задачи становятся крупнее.

Гид SENAR: Сравнение с SAFe

SENAR — самостоятельная методология, вдохновлённая концепциями SAFe и спроектированная для AI-нативных команд. Организации, использующие SAFe для «живых» команд, могут внедрить SENAR для AI-нативных потоков, планируя сосуществование, а не наложение.

Ключевые отличия

Аспект	SAFe	SENAR	Почему иначе
Единица производства	Команда (5–9 человек)	Пара Супервайзер+AI	AI заменяет команду как производитель
Метрика доставки	Velocity (SP/спринт)	Throughput (задачи/сессия)	Пропускная способность AI непредсказуема; оценка в SP создаёт лишние накладные расходы
Единица времени	Спринт (2 недели)	Сессия (часы)	AI работает всплесками, а не ритмично
Планирование	PI Planning (2 дня, весь ART)	Планирование инкремента (1 сессия)	Синхронизация нескольких команд не нужна
Механизм качества	DoR/DoD (командные договорённости)	Шлюзы качества (автоматизированный код)	AI не чувствует ответственности
Знания	CoP, wiki, племенные знания	Явная база знаний (обязательная)	У AI нет долговременной памяти
Ретроспектива	Качественная (чувства + данные)	Количественная (только метрики)	AI-работа порождает точные измерения

Координация	Scrum of Scrums, ART Sync	Federation Sync	Программная трассировка зависимостей
Масштабирование	Essential → Full → Portfolio SAFe	Базовая → Командная → Корпоративная SENAR	Та же концепция, другая модель производства

Сопоставление ролей

Роль в SAFe	Эквивалент в SENAR	Ключевое отличие
Разработчик	Супервайзер	Супервайзер направляет AI, а не пишет код как основную деятельность
Владелец продукта	Архитектор контекста (частично)	АК проектирует контекст для AI, а не бэклог для людей. Бизнес-ценность делится с Менеджером потока
Скрам-мастер	Менеджер потока	МП управляет ритмом и затратами, а не динамикой команды. Servant leadership менее релевантно с AI
QA	Инженер верификации	ИВ пишет критерии приёмки (AI пишет тесты). Фокус смещается на AI-специфичные паттерны дефектов
RTE	Менеджер потока (корпоративный)	Корпоративный МП координирует несколько пар, как RTE координирует команды
Системный архитектор	Главный Супервайзер (корпоративный)	Архитектурное управление по всем парам

Что есть в SAFe, чего нет в SENAR

Элемент SAFe	Подход SENAR
Сообщества практик	База знаний заменяет устную передачу знаний машиночитаемыми записями
Architectural Runway	Сохранение знаний + документация тупиковых подходов выполняют схожую роль

IP Iteration	Время на инновации рекомендуется в Ретроспективе инкремента
Built-In Quality	Шлюзы качества (более строгое принуждение)
Solution Train / Large Solution	Корпоративная конфигурация с координаторами федерации (менее развито)

Миграция с SAFe на SENAR

Организации, переводящие AI-потоки с SAFe на SENAR:

1. Начать с SENAR Core на одной паре — валидировать минимально жизнеспособный процесс
2. Запускать SAFe и SENAR параллельно для потоков с людьми и AI-потоков
3. Отслеживать оба набора метрик; сравнивать доставку и качество
4. Расширять SENAR до Командной конфигурации, когда 3+ пар продуктивны
5. SAFe остаётся для потоков, где люди пишут основную часть кода

Руководство SENAR: Режимы отказа

SENAR может давать сбои. Любая методология может. Этот документ описывает, как именно внедрения SENAR терпят неудачу, как обнаружить проблемы на ранней стадии и что с ними делать.

Режимы отказа разделены на три категории: процессные (методология принята, но практикуется неправильно), организационные (люди и структуры вокруг SENAR деградируют) и технические (AI-инструменты и инфраструктура подрывают процесс).

Процессные отказы

PF-1: Теневое кодирование

Описание: Супервайзеры тайно пишут код вместо направления AI-агентов. Задача выглядит как AI-сгенерированная, метрики в норме, но реализацию делает человек. Это подрывает главную предпосылку SENAR: ценность Супервайзера — в суждениях, а не в наборе текста.

Ранние признаки:

- Manual Intervention Rate превышает 30% без соответствующих записей Dead End, объясняющих, почему AI не справился
- Временные метки коммитов группируются за пределами окон сессий
- Супервайзеры сообщают, что «быстрее без AI», но не могут сформулировать, с чем именно AI не справляется
- В базе знаний нет записей Dead End или Gotcha — всё «просто работает» (потому что человек сделал сам)
- Количество вызовов инструментов за сессию подозрительно мало относительно сложности задач

Действия по восстановлению:

1. Провести непредвзятый аудит: сравнить git diff с логами сессий AI на выборке задач
2. Определить, какие типы задач провоцируют теневое кодирование — скорее всего, у них плохие шаблоны контекста

3. Улучшить качество контекста для этих типов задач (более точные критерии приёмки, паттерны, архитектурные ограничения)
4. Если AI действительно не справляется с определёнными типами задач, создать явный тип работы «ручная реализация» с отдельным отслеживанием

Профилактика:

- Нормализовать ручное вмешательство как документируемую, отслеживаемую активность — а не постыдный секрет
- Отслеживать Manual Intervention Rate как диагностическую метрику, а не метрику эффективности
- Убедиться, что Супервайзеры понимают: высокий MIR для определённых типов задач — сигнал к улучшению контекста, а не личная неудача
- Логи сессий должны проверяться при Quality Sweep — не для контроля, а для выявления пробелов в контексте

PF-2: Формальное утверждение

Описание: Супервайзеры принимают результаты AI без содержательной проверки. Код проходит QG-2 (автоматические тесты, линтер, типы), но никто не проверяет, правильно ли он решает задачу, обрабатывает ли граничные случаи, соответствует ли архитектуре. Человек превращается в нажимателя кнопок.

Ранние признаки:

- Defect Escape Rate растёт, а First-Pass Success Rate остаётся подозрительно высоким (всё «проходит», но баги появляются позже)
- Комментарии к code review отсутствуют или формальны («LGTM», «looks good»)
- Время верификации задачи опускается ниже правдоподобного минимума (изменение в 200 строк проверено за 30 секунд)
- Архитектурный дрейф: паттерны расходятся по кодовой базе, потому что никто не следит за единообразием
- Уязвимости безопасности появляются в продакшене, хотя рецензент-человек должен был их заметить

Действия по восстановлению:

1. Обязательный Quality Sweep с фокусом на последние 2 инкремента — сравнить слитый код с требованиями
2. Ввести QG-3 (Шлюз качества верификации), если ещё не активен: независимая проверка другим Супервайзером
3. Установить минимальные пороги времени проверки пропорционально объёму изменений

4. Создать «Чек-лист проверки AI» (см. Руководство 02) и требовать его заполнения для каждой задачи

Профилактика:

- QG-3 с независимой верификацией — самая сильная защита
- Ротация назначений Verification Engineer, чтобы проверяющие оставались внимательными
- Отслеживать Defect Escape Rate на видном месте — это главный индикатор формального утверждения
- Quality Sweep должен проверять реальный код, а не только дашборды метрик

PF-3: Нормализация обхода шлюзов

Описание: Шлюзы качества существуют, но их систематически обходят с документированными исключениями. «Исправим потом» становится нормой. Обход шлюзов задуман для настоящих экстренных ситуаций; когда он входит в привычку, шлюзы превращаются в декорацию.

Ранние признаки:

- Более 10% задач имеют записи об обходе шлюзов
- Один и тот же шлюз обходится многократно (например, QG-0 пропускается, потому что «мы и так знаем, что делать»)
- Обоснования обхода становятся шаблонными («нехватка времени», «исправим в следующем инкременте»)
- Задачи технического долга, созданные из обходов, никогда не завершаются
- Новые члены команды усваивают обход шлюзов как стандартную практику

Действия по восстановлению:

1. Аудит всех записей обходов за последние 2 инкремента — категоризировать по шлюзу и обоснованию
2. Для каждого регулярно обходимого шлюза определить: шлюз слишком строгий (скорректировать критерии) или команда недообучена (вложиться в процесс)?
3. Требовать утверждения обхода от кого-то, кроме самого Супервайзера
4. Ввести «бюджет обходов» на инкремент — максимальное число допустимых обходов. По исчерпании шлюз блокируется жёстко

Профилактика:

- Отслеживать частоту обходов как метрику на ретроспективах инкрементов
- Ретроспектива должна включать «обзор обходов» — каждый обход обсуждается, а не просто считается

- Критерии шлюзов следует пересматривать ежеквартально: если шлюз регулярно обходится, возможно, нужна корректировка критериев, а не ужесточение дисциплины
- Различать «шлюз слишком строгий» (проблема критериев) и «дисциплина слишком низкая» (проблема культуры) — реакция разная

PF-4: Устаревание базы знаний

Описание: База знаний была наполнена при первоначальном внедрении, но больше не обновляется. Записи ссылаются на старые паттерны, устаревшие API или архитектурные решения, которые уже пересмотрены. AI-агенты получают устаревший контекст и генерируют результат на основе неактуальной информации.

Ранние признаки:

- Knowledge Capture Rate падает ниже 0.3 записей на задачу на протяжении 3+ последовательных сессий
- Новые Супервайзеры сообщают, что записи базы знаний противоречат тому, что они видят в коде
- AI-агенты генерируют результат с использованием паттернов из базы знаний, от которых команда уже отказалась
- Даты «последней проверки» записей старше 3 инкрементов
- Записи Dead End ссылаются на инструменты или библиотеки, которых уже нет в стеке

Действия по восстановлению:

1. Запланировать выделенную сессию аудита базы знаний — оформить как задачу, а не побочную активность
2. Пометить каждую запись статусом свежести: актуальна, требует проверки, устарела
3. Немедленно удалить или архивировать устаревшие записи — устаревший контекст хуже его отсутствия
4. Назначить обслуживание базы знаний явной обязанностью (роль Knowledge Engineer)

Профилактика:

- Церемония Quality Sweep (Стандарт, раздел 7.4) должна включать аудит свежести базы знаний
- Записи должны иметь метку «последняя проверка», обновляемую при подтверждении актуальности

- Установить целевой Knowledge Capture Rate (0.3–0.5 записей/задача) и отслеживать на ретроспективах
- Если задача порождает результат, противоречащий записи базы знаний, запись должна быть обновлена до закрытия задачи

PF-5: Прекращение документирования тупиков

Описание: Команды перестают документировать неудачные подходы. Dead End — один из самых ценных типов знаний: они предотвращают повторение ошибок AI-агентами. Под давлением сроков документирование тупиков сокращается первым, потому что оно описывает то, что не сработало, а это кажется малоприоритетным.

Ранние признаки:

- Записи Dead End за инкремент падают до нуля
- AI-агенты пробуют подходы, которые уже были опробованы и отвергнуты в предыдущих сессиях
- Длительность сессий увеличивается, потому что AI исследует пути, о провале которых человек помнит, но не задокументировал
- Документы передачи упоминают «мы пробовали X, но не сработало» без создания записи в базе знаний

Действия по восстановлению:

1. Проверить последние 5 документов передачи на упоминание неудачных подходов — создать запись Dead End для каждого
2. Добавить «фиксацию Dead End» как явный шаг в церемонию Session End
3. Сделать документирование Dead End частью критериев приёмки QG-2 для сложных задач

Профилактика:

- Шаблон Session End должен включать раздел «Неудачные подходы», который становится записью Dead End
- Метрику Knowledge Capture Rate следует декомпозировать по типам записей — здоровая база знаний содержит микс решений, паттернов, подводных камней и тупиков
- Отмечать документирование Dead End как ценность: это не хроника неудач, а карта территории

PF-6: Размывание дисциплины сессий

Описание: Сессии растягиваются за разумные пределы. Супервайзеры пропускают Session Start (нет загрузки контекста, нет выбора задач) или Session End (нет передачи, нет сбора метрик). Граница сессии — основной ритмический механизм SENAR — растворяется в непрерывной, бесструктурной работе.

Ранние признаки:

- Длительность сессий регулярно превышает 8 часов
- Записи Session Start не содержат списка задач или проверки контекста
- Документы передачи пусты или отсутствуют
- Контрольные точки никогда не создаются
- Количество вызовов инструментов за сессию превышает 500 (зона усталости)
- Качество результатов AI деградирует во второй половине сессий (исчерпание контекстного окна)

Действия по восстановлению:

1. Установить жёсткие ограничения по времени сессий — рекомендуемый диапазон 4–6 часов
2. Сделать Session End блокирующим шагом: следующая сессия не может начаться без завершённой передачи
3. Внедрить автоматические напоминания о контрольных точках с интервалом 40–60 вызовов инструментов
4. Проверить последние 10 сессий на отсутствие передач и создать их ретроспективно

Профилактика:

- Session Start и Session End — это церемонии, а не необязательные шаги
- Инструментарий должен обеспечивать: сессия не может начаться, если предыдущая не имеет передачи
- Установить максимальный бюджет вызовов инструментов на сессию (300–500 в зависимости от сложности задач)
- Усталость Супервайзера реальна: качество верификации деградирует через 4–6 часов. Процесс должен учитывать человеческие ограничения

PF-7: Метрики тщеславия

Описание: Команды оптимизируют метрики, не улучшая результатов. Производительность раздувается за счёт дробления задач на тривиально мелкие единицы. First-Pass Success Rate накручивается снижением критериев приёмки. Lead Time сокращается за счёт начала задач до их готовности. Цифры выглядят хорошо; программное обеспечение — нет.

Ранние признаки:

- Производительность резко возрастает, а удовлетворённость заинтересованных сторон — нет
- Средняя сложность задач падает до «тривиальной» для 80%+ задач
- FPSR выше 95%, но Defect Escape Rate тоже выше 10% (противоречивые сигналы)
- Lead Time уменьшается, но развёрнутые фичи неполны или содержат баги
- Команды сопротивляются добавлению новых метрик или изменению существующих
- Cost per Task падает, но общая стоимость инкремента — нет

Действия по восстановлению:

1. Перекрёстная проверка метрик: Throughput x DER, FPSR x дефекты после развёртывания, Lead Time x процент доработки
2. Ввести «метрики результатов» наряду с процессными: частота развёртываний, процент отказов при изменениях, время восстановления
3. Пересмотреть рекомендации по гранулярности задач — задача должна представлять значимое, проверяемое изменение
4. Ретроспектива должна сопоставлять метрики с реальными результатами для заинтересованных сторон

Профилактика:

- Никогда не поощрять и не наказывать на основе одной метрики
- Метрики существуют для диагностики, а не для суждений — закрепить это явно в нормах команды
- DER — метрика, которую сложнее всего накрутить, потому что она измеряется внешней реальностью (баги в продакшене)
- Quality Sweep должен проверять, что метрики рассказывают непротиворечивую историю

Организационные отказы

OF-1: Выгорание Супервайзеров

Описание: Постоянная верификация результатов AI создаёт когнитивную усталость. Супервайзер должен поддерживать глубокое техническое понимание, одновременно обрабатывая большие объёмы сгенерированного кода. В отличие от традиционной

разработки, где разработчик наращивает контекст постепенно, Супервайзер должен быстро переключаться между задачами и верифицировать незнакомые паттерны кода.

Ранние признаки:

- Качество верификации падает в послеобеденных сессиях (формальное утверждение после обеда)
- Супервайзеры сообщают об ощущении «измотанности», несмотря на отсутствие написания кода
- Частота обходов шлюзов возрастает к концу рабочей недели
- Растут больничные и текучка кадров
- Супервайзеры предпочитают простые задачи и избегают сложных

Действия по восстановлению:

1. Сократить длительность сессий до 3–4 часов с обязательными перерывами
2. Чередовать глубокую работу по верификации с более лёгкими задачами (документация, курирование базы знаний)
3. Внедрить парную супервизию: два Супервайзера верифицируют сложные задачи друг друга
4. Обеспечить Супервайзерам автономию в выборе задач и планировании сессий

Профилактика:

- Проектировать сессии с учётом нагрузки на верификацию — не все задачи требуют одинакового уровня тщательности
- Закладывать время на работу вне верификации: обслуживание базы знаний, улучшение контекста, обучение
- Отслеживать количество вызовов инструментов и длительность сессий — это опережающие индикаторы выгорания
- Учитывать разницу между «AI может работать 24 часа» и «человек не может верифицировать 24 часа»

OF-2: Атрофия навыков

Описание: Супервайзеры полностью перестают писать код и постепенно теряют способность оценивать результаты AI на глубоком техническом уровне. Они замечают поверхностные проблемы (опечатки, очевидные баги), но пропускают архитектурные проблемы, антипаттерны производительности и тонкие уязвимости безопасности. Ценность «суждение важнее нажатий клавиш» требует, чтобы суждение оставалось острым.

Ранние признаки:

- Супервайзеры не могут объяснить, почему конкретный подход к реализации ошибочен
- Архитектурные решения всё чаще делегируются AI без возражений со стороны человека
- Code review сводится к «работает ли это?» вместо «правильный ли это подход?»
- Супервайзеры избегают задач в незнакомых частях кодовой базы
- Новые технологии принимаются без понимания Супервайзером компромиссов

Действия по восстановлению:

1. Ввести «дни ручной реализации» — периодические сессии, где Супервайзеры пишут код без помощи AI
2. Требовать от Супервайзеров описания архитектурного подхода до того, как AI приступит к реализации
3. Ротировать Супервайзеров между разными частями кодовой базы для стимулирования обучения
4. Включить «объясните реализацию» как часть верификации: если не можешь объяснить — не можешь проверить

Профилактика:

- Супервайзеры должны периодически реализовывать сложные задачи вручную для поддержания квалификации
- Время на техническое обучение должно быть заложено в бюджет — не как накладные расходы, а как поддержание компетенций
- Роль Context Architect поддерживает вовлечённость Супервайзеров в проектирование системы, а не только в верификацию задач
- Quality Sweep, требующий глубокого понимания кода, служит двойной цели: обеспечение качества и поддержание навыков

OF-3: Привязка к поставщику AI-инструментов

Описание: Внедрение SENAR становится глубоко связанным с конкретным инструментом, моделью или платформой AI. Шаблоны сессий, формат базы знаний, механизмы инъекции контекста и процессы верификации предполагают API и возможности конкретного поставщика. Переход становится непомерно дорогим.

Ранние признаки:

- Все шаблоны контекста используют специфичный для поставщика синтаксис или функции
- Формат базы знаний оптимизирован под размер контекстного окна одной модели

- Управление сессиями встроено в проприетарную платформу без экспорта
- Отслеживание затрат зависит от API биллинга конкретного поставщика
- Экспертиза команды в «использовании инструмента X», а не в «супервизии AI-агентов»

Действия по восстановлению:

1. Аудит всех точек соприкосновения между процессом SENAR и инструментарием AI — составить инвентарь абстракций
2. Определить вендоронезависимый интерфейс для инъекции контекста, захвата результатов и управления сессиями
3. Извлечь базу знаний в портативный формат (структурированный markdown, JSON)
4. Провести параллельные сессии с альтернативными инструментами для проверки портативности

Профилактика:

- SENAR намеренно не предписывает конкретные AI-инструменты — сохраняйте эту нейтральность
- Хранить базу знаний в формате, читаемом любой LLM (структурированный текст, не проприетарные эмбеддинги)
- Управление сессиями должно быть тонким слоем, а не платформой
- Периодически оценивать альтернативные инструменты для сохранения вариативности

OF-4: Избыточный процесс (бюрократический SENAR)

Описание: SENAR становится самоцелью. Каждая задача требует исчерпывающей документации. Каждая сессия начинается с 30-минутной церемонии. Каждая запись в базе знаний нуждается в трёх утверждениях. Методология, призванная повысить производительность, становится главным препятствием на пути к ней.

Ранние признаки:

- Session Start занимает больше 30 минут
- На церемонии SENAR тратится больше времени, чем на реальную работу
- Супервайзеры создают задачи «соответствие процессу», не приносящие ценности ПО
- Новые Супервайзеры подавлены количеством обязательных шагов
- Команды ведут две системы: «официальный» процесс SENAR и свой реальный рабочий процесс

Действия по восстановлению:

1. Измерить «долю процессных накладных расходов»: время на церемонии / время на продуктивную работу. Если превышает 20%, процесс слишком тяжёлый
2. Пересмотреть, какой уровень SENAR (Базовая/Командная/Корпоративная) действительно соответствует потребностям команды — многие команды перестарались с внедрением
3. Автоматизировать всё, что можно автоматизировать — ручных церемониальных шагов должно быть минимум
4. Применить собственный принцип SENAR: «Принуждение вместо церемонии» — если шаг можно автоматизировать, он не должен быть ручной церемонией

Профилактика:

- Начинать с SENAR Core и повышать уровень только когда конкретные болевые точки оправдывают дополнительный процесс
- У каждой церемонии должен быть определённый таймбокс
- У каждого ручного шага должно быть обоснование: «какой отказ он предотвращает?» Если ответ неясен — убрать шаг
- Периодически проводить «аудит процесса» — все ли шаги по-прежнему оправдывают свою стоимость?

OF-5: Недостаточный процесс (вечный L2)

Описание: Команда принимает SENAR Core (уровень зрелости 2) и никогда не развивается дальше. Задачи существуют, Стартовый шлюз и Финишный шлюз работают, FPSR и DER отслеживаются — но нет независимой верификации, нет дисциплины ведения базы знаний, нет координации через федерацию. Команда получает минимальную пользу и выходит на плато.

Ранние признаки:

- Один и тот же процесс 6+ месяцев без эволюции
- Рост базы знаний остановился
- DER стабилен, но никогда не улучшается
- Кросс-проектные зависимости управляются хаотично (почта, мессенджеры), а не через федерацию
- Нет ретроспектив инкрементов — метрики собираются, но не анализируются
- Команда говорит «SENAR работает нормально», но не может указать на конкретные улучшения

Действия по восстановлению:

1. Провести оценку зрелости по всем 6 измерениям (раздел 12.2) — определить самое слабое
2. Выбрать ОДНО измерение для улучшения в следующем инкременте — не пытаться прыгнуть с L2 на L3 по всем измерениям одновременно
3. Назначить ответственного за улучшение (обязанность Flow Manager)
4. Установить измеримую цель: «Снизить DER с 12% до 8% к концу инкремента»

Профилактика:

- Включить «улучшение процесса» как постоянный пункт в планирование инкремента
- Роль Flow Manager существует именно для движения эволюции процесса — убедиться, что она укомплектована
- Установить явные целевые уровни зрелости со сроками
- Пересматривать модель зрелости SENAR (раздел 12) на каждой ретроспективе

OF-6: Сопротивление Супервайзеров

Описание: Разработчики отказываются принимать роль Супервайзера. Они воспринимают SENAR как деквалификацию: творческая, интересная работа по написанию кода заменяется бюрократической верификацией машинных результатов. Сопротивление может быть тихим (пассивное несоответствие) или громким (активное противодействие).

Ранние признаки:

- Высокий уровень теневого кодирования (PF-1) — разработчики пишут код и ретроактивно создают задачи под него
- Супервайзеры описывают себя как «рецензентов» или «тестировщиков», а не «инженеров»
- Найм на роли Супервайзеров затруднён — кандидаты хотят «настоящие инженерные» позиции
- Опросы настроений команды показывают неудовлетворённость определением роли
- Опытные разработчики уходят на традиционные позиции

Действия по восстановлению:

1. Напрямую обратиться к вопросу идентичности: Супервайзер — это более старшая роль, чем разработчик, а не более низкая
2. Подчеркнуть аспекты роли, требующие более глубоких навыков: архитектурное суждение, проектирование контекста, верификация сложных систем

3. Позволить Супервайзерам выбирать, когда кодить вручную — автономия снижает сопротивление
4. Приводить конкретные примеры того, как роль Супервайзера усиливает индивидуальное влияние (производительность, умноженная на AI)

Профилактика:

- Позиционировать переход как карьерный рост, а не понижение роли
- Обеспечить соответствие компенсации увеличенному масштабу и ответственности роли Супервайзера
- Предоставить Руководство по переходу (Руководство 04) и дать достаточное время на адаптацию
- Создать сообщества практик, где Супервайзеры делятся техниками и отмечают успехи

OF-7: Информационные силосы

Описание: Каждый Супервайзер вырабатывает собственные недокументированные паттерны подготовки контекста, структурирования задач и взаимодействия с AI. Эти паттерны эффективны, но существуют только в голове Супервайзера. Когда он недоступен, другие Супервайзеры не могут воспроизвести его результаты.

Ранние признаки:

- Производительность сильно варьируется между Супервайзерами на однотипных задачах
- Записи в базе знаний скудны или общи — «настоящий» контекст в личных заметках
- Супервайзеры не могут подменять друг друга во время отсутствия
- Документы передачи не содержат «приёмов», делающих сессию продуктивной
- Новым Супервайзерам требуются месяцы для достижения базовой продуктивности

Действия по восстановлению:

1. Провести сессии «контекстного парного программирования»: высокоэффективные Супервайзеры работают совместно с другими, фиксируя свои недокументированные паттерны
2. Преобразовать личные заметки и шаблоны в общие записи базы знаний
3. Стандартизировать шаблоны контекста для типовых задач
4. Включить «обмен знаниями» как явную цель Quality Sweep

Профилактика:

- Метрику Knowledge Capture Rate следует отслеживать по каждому Супервайзеру, а не только в совокупности
- Шаблоны контекста должны быть общими артефактами, а не личными файлами
- Передача сессий должна содержать «что сработало хорошо» для подготовки контекста, а не только статус задач
- Роль Knowledge Engineer существует для предотвращения силосов — убедиться, что она активно практикуется

Технические отказы

TF-1: Смена модели обнуляет базовые показатели

Описание: Модель AI обновляется (новая версия, другой поставщик или изменение дообучения), и все установленные базовые показатели становятся недействительными. Throughput, FPSR, Lead Time и Cost per Task смещаются непредсказуемо. Команда теряет способность обнаруживать процессные проблемы, потому что сигнал тонет в шуме от смены модели.

Ранние признаки:

- Резкий сдвиг всех метрик после обновления модели
- Задачи, ранее имевшие высокий FPSR, начинают проваливаться
- Cost per Task значительно меняется (часто в обе стороны для разных типов задач)
- Шаблоны контекста, которые работали хорошо, теперь дают худшие результаты
- Поведение AI меняется тонко: другой стиль кода, другие предпочтения библиотек, другие паттерны обработки ошибок

Действия по восстановлению:

1. Рассматривать смену модели как «событие сброса базовых показателей» — задокументировать изменение и зафиксировать метрики до и после
2. Прогнать выборку типичных задач на новой модели для установления новых базовых показателей
3. Обновить шаблоны контекста и записи базы знаний, предполагавшие специфическое поведение модели
4. Допустить 1–2 сессии со сниженными ожиданиями по производительности, пока команда калибруется

Профилактика:

- Фиксировать версии моделей в продакшене — не обновлять автоматически

- При оценке новой модели запускать параллельную проверку на репрезентативном наборе задач перед переходом
- Писать шаблоны контекста модель-нейтрально: описывать что вы хотите, а не как модель должна это сделать
- Вести «журнал смены моделей» с указанием версии модели для базовых показателей каждого инкремента

TF-2: Ограничения контекстного окна

Описание: Контекстное окно AI конечно. По мере роста проектов контекст, необходимый для задачи (знание кодовой базы, архитектурные ограничения, связанные паттерны, предыдущие решения), превышает то, что помещается в одну сессию. Задачи дробятся искусственно — не потому, что естественно разделяемы, а потому что AI не может удержать достаточно контекста.

Ранние признаки:

- Задачи разбиваются на подзадачи, не имеющие смысла как самостоятельные единицы
- AI «забывает» инструкции, данные ранее в сессии
- Результаты в конце сессии противоречат решениям начала
- Супервайзеры тратят всё больше времени на повторное объяснение контекста после контрольных точек
- Сложные задачи рефакторинга проваливаются, потому что AI не видит достаточно кодовой базы одновременно

Действия по восстановлению:

1. Реструктурировать инъекцию контекста: приоритизировать высокоценный контекст (архитектурные ограничения, активные паттерны) над низкоценным (полное содержимое файлов)
2. Использовать записи базы знаний как сжатый контекст: 5-строчная запись решения заменяет чтение 500 строк кода
3. Реализовать прогрессивную загрузку контекста: начинать с обзора архитектуры, подгружать детали по необходимости
4. Принять, что некоторые задачи требуют нескольких сессий — проектировать документы передачи для переноса контекста через границу сессий

Профилактика:

- Записи базы знаний — инструменты сжатия контекста. Вкладываться в качественные, лаконичные записи

- Архитектурная документация должна быть структурирована для потребления AI: ясно, сжато, с явными ограничениями
- Проектировать гранулярность задач вокруг естественных границ, а не размера контекстного окна
- Мониторить утилизацию контекстного окна и установить командные соглашения по распределению контекстного бюджета

TF-3: Архитектурно неверный, но проходящий шлюзы код

Описание: AI генерирует код, который проходит все автоматические шлюзы качества (тесты проходят, типы проверяются, линтер чист, сканер безопасности не находит проблем), но архитектурно ошибочен. Он может использовать неправильный паттерн, создавать неуместную связность, обходить устоявшиеся абстракции или вводить зависимость, которая создаст проблемы через 6 месяцев. Автоматические шлюзы проверяют корректность; они не могут проверить суждение.

Ранние признаки:

- Code review всё чаще выявляют проблемы «работает, но неправильно»
- Аналогичная функциональность реализована по-разному в разных частях кодовой базы
- Устоявшиеся абстракции обходятся — AI создаёт новые паттерны вместо использования существующих
- Граф зависимостей растёт в неожиданных направлениях
- Стоимость рефакторинга растёт со временем по мере накопления архитектурного дрейфа

Действия по восстановлению:

1. Усилить Шлюз качества верификации QG-3 явными критериями архитектурного ревью
2. Создать записи «архитектурных ограничений» в базе знаний, инъецируемые в контекст каждой сессии
3. Провести аудит архитектурного здоровья — выявить и задокументировать все устоявшиеся паттерны
4. Рассмотреть архитектурно-специфичные правила линтинга, поддающиеся автоматизации (ограничения зависимостей, нарушения слоёв)

Профилактика:

- Архитектурные решения должны быть задокументированы в базе знаний, а не только в коде

- Шаблоны контекста для каждой области кодовой базы должны содержать инструкции «используй этот паттерн, а не тот»
- Quality Sweep должен включать проверку архитектурной согласованности
- Роль Context Architect критически важна здесь: она обеспечивает получение AI правильных архитектурных ограничений
- Автоматизированные функции архитектурной пригодности (правила зависимостей, проверки слоёв) превращают суждение в принуждение

TF-4: Фантомные зависимости

Описание: На протяжении сессий AI-агенты вводят зависимости (библиотеки, сервисы, вызовы API, общее состояние), которые не отслеживаются явно. Каждая зависимость по отдельности обоснована, но в совокупности создаётся хрупкая система. Ни один человек или документ не охватывает полный граф зависимостей.

Ранние признаки:

- Время сборки растёт без очевидных причин
- Манифест пакетов (package.json, requirements.txt, go.mod) неуклонно разрастается
- Учащаются проблемы «на моей машине работает» — настройка окружения ненадёжна
- Обновление одной зависимости вызывает каскадные отказы
- Никто не может объяснить, зачем была добавлена конкретная зависимость
- Связность сервисов растёт: изменения в одном сервисе требуют изменений в других

Действия по восстановлению:

1. Провести аудит зависимостей: для каждой зависимости найти задачу, которая её ввела, и проверить, нужна ли она до сих пор
2. Создать тип записи «решение о зависимости» в базе знаний — каждая новая зависимость требует задокументированного обоснования
3. Внедрить обнаружение изменений зависимостей в CI — любая новая зависимость вызывает флаг ревью
4. Вычистить неиспользуемые зависимости

Профилактика:

- Добавить обоснование зависимостей в критерии приёмки QG-2: «никаких новых зависимостей без задокументированной причины»
- Отслеживать количество зависимостей как вторичную метрику
- Quality Sweep должен включать ревью зависимостей

- Шаблоны контекста должны перечислять одобренные зависимости и их назначение — AI предпочитает то, о чём знает

TF-5: Бессмысленное покрытие тестами

Описание: AI генерирует тесты, которые достигают высоких метрик покрытия, но не проверяют осмысленное поведение. Тесты зеркалируют реализацию: они проверяют, что код делает то, что код делает, а не что код делает то, что требуют требования. Это особенно коварно, потому что показатель покрытия выглядит прекрасно.

Ранние признаки:

- Покрытие тестами выше 90%, но Defect Escape Rate тоже высок
- Тесты ломаются при изменении реализации, но не при изменении поведения (хрупкие, тесно связанные тесты)
- Описания тестов обобщённые: «должен работать корректно», «обрабатывает входные данные»
- Тесты не покрывают граничные случаи, пути ошибок или граничные условия
- Мутационное тестирование убивает мало мутантов при высоком покрытии строк
- Интеграционные тесты на самом деле являются юнит-тестами с замоканными зависимостями

Действия по восстановлению:

1. Запустить мутационное тестирование для оценки реальной эффективности тестов — покрытие ничего не значит без показателя убийства мутантов
2. Проверять качество тестов в Quality Sweep: тесты верифицируют требования или зеркалируют реализацию?
3. Требовать, чтобы критерии приёмки включали конкретные тестовые сценарии — AI пишет тесты от сценариев, а не от кода
4. Ввести property-based тестирование для критичных путей

Профилактика:

- Критерии приёмки должны определять тестовые случаи: «при X, когда Y, тогда Z» — а не «напиши тесты для этой функции»
- Отслеживать показатель мутационного тестирования наряду с покрытием
- QG-2 должен включать проверку качества тестов, а не только порогов покрытия
- База знаний должна содержать паттерны тестирования: «как мы тестируем функциональность типа X»

- Разделять промпт агента для написания тестов и промпт агента для реализации — тесты должны верифицировать требования, а не зеркалировать код

Итоги

Ни один режим отказа в этом документе не является теоретическим. Каждый наблюдался на практике — либо в AI-нативной разработке, либо в аналогичных паттернах традиционного внедрения Agile/SAFe.

Общая нить всех режимов отказа одна: **когда неудобные части процесса пропускаются, процесс перестаёт работать.** Верификация неудобна. Документация утомительна. Тупики кажутся напрасной тратой времени. Соблюдение шлюзов воспринимается как бюрократия. Но именно это — несущие элементы SENAR: убери их, и структура рухнет.

Лучшая защита — не больше процесса, а лучшие петли обратной связи:

1. **Метрики с перекрёстной проверкой** — ни одна метрика в одиночку не говорит правды
2. **Quality Sweep, проверяющие реальность** — не только дашборды метрик, а реальный код и реальные записи базы знаний
3. **Ретроспективы, задающие трудные вопросы** — не «хороши ли наши цифры?», а «хорошо ли наше программное обеспечение?»
4. **Культура, где документирование неудач ценится** — осведомлённость о Dead End и режимах отказа предотвращает повторение ошибок

Руководство SENAR: Инженерия требований для AI-нативной разработки

Почему требования важнее при работе с AI

В традиционной разработке расплывчатое требование приводит к диалогу: «Вы имели в виду X или Y?» Разработчик спрашивает, уточняет и реализует правильно.

С AI расплывчатое требование приводит к уверенной, правдоподобной реализации — но не того, что нужно. AI не задаёт уточняющих вопросов — он молча выбирает интерпретацию. Код компилируется, проходит тесты, которые AI сам написал (и которые проверяют неправильное поведение), и выглядит корректным при ревью.

Принцип каскада: Дефект в требовании — самый дорогой дефект. Он порождает код, который выглядит правильно, но решает не ту задачу; тесты, которые выглядят правильно, но проверяют не то поведение; документацию, которая выглядит правильно, но описывает не ту систему. Каждый последующий артефакт наследует исходный дефект.

Вот почему первая ценность SENAR — «Контекст важнее кода», и требования — это важнейший контекст.

Три уровня

SENAR определяет три уровня требований. Не все нужны для каждой задачи — глубина зависит от сложности и размера команды.

BR — Бизнес-требование

Что: Потребность уровня заинтересованных сторон, выраженная в бизнес-терминах. **Кто пишет:** Стейкхолдер, Product Owner или Контекстный архитектор. **Где хранится:** Цель инкремента, цель эпика или выделенная запись BR в базе знаний.

Примеры:

- «Пользователи должны иметь возможность сбросить пароль без обращения в поддержку»
- «Система должна обрабатывать заказы в течение 30 секунд при пиковой нагрузке»
- «API должен поддерживать OAuth 2.0 для интеграций с третьими сторонами»

Свойства: Хороший BR:

- Бизнес-ориентирован (без деталей реализации)
- Проверяем (можно определить, удовлетворяет ли система требованию)
- Независим (достижим без предварительного решения другого BR, либо зависимость явно указана)

SR — Системное требование

Что: Требование уровня системы, выведенное из BR. Описывает, что система должна делать, а не как. **Кто пишет:** Контекстный архитектор (Командная+) или Супервайзер (Базовая) (Core). **Где хранится:** Цель истории или выделенная запись SR, связанная с родительским BR.

Примеры (выведены из BR «сброс пароля»):

- «POST /auth/reset-password отправляет ссылку для сброса на email пользователя»
- «Ссылка для сброса действительна 1 час»
- «Пользователь может установить новый пароль по действительному токену сброса»
- «Система логирует все попытки сброса пароля для аудита безопасности»

Свойства: Хороший SR:

- Достаточно конкретен для декомпозиции на задачи
- Прослеживается до родительского BR
- Тестируем на уровне системы (интеграционный или E2E-тест)

TR — Требование к задаче

Что: Требование уровня реализации. Напрямую соответствует цели задачи и критериям приёмки. **Кто пишет:** Супервайзер. **Где хранится:** Поля цели задачи + критериев приёмки.

Пример (выведен из SR «POST /auth/reset-password»):

- **Цель:** Реализовать эндпоинт POST /auth/reset-password
- **Критерии приёмки:**
    1. Возвращает 200 и отправляет email для существующего пользователя
    2. Возвращает 200 для несуществующего пользователя (без утечки информации)
    3. Возвращает 422, если поле email отсутствует
    4. Токен сброса криптографически случаен, 32 байта
    5. Токен хранится в БД в хешированном виде (не открытым текстом)
    6. Ограничение частоты: 3 запроса на email в час

Свойства: Хороший TR (= хорошие критерии приёмки):

- Независимо проверяем (каждый критерий можно протестировать отдельно)
- Содержит хотя бы один негативный сценарий (случай ошибки)
- Описывает поведение, а не реализацию («возвращает 422», а не «используй Pydantic-валидатор»)

Тестовая модель (TM) — мост верификации

Тестовая модель — НЕ четвёртый уровень требований. Это артефакт верификации, выведенный из требований к задаче.

```
BR → SR → TR → [TM] → Код + Тесты
                ↑ требование    ↑ артефакт верификации
```

Что содержит тестовая модель:

- Тестовые случаи для каждого критерия приёмки (что тестировать)
- Тестовые данные (граничные значения, ошибочные входные данные)
- Ожидаемые результаты (как выглядит «пройдено»)
- Метод верификации (автоматический юнит-тест, интеграционный тест, ручная демонстрация, замер)

В AI-нативной разработке AI обычно генерирует тесты вместе с кодом. Опасность: AI выдаёт тесты, которые проходят, но на самом деле не проверяют заявленные критерии приёмки. Тестовая модель — это проверка Супервайзером того, что сгенерированные тесты соответствуют требованиям, а не просто набирают покрытие.

Конфигурация	Дисциплина тестовой модели

Базовая	Неявная — Супервайзер проверяет сгенерированные тесты относительно критериев приёмки
Начальная	Неявная — аналогично Базовой (Core); совмещённая роль Супервайзера выполняет проверку по критериям
Командная	РЕКОМЕНДУЕТСЯ явная проверка — каждый критерий имеет соответствующий тест
Корпоративная/ Регулируемая	ОБЯЗАТЕЛЬНО документируется, прослеживается до TR, независимо проверяема

Почему не уровень требований? Тестовые случаи описывают *как верифицировать*, а не *что система должна делать*. Это следует ISO/IEC 29119 (тестирование ПО), который отделяет проектирование тестов от требований. Превращение TM в уровень требований заставило бы пользователей Базовой (Core) управлять 4 уровнями — накладные расходы, убивающие внедрение.

Свойства качества требований

Свойство	Определение	Базовая	Командная+
Проверяемость	Каждое требование можно протестировать, измерить или продемонстрировать	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
Непротиворечивость	Нет противоречий между требованиями одного или вышестоящего уровня	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО
Достаточность	Набор дочерних требований полностью покрывает родительское	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО
Неизбыточность	Нет дублирующих требований между историями	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО
Прослеживаемость	Каждый TR прослеживается вверх до BR; каждый BR	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО

	декомпозируется вниз до TR		
Повторное использование	Проверенные требования можно параметризировать для аналогичных задач	ДОПУСКАЕТСЯ	РЕКОМЕНДУЕТСЯ

Проверяемость на практике

Плохо: «Система должна быть быстрой» — быстрой по чьим стандартам? Хорошо: «Время ответа API < 200 мс на p95 при 100 одновременных пользователях»

Плохо: «Обработка ошибок должна быть надёжной» — что значит «надёжной»? Хорошо: «Все API-эндпоинты возвращают структурированные ответы об ошибках с HTTP-статусом, кодом ошибки и понятным человеку сообщением»

Плохо: «Интерфейс должен быть удобным для пользователя» — неизмеримо. Хорошо: «Новый пользователь может завершить процесс регистрации менее чем за 2 минуты без посторонней помощи»

Шаблоны критериев приёмки для типовых задач

REST API эндпоинт

1. Возвращает ожидаемый статус-код и тело для валидного ввода
2. Возвращает 401/403 для неавторизованного/запрещённого доступа
3. Возвращает 422 с описательной ошибкой для невалидного ввода
4. Возвращает 404 для несуществующего ресурса
5. Ответ соответствует задокументированной схеме
6. Ограничение частоты применено согласно конфигурации

Миграция базы данных

1. Миграция идемпотентна (безопасно повторяемая)
2. Откатная миграция существует и работает
3. Нет потери данных при миграции
4. Миграция завершается за приемлемое время для объёма продакшен-данных
5. Индексы созданы для новых паттернов запросов

UI-компонент

1. Корректно отображается в целевом диапазоне viewport (320–1920px)
2. Доступен: навигация с клавиатуры, совместимость со скринридером (WCAG AA)
3. Обрабатывает состояния загрузки, пустоты и ошибки
4. Адаптирован к теме/тёмному режиму, если применимо

Интеграция / внешний API

1. Корректно обрабатывает успешный ответ
2. Обрабатывает таймаут (настраиваемый, по умолчанию 5 с)
3. Обрабатывает ответ с ошибкой со структурированным описанием
4. Политика повторных попыток для транзиентных ошибок
5. Учётные данные не зашиты в код (env/config)

Инфраструктура / DevOps

1. Изменение конфигурации обратимо
2. Эндпоинт проверки здоровья обновлён, если применимо
3. Мониторинг/алертинг покрывает изменение
4. Документация обновлена (runbook, архитектурная диаграмма)

Антипаттерны

Вакуум спецификации

Проблема: У задачи есть цель, но нет критериев приёмки. AI производит нечто, что «выглядит правильно», но никто не может это проверить. **Решение:** Каждая задача ОБЯЗАТЕЛЬНО имеет критерии приёмки (QG-0). Нет критериев — нет старта.

Избыточная спецификация

Проблема: 20 критериев приёмки для тривиальной задачи. Накладные расходы превышают ценность. **Решение:** Соотносить глубину критериев со сложностью задачи. Тривиальная: 2–3 критерия. Сложная: 5–8 критериев. Если нужно 15+, задачу следует разделить.

Копирование критериев

Проблема: Одни и те же критерии копируются между задачами без адаптации. «Возвращает 200» повсюду, хотя некоторые эндпоинты должны возвращать 201 или 204. **Решение:** Использовать шаблоны как отправную точку, затем адаптировать. Шаблоны экономят время; слепое копирование создаёт баги.

Непрослеживаемые требования

Проблема: Задачи существуют без связи с причиной их существования. При смене приоритетов никто не знает, какие задачи можно отбросить. **Решение:** Каждая задача связана с историей или BR (правило 9.10). Если не можешь объяснить, зачем задача существует, — вероятно, она не нужна.

Двусмысленные критерии

Проблема: «Система должна правильно обрабатывать граничные случаи.» Какие граничные случаи? Что значит «правильно»? **Решение:** Назвать конкретные граничные случаи. «Система возвращает 409 при дублировании email. Система возвращает 413 при превышении файлом 10 МБ. Система возвращает 429 при превышении лимита запросов.»

Масштабирование по конфигурации

Аспект	Базовая (1–2 пары)	Начальная (1–3 пары)	Командная (3–10 пар)	Корпора (10+)
Глубина иерархии	BR → TR (2 уровня)	BR → TR (2 уровня)	BR → SR → TR (3 уровня)	Полная + тестов
Кто управляет	Супервайзер	Супервайзер (совмещённые роли)	Контекстный архитектор	Контекст архитект ревью
Хранение	Поля задачи	Поля задачи + записи в БЗ	База знаний	Версиони аудируем
Повторное использование	Неформальные паттерны	Неформальные паттерны	Шаблоны базы знаний	Кросс-пр библиоте
Прослеживаемость	РЕКОМЕНДУЕТСЯ	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕ аудиторс
Проверка качества	Самопроверка	Ежемесячный Обзор качества	QG-1	QG-1 + независи

				проверка
Накладные расходы на задачу	~1 мин (написать критерии)	~1–2 мин (написать критерии + лог)	~3 мин (декомпозиция + связи)	~5 мин (п прослеж
Тестовая модель	Неявная (проверка тестов относительно критериев)	Неявная (проверка тестов относительно критериев)	Явная проверка	Формаль документ

Требования-как-код (Корпоративная)

На Корпоративном уровне требования управляются с той же дисциплиной, что и код: версионируются, рецензируются, проверяются CI.

Что это значит

```
requirements/
  BR-001-oauth.md           # Бизнес-требование
  SR-001-google-oauth.md    # Системное требование (связано с BR-001)
  SR-002-token-refresh.md   # Системное требование (связано с BR-001)
```

Каждый файл требования содержит:

- Уникальный ID и заголовок
- Уровень (BR/SR)
- Ссылку на родителя (SR → BR)
- Статус (черновик/утверждено/верифицировано/устарело)
- Историю версий (git log)
- Ссылки на дочерние элементы (BR → список SR, SR → список слагов задач)

Требования к задачам (TR) остаются в трекере задач как цель + критерии приёмки — им не нужны отдельные файлы, потому что они И ЕСТЬ задача.

Проверки CI для требований

Проверка	Что обнаруживает
Обнаружение сирот	BR без дочерних SR; SR без задач

Валидация прослеживаемости	Задачи без ссылок на требования; битые ссылки на родителей
Согласованность статусов	Задачи, ссылающиеся на `устаревшие` требования
Отчёт покрытия	% BR, полностью декомпозированных и реализованных
Анализ влияния изменений	При изменении BR — список всех затронутых SR и задач

Библиотека требований и повторное использование

Проверенные требования (статус: `верифицировано`, успешно использованы в N проектах) становятся записями библиотеки:

```
library/
  patterns/
    rest-api-endpoint.md    # Шаблон: критерии для любого REST-эндпоинта
    db-migration.md         # Шаблон: критерии для любой миграции
    auth-integration.md     # Шаблон: критерии для потоков авторизации
```

Новый проект импортирует паттерн из библиотеки, параметризирует его и получает проверенные критерии приёмки, протестированные на нескольких реализациях. Это напрямую улучшает First-Pass Success Rate — AI получает проверенный боем контекст вместо черновых требований.

Когда внедрять

Требования-как-код добавляют накладные расходы. Они окупаются, когда:

- 10+ пар Супервайзер+AI разделяют требования между проектами
- Регуляторные требования предполагают аудиторский след (ISO 9001, ASPICE, медицинские устройства)
- Кросс-проектное повторное использование требований — стратегическая цель
- Требования меняются часто и нужен анализ влияния

Для SENAR Core и большинства Командных конфигураций требований в трекере задач и базе знаний достаточно.

Руководство SENAR: Внедрение SENAR в устаревших кодовых базах

Большинство софта — не greenfield. У вас 200 000 строк недокументированного кода, а правило 9.11 говорит: «документация, достаточная для понимания назначения модуля AI». Вы не станете документировать всё до начала работы. Вот как внедрить SENAR поэтапно.

Принцип: следующая задача, а не полная модернизация

Не нужно «сенарить всё». Начните со следующей задачи. Задача затрагивает модуль — задокументируйте его. Следующая задача затрагивает другой модуль — задокументируйте тот. Со временем фронт документации продвигается вместе с реальной работой.

Чего НЕ делать: Не устраивайте «спринт документации» на 3 недели. Он произведёт документацию, которую никто не читает. Документация в отрыве от реализации устаревает ещё до завершения.

Фаза 1: Начать с привычек (День 1)

Внедрите 6 привычек быстрого старта немедленно. Они не требуют подготовки кодовой базы:

1. Писать цель + критерии приёмки перед каждой задачей для AI (даже в legacy-коде)
2. Устанавливать границы области действия («изменять ТОЛЬКО этот файл, не рефакторить модуль»)
3. Проверять результат по критериям приёмки, а не по интуиции
4. Документировать тупики (особенно важно в legacy — «пробовали X, не получилось из-за Y»)
5. Запускать тесты
6. Фиксировать знания

Границы области действия (привычка 2) критичны для legacy: AI-агенты попытаются «улучшить» окружающий код, если не оградить их.

Фаза 2: Документировать при контакте (Неделя 1+)

Каждый раз, когда задача затрагивает модуль, добавляйте минимальную документацию:

```
"""
Module: user_auth
Purpose: Handles login, registration, and session management.
Public API:
  - login(email, password) -> Session
  - register(email, password, name) -> User
  - verify_session(token) -> User | None
Dependencies: database (PostgreSQL), redis (session store)
Boundaries: Does NOT handle OAuth (see oauth_provider module).
"""
```

На это уходит 5 минут на модуль. Этого достаточно для выполнения правила 9.11 на уровне SENAR Core. С этого момента задачи AI, затрагивающие этот модуль, требуют меньше явного контекста в цели задачи — docstring обеспечивает его.

Ключевое: пишите для AI, а не для людей. AI не интересует ваша философия дизайна. Ему нужно: что делает модуль, каков публичный интерфейс, с чем он связан и что он НЕ делает.

Фаза 3: Построение базы знаний из племенного знания (Неделя 2+)

В legacy-кодовых базах есть «племенное знание» — то, что люди знают, но не записали. По мере столкновения с ним при работе по SENAR:

- **Тупик (Dead End):** «Нельзя использовать async в модуле авторизации, потому что он зависит от цепочки синхронного middleware» — задокументировать
- **Подводный камень (Gotcha):** «Поле статуса заказа использует целые числа 1–7, а не enum — legacy-миграция так и не была проведена» — задокументировать

- **Решение (Decision):** «Мы используем сырой SQL вместо ORM в модуле отчётов из-за сложных JOIN-запросов» — задокументировать

Всё это становится записями базы знаний для будущих AI-сессий. Каждый задокументированный подводный камень предотвращает один будущий пропущенный дефект ценой $105.

Фаза 4: Связи требований для новой работы (Месяц 1+)

Новые фичи на legacy-кодовых базах часто лишены чётких требований — кто-то говорит «почини ту штуку», и вы разбираетесь. Исследование по SENAR (раздел 6.1) помогает:

1. Начните исследование (ограниченное по времени расследование)
2. Когда поймёте, что нужно сделать, — создайте задачу с целью + критериями приёмки
3. Свяжите задачу с историей или BR — даже если BR звучит как «снизить количество обращений в поддержку по теме X»

Для исправления ошибок в legacy-коде:

- BR: исходная бизнес-потребность, которую нарушает баг (например, «пользователи должны иметь возможность сбрасывать пароли»)
- TR: конкретное исправление с критериями приёмки (например, «POST /reset-password возвращает 200 для несуществующих email — без утечки информации»)

Фаза 5: Шлюзы качества для legacy (Месяц 2+)

QG-0 работает сразу на legacy: каждая задача имеет цель + критерии приёмки перед началом.

QG-2 может потребовать адаптации:

- «CI проходит» требует наличия CI. Если его нет — добавьте минимальный CI как единовременную инвестицию.
- «Тесты проходят» требует наличия тестов. Для непротестированных legacy-модулей минимум: добавить тесты для кода, который вы меняете. Не тестируйте то, что не трогаете.
- «Типы чисты» может не применяться к динамически типизированному legacy. Используйте то, что доступно (mypy для Python, TypeScript strict для миграций с

JS).

QG-3 и QG-4 — это уровни Командная+ и Корпоративная — отложите до тех пор, пока SENAR Core не будет твёрдо отработан.

Что вы получаете

После 3 месяцев SENAR Core на legacy-кодовой базе:

- Фронт документации продвигается вместе с реальной работой (не гниёт в вики)
- База знаний подводных камней, предотвращающая повторные сбои
- Измеримый FPSR, показывающий, улучшается ли качество контекста
- Тупики, экономящие часы за сессию
- Никакого «большого взрыва» документации, никакого нарушения процессов

Антипаттерн: Спринт документации

Проблема: Руководитель говорит: «Давайте потратим 2 недели на документирование всего, прежде чем включать AI.» **Почему не работает:** Документация без контекста реализации абстрактна и мгновенно устаревает. Человек, документирующий модуль X, не работал с ним 6 месяцев — он упустит подводные камни. **Подход SENAR:** Документировать при контакте. Тот, кто документирует модуль X, — тот же человек, который прямо сейчас пишет задачу в этом модуле. Он знает ровно то, что нужно знать AI, потому что только что это выяснил.

Руководство SENAR: Полный цикл одной задачи

Это не диаграмма. Это одна задача, от требования до завершения, именно так, как это происходит на практике. Задача: добавить эндпоинт сброса пароля в REST API.

Стек: Python 3.14, FastAPI, PostgreSQL, pytest. Конфигурация: SENAR Core (одиночный Супервайзер).

---

1. Требование

В бэклоге продукта есть бизнес-требование:

> ***BR-042:*** *Пользователи должны иметь возможность сбрасывать пароли без обращения в поддержку.*

Супервайзер декомпозирует его в запись задачи:

```
task: impl-password-reset
story: user-authentication
goal: >
  Реализовать эндпоинты POST /auth/password-reset/request и
  POST /auth/password-reset/confirm, позволяющие пользователям
  сбрасывать пароль через токен с ограниченным сроком действия.
acceptance_criteria:
  1. POST /auth/password-reset/request с валидным email возвращает
202 и создаёт токен сброса
  2. POST /auth/password-reset/request с неизвестным email возвращает
202 (без утечки информации)
  3. Токен сброса истекает через 30 минут
  4. POST /auth/password-reset/confirm с валидным токеном + новым
паролем обновляет пароль и аннулирует токен
  5. POST /auth/password-reset/confirm с истёкшим или невалидным
токеном возвращает 400
  6. POST /auth/password-reset/confirm с паролем короче 8 символов
возвращает 422
```

```
role: supervisor
complexity: M
```

Затраченное время: ~2 минуты.

---

2. Проверка Стартового шлюза (QG-0 в Стандарте)

Перед переводом задачи в статус `active` система проверяет критерии Шлюза качества контекста:

Критерий	Статус
Цель непустая	**ПРОЙДЕН** — цель описывает оба эндпоинта и их назначение
Критерии приёмки существуют	**ПРОЙДЕН** — 6 критериев, каждый тестируем независимо
Связь с историей установлена	**ПРОЙДЕН** — привязано к `user-authentication`
Роль назначена	**ПРОЙДЕН** — `supervisor`

Стартовый шлюз (QG-0): **ПРОЙДЕН**. Задача переходит в `active`.

---

3. Промпт для AI

Супервайзер открывает терминал и даёт AI-агенту такой промпт:

```
Implement password reset for our auth module.

Goal: POST /auth/password-reset/request and POST /auth/password-
reset/confirm
endpoints. Request creates a time-limited token. Confirm validates the
token
and updates the password.

Acceptance criteria:
1. POST /request with valid email → 202, creates reset token
2. POST /request with unknown email → 202 (no information leak)
```

```
3. Token expires after 30 minutes
4. POST /confirm with valid token + new password → updates password,
invalidates token
5. POST /confirm with expired/invalid token → 400
6. POST /confirm with password < 8 chars → 422

Scope:
- Change ONLY files in src/auth/ and tests/auth/
- Follow the patterns in src/auth/router.py and src/auth/service.py
- Do NOT modify the User model — add a new PasswordResetToken model
- Use the existing get_password_hash() for hashing
- Do NOT add new dependencies without asking

Plan:
1. Create PasswordResetToken model in src/auth/models.py
2. Add reset request/confirm logic in src/auth/service.py
3. Add endpoints in src/auth/router.py
4. Add schemas in src/auth/schemas.py
5. Write tests for all 6 AC
```

Затраченное время: ~1 минута (большая часть уже была в записи задачи).

---

4. Результат работы AI

AI генерирует четыре файла:

`src/auth/models.py` — добавляет модель `PasswordResetToken`:

```
class PasswordResetToken(Base):
    __tablename__ = "password_reset_tokens"

    id: Mapped[int] = mapped_column(primary_key=True)
    user_id: Mapped[int] = mapped_column(ForeignKey("users.id"))
    token: Mapped[str] = mapped_column(String(64), unique=True,
index=True)
    expires_at: Mapped[datetime] = mapped_column()
    used: Mapped[bool] = mapped_column(default=False)
```

`src/auth/schemas.py` — добавляет схемы запрос/ответ:

```
class PasswordResetRequest(BaseModel):
    email: EmailStr

class PasswordResetConfirm(BaseModel):
    token: str
    new_password: str = Field(min_length=8)
```

src/auth/service.py — добавляет две сервисные функции:

```
from datetime import timezone

async def request_password_reset(db: AsyncSession, email: str) ->
None:
    user = await get_user_by_email(db, email)
    if user is None:
        return  # AC-2: без утечки информации
    token = secrets.token_urlsafe(48)
    expires_at = datetime.now(timezone.utc) + timedelta(minutes=30)
# AC-3
    reset = PasswordResetToken(user_id=user.id, token=token,
expires_at=expires_at)
    db.add(reset)
    await db.commit()


async def confirm_password_reset(
    db: AsyncSession, token: str, new_password: str
) -> None:
    reset = await db.execute(
        select(PasswordResetToken).where(
            PasswordResetToken.token == token,
            PasswordResetToken.used == False,
        )
    )
    reset = reset.scalar_one_or_none()
    if reset is None or reset.expires_at <
datetime.now(timezone.utc):
        raise HTTPException(status_code=400, detail="Invalid or
expired token")
```

```
    user = await db.get(User, reset.user_id)
    user.password_hash = get_password_hash(new_password)
    reset.used = True  # AC-4: аннулировать токен
    await db.commit()
```

src/auth/router.py — добавляет два эндпоинта:

```
@router.post("/password-reset/request", status_code=202)
async def password_reset_request(
    body: PasswordResetRequest, db: AsyncSession = Depends(get_db)
):
    await request_password_reset(db, body.email)
    return {"detail": "If the email exists, a reset link has been
sent"}


@router.post("/password-reset/confirm", status_code=200)
async def password_reset_confirm(
    body: PasswordResetConfirm, db: AsyncSession = Depends(get_db)
):
    await confirm_password_reset(db, body.token, body.new_password)
    return {"detail": "Password updated"}
```

tests/auth/test_password_reset.py — шесть тестов:

```
async def test_request_reset_valid_email(client, db, user):
    """AC-1: валидный email возвращает 202, токен создан."""
    resp = await client.post("/auth/password-reset/request", json=
{"email": user.email})
    assert resp.status_code == 202
    token = await db.execute(select(PasswordResetToken).where(
        PasswordResetToken.user_id == user.id
    ))
    assert token.scalar_one_or_none() is not None


async def test_request_reset_unknown_email(client):
    """AC-2: неизвестный email возвращает 202 (без утечки)."""
```

```
    resp = await client.post("/auth/password-reset/request",
                             json={"email": "nobody@example.com"})
    assert resp.status_code == 202


async def test_token_expires_after_30_minutes(client, db, user):
    """AC-3: токен создан со сроком действия 30 минут."""
    await client.post("/auth/password-reset/request", json={"email":
user.email})
    token = (await
db.execute(select(PasswordResetToken))).scalar_one()
    delta = token.expires_at - datetime.now(timezone.utc)
    assert 29 * 60 < delta.total_seconds() < 31 * 60


async def test_confirm_reset_valid_token(client, db, user,
reset_token):
    """AC-4: валидный токен + новый пароль → пароль обновлён, токен
аннулирован."""
    resp = await client.post("/auth/password-reset/confirm",
                             json={"token": reset_token.token,
"new_password": "newsecure123"})
    assert resp.status_code == 200
    await db.refresh(reset_token)
    assert reset_token.used is True
    await db.refresh(user)
    assert verify_password("newsecure123", user.password_hash)


async def test_confirm_reset_expired_token(client,
expired_reset_token):
    """AC-5: истёкший токен возвращает 400."""
    resp = await client.post("/auth/password-reset/confirm",
                             json={"token":
expired_reset_token.token, "new_password": "newsecure123"})
    assert resp.status_code == 400


async def test_confirm_reset_short_password(client, reset_token):
    """AC-6: пароль короче 8 символов возвращает 422."""
    resp = await client.post("/auth/password-reset/confirm",
                             json={"token": reset_token.token,
```

```
    "new_password": "short"})
    assert resp.status_code == 422
```

---

5. Верификация по критериям приёмки

Супервайзер не просматривает код бегло, чтобы сказать «выглядит нормально». Супервайзер проходит критерии приёмки один за другим:

Критерий	Что проверить	Вердикт
1. Валидный email → 202 + токен создан	`test_request_reset_valid_email` проверяет 202 и наличие токена в БД	**ПРОЙДЕН**
2. Неизвестный email → 202	`test_request_reset_unknown_email` проверяет 202 без ошибки	**ПРОЙДЕН**
3. Токен истекает через 30 мин	`test_token_expires_after_30_minutes` проверяет дельту `expires_at`	**ПРОЙДЕН**
4. Валидное подтверждение → обновляет пароль, аннулирует токен	`test_confirm_reset_valid_token` проверяет `used=True` и смену пароля	**ПРОЙДЕН**
5. Истёкший/невалидный токен → 400	`test_confirm_reset_expired_token` проверяет 400	**ПРОЙДЕН**
6. Короткий пароль → 422	`test_confirm_reset_short_password` проверяет 422	**ПРОЙДЕН**

Каждый критерий имеет соответствующий тест. Каждый тест проверяет ожидаемое поведение, а не реализацию.

Затраченное время: ~3 минуты на чтение тестов и подтверждение соответствия.

---

6. Тупик

Но сначала — кое-что пошло не так в ходе выполнения.

Первая попытка AI использовала `bcrypt` для утилиты хеширования паролей, добавленной вместе с эндпоинтом:

```python
from bcrypt import hashpw, gensalt

def get_password_hash(password: str) -> str:
    return hashpw(password.encode(), gensalt()).decode()
```

При запуске тестов:

```
E   ModuleNotFoundError: No module named '_bcrypt'
```

Пакет `bcrypt` содержит C-расширение, которое не компилируется на Python 3.14 (по состоянию на март 2026). Супервайзер направил AI:

```
bcrypt import fails on Python 3.14. Switch to argon2-cffi with argon2id
hasher.
The existing get_password_hash() already uses argon2 — don't create a new
one,
use the existing function.
```

AI переключился на использование существующей `get_password_hash()` из `src/auth/utils.py` (которая уже использует `argon2-cffi`). Тесты проходят.

Супервайзер создаёт запись тупика:

```
Dead end: bcrypt package — C extension fails to compile on Python 3.14.
Switched to argon2-cffi (argon2id). The existing get_password_hash()
utility
already uses argon2id. Don't add bcrypt to new endpoints.
```

Затраченное время: ~30 секунд на написание записи.

---

7. Проверка Финального шлюза (QG-2 в Стандарте)

Супервайзер запускает Шлюз качества реализации:

```
$ pytest tests/auth/test_password_reset.py -v
======================== 6 passed in 2.14s ========================
```

```
$ mypy src/auth/ --strict
Success: no issues found in 6 source files

$ ruff check src/auth/
All checks passed!
```

Критерий	Статус
CI-пайплайн проходит	**ПРОЙДЕН** — всё зелёное
Все тесты проходят	**ПРОЙДЕН** — 6/6
Типы чисты	**ПРОЙДЕН** — mypy strict, ноль ошибок
Нет новых нарушений линтера	**ПРОЙДЕН** — ruff чист
Критерии приёмки верифицированы	**ПРОЙДЕН** — все 6 проверены в разделе 5
Нет уязвимостей безопасности	**ПРОЙДЕН** — `argon2id` рекомендован OWASP; нет новых зависимостей с CVE

Финальный шлюз (QG-2): **ПРОЙДЕН**. Задача переходит в `done`.

Затраченное время: ~2 минуты (в основном ожидание CI).

> ***Измерение FPSR на практике:*** *Считайте задачу «успехом с первой попытки», если она проходит QG-2 без возврата в статус active. Отслеживайте через инструмент управления задачами: задачи, прошедшие путь planning→active→done = успех; задачи, прошедшие active→done→active→done = переделка. FPSR = успехи / всего завершённых × 100%.*

---

8. Фиксация знаний

Супервайзер создаёт запись знания для решения, принятого при выполнении задачи:

```
type: decision
title: "Using argon2id for all password hashing"
context: >
  Password reset endpoint needed hashing. bcrypt fails on Python
3.14.
  argon2id is the OWASP recommendation and already used in
```

```
  get_password_hash().
  decision: >
    All password operations use argon2-cffi with argon2id algorithm via
    the shared get_password_hash() utility in src/auth/utils.py.
    Do not introduce bcrypt or any other hashing library.
  related_dead_end: "bcrypt C extension incompatible with Python 3.14"
```

Затраченное время: ~30 секунд.

---

9. Фрагмент завершения сессии

При завершении сессии передача фиксирует всё, что нужно будущей сессии (или другому Супервайзеру):

```
Session #12 — 2026-03-22 — 95 minutes — 4 tasks done

Completed:
  - impl-password-reset (this task)
  - impl-email-verification
  - fix-jwt-refresh-race
  - impl-logout-endpoint

Dead ends:
  - bcrypt C extension fails on Python 3.14 → use argon2-cffi

Knowledge entries:
  - decision: argon2id for all password hashing (OWASP, Python 3.14
compat)
  - gotcha: JWT refresh token race condition — need DB-level locking

Next session:
  - impl-oauth-google — Google OAuth integration
  - impl-rate-limiting — rate limit on auth endpoints (5 req/min)

Warnings:
  - OAuth requires new env vars (GOOGLE_CLIENT_ID, GOOGLE_CLIENT_SECRET)
    — add before starting the task
```

Что это стоило

Шаг	Время
Написание цели + критериев приёмки	~2 мин
Написание промпта для AI	~1 мин
AI генерирует код + тесты	~3 мин (ожидание)
Верификация по критериям приёмки	~3 мин
Документирование тупика	~30 сек
Запись знания	~30 сек
Итого	**~10 мин**

Без AI эта задача — два эндпоинта, модель, схемы, шесть тестов, исследование хеширования — заняла бы примерно 25–40 минут ручного кодирования.

Накладные расходы дисциплины SENAR (цель, критерии, верификация, фиксация знаний) добавляют к задаче примерно 4 минуты. AI генерирует остальные 6 минут работы. Итог: полностью протестированный, задокументированный, прослеживаемый эндпоинт менее чем за 10 минут.

Но реальная экономия — не в этой сессии. Она в следующей сессии, когда кто-то (или AI-агент) будет добавлять ещё один эндпоинт авторизации и найдёт:

- Тупик, который убережёт от 15 минут потерянных на bcrypt.
- Решение, указывающее использовать `get_password_hash()`.
- Паттерн критериев приёмки, который можно скопировать для следующего эндпоинта.

Именно эта будущая экономия времени — то, за счёт чего SENAR многократно окупает себя.

Ключевые выводы

1. **Критерии приёмки сделали основную работу.** Шесть критериев, написанных за 2 минуты, определили всю реализацию — что AI сгенерировал, что тесты проверяют, что Супервайзер верифицировал. Без них Супервайзеру пришлось бы читать код и гадать, правильно ли он написан.

2. **Тупик — инвестиция, а не накладные расходы.** Тридцать секунд набора текста экономят следующему человеку (или AI) повторение той же ошибки. В базе знаний со 100+ тупиками это складывается в часы экономии за инкремент.

3. **Финальный шлюз (QG-2) — не церемония.** Это ваш тестовый раннер, чекер типов, линтер и таблица верификации — конкретные инструменты зависят от стека. Это занимает 2 минуты. Если хоть одна строка красная — задача не завершена. Никаких суждений, никаких «наверное, нормально».

4. **Передача делает сессии независимыми.** Сессию #13 может начать другой человек, другой AI-агент или тот же Супервайзер после выходных. Всё необходимое — в передаче: никакого племенного знания, никакого «дай вспомню, на чём я остановился».

Примечание к этому примеру

Этот разбор показывает сценарий чистого прохождения: один тупик, одна повторная попытка, всё зелёное со второго раза. Реальные сессии часто более запутанны — множественные тупики, частичные провалы тестов, критерии приёмки, оказавшиеся двусмысленными в ходе реализации, неожиданные конфликты зависимостей. Этот пример демонстрирует структуру рабочего процесса SENAR, а не типичный уровень сложности. Ваши сессии будут различаться.

Вариации стека

Практики, описанные выше, применимы одинаково вне зависимости от технологического стека. Вот те же паттерны критериев приёмки для распространённых стеков:

> ***Примечание:*** *Вариации стека иллюстрируют паттерны, специфичные для конкретного фреймворка, а не точный перевод примера на Python. Адаптируйте критерии приёмки под соглашения вашего стека.*

Java / Spring Boot

Задача: Реализовать эндпоинт сброса пароля **Критерии приёмки:**

1. POST /api/v1/auth/reset-password принимает {email} → возвращает 200 (без перечисления email)

2. Токен хранится с хешем BCrypt, истекает через 1 час
3. Ограничение частоты: 3 запроса на email в час (@RateLimiter или фильтр)
4. Интеграционный тест с @SpringBootTest и MockMvc
5. Негативный: невалидный токен возвращает 400, истёкший — 410

Go / Gin

Задача: Реализовать эндпоинт сброса пароля **Критерии приёмки:**

1. POST /api/v1/auth/reset-password принимает {email} → возвращает 200
2. Токен хранится с bcrypt.GenerateFromPassword, срок действия 1 час
3. Ограничение частоты: middleware с sync.Map или Redis
4. Табличные тесты с testify
5. Негативный: невалидный/истёкший токен возвращает соответствующий статус

TypeScript / NestJS

Задача: Реализовать эндпоинт сброса пароля **Критерии приёмки:**

1. POST /api/v1/auth/reset-password принимает {email} → возвращает 200
2. Токен хранится с bcrypt.hash, TTL 1 час
3. Декоратор @Throttle() или кастомный guard
4. Jest e2e-тест с supertest
5. Негативный: ValidationPipe отклоняет некорректный ввод

Полные разборы для Java/Spring Boot, Go и TypeScript/NestJS запланированы для SENAR Guide v1.4.

Руководство SENAR: Интеграция с инструментами

SENAR не привязан к инструментам — стандарт никогда не требует конкретного продукта. Но практики по-разному ложатся на каждый инструмент. Эта глава показывает, как применять SENAR с тремя распространёнными AI-инструментами для кодирования, а также как настроить минимальную базу знаний для команд без инфраструктуры.

SENAR с Claude Code

Claude Code — терминальный AI-агент с постоянной памятью, слеш-командами и интеграцией MCP (Model Context Protocol). Из трёх инструментов он ближе всего к модели AI-агента, которую предполагает SENAR.

Цель + критерии приёмки → описание задачи в промпте

Передавайте цель и критерии приёмки прямо в промпте. Claude Code хорошо обрабатывает структурированный текст:

```
Implement POST /orders/{id}/cancel endpoint.

Goal: Allow users to cancel their own orders if the order is in "pending"
status.

Acceptance criteria:
1. POST /orders/{id}/cancel with valid order in "pending" → 200, status
changes to "cancelled"
2. POST /orders/{id}/cancel with order in "shipped" → 409 Conflict
3. POST /orders/{id}/cancel on another user's order → 403
4. Cancelled order cannot be cancelled again → 409
5. Returns 404 for non-existent order ID

Scope: change ONLY src/orders/ and tests/orders/. Follow patterns in
src/orders/router.py.
```

Границы области действия → CLAUDE.md + слеш-команды

`CLAUDE.md` в корне проекта определяет постоянные границы, которые агент читает при каждой сессии:

```
# Boundaries
- NEVER modify files outside src/ and tests/
- NEVER change database migration files directly — use alembic
- Follow existing patterns in router.py, service.py, schemas.py
- Do NOT add dependencies without asking
```

Используйте `/plan` перед сложными задачами, чтобы агент предложил план для проверки перед выполнением.

База знаний → CLAUDE.md + система памяти

Claude Code имеет два слоя знаний:

Слой	Соответствие SENAR	Постоянство
`CLAUDE.md`	Знания уровня проекта (правила, паттерны, границы)	В репозитории, под контролем версий
Файлы памяти (`~/.claude/projects/*/memory/`)	Тупики, решения, подводные камни	Для каждого пользователя, сохраняются между сессиями

При столкновении с тупиком прямо скажите Claude Code:

```
Remember this: bcrypt fails on Python 3.14. Always use argon2-cffi.
```

Он запишет в свою систему памяти. Будущие сессии прочтут это автоматически.

Для команд, использующих MCP, записи знаний можно отправлять в общую базу знаний:

```
Create a knowledge entry: "argon2id for password hashing — bcrypt
incompatible
```

```
with Python 3.14. OWASP recommendation. Use get_password_hash() from
auth/utils.py."
```

Тупики → записи знаний через память или MCP

Немедленная фиксация во время сессии:

```
That approach failed. Remember: SQLAlchemy async sessions cannot use lazy
loading. Must use selectinload() or joinedload() for relationships.
```

QG-0 → навык /plan

Перед началом задачи используйте `/plan` для проверки достаточности контекста:

```
/plan — Review this task before implementation. Check that goal, AC, and
scope
are clear. Flag anything ambiguous.
```

Агент проверяет описание задачи и отмечает недостающие критерии, неоднозначную область действия или невысказанные допущения.

QG-2 → навык /review

После реализации используйте `/review` для запуска Шлюза качества реализации:

```
/review — Check this implementation against the acceptance criteria.
Verify: tests exist for each AC, mypy clean, no linting violations.
```

Дисциплина сессий

Claude Code отслеживает количество вызовов инструментов внутренне. Используйте контрольные точки:

```
/checkpoint — Save progress. 47 tool calls since last checkpoint.
```

Передачи сессий сохраняются в системе памяти или в файле передачи, который агент читает при следующем запуске.

SENAR с Cursor

Cursor — форк VS Code со встроенным AI-чатом, контекстом на уровне файлов через `@-упоминания` и правилами уровня проекта через `.cursorrules`.

Цель + критерии приёмки → .cursorrules + структура промпта

Создайте файл `.cursorrules` с постоянными инструкциями:

```
# Project Rules
- Every implementation MUST have tests for all acceptance criteria
- Follow patterns in existing router/service/schema files
- Never modify database migrations directly
- Use type hints on all function signatures
```

Для каждой задачи структурируйте промпт в чате Cursor одинаково:

```
Goal: [одно предложение]
AC: [нумерованный список]
Scope: [файлы для изменения, файлы НЕ для изменения]
Plan: [упорядоченные шаги]
```

Cursor не принуждает к этой структуре — вы обеспечиваете её привычкой.

Границы области действия → @file-упоминания

Синтаксис `@file` в Cursor управляет контекстом. Используйте его как ограждение:

```
@src/orders/router.py @src/orders/service.py
Implement order cancellation. Follow the patterns in these files.
Only change files in src/orders/ and tests/orders/.
```

Указание конкретных файлов фокусирует внимание модели и снижает вероятность того, что она выдумает паттерны, не соответствующие вашей кодовой базе.

База знаний → .cursorrules + документация проекта

Cursor читает `.cursorrules` при каждом промпте. Используйте для накопленных знаний:

```
# Known Issues
- bcrypt fails on Python 3.14 — use argon2-cffi
- SQLAlchemy async: no lazy loading, use selectinload()
- JWT refresh: must use DB-level locking to prevent race conditions
```

Для более крупных баз знаний ведите директорию `docs/knowledge/` и ссылайтесь на неё:

```
@docs/knowledge/dead-ends.md — Read this before implementing auth-related features.
```

Тупики → файлы знаний или инлайн-документация

Cursor не имеет встроенной системы знаний. Два варианта:

Вариант A: Файлы знаний (рекомендуется) — ведите `docs/knowledge/dead-ends.md` и ссылайтесь через `@`:

```
# Dead Ends

- 2026-03-15: bcrypt C extension fails on Python 3.14 → use argon2-cffi
- 2026-03-18: Celery beat + async → use APScheduler instead
- 2026-03-20: pytest-asyncio auto mode breaks fixtures → use mode=strict
```

Вариант B: Инлайн-комментарии — добавляйте заметки о тупиках рядом с соответствующим кодом:

```
# DEAD END: Do not use bcrypt here — C extension fails on Python 3.14.
# Use argon2-cffi via get_password_hash(). See docs/knowledge/dead-ends.md.
from auth.utils import get_password_hash
```

QG-0 → ручная дисциплина (написать критерии перед промптом)

Cursor не имеет встроенного шлюза. Вы и есть шлюз. Дисциплина:

1. Написать цель + критерии приёмки в промпте чата **до** запроса кода
2. Если ловите себя на набирании «реализуй X» без критериев — остановитесь. Сначала напишите критерии.
3. Держите шаблон задачи (см. Стартовый набор ниже) и вставляйте каждый раз.

QG-2 → функции ревью Cursor + терминал

После генерации кода AI:

1. Используйте вид diff в Cursor для ревью изменений
2. Запустите тесты в терминале: `pytest tests/ -v`
3. Запустите чекер типов: `mypy src/ --strict`
4. Пройдите критерии приёмки один за другим — есть ли для каждого проходящий тест?

SENAR с GitHub Copilot

GitHub Copilot работает через инлайн-подсказки и Copilot Chat (боковая панель VS Code или ревью PR). Из трёх инструментов у него самое слабое управление контекстом.

Цель + критерии приёмки → структурированные комментарии перед кодом

Copilot читает окружающий код и комментарии. Используйте структурированные комментарии как промпт:

```
# TASK: Implement order cancellation endpoint
# GOAL: Allow users to cancel pending orders
# AC-1: POST /orders/{id}/cancel + pending order → 200,
status=cancelled
# AC-2: POST /orders/{id}/cancel + shipped order → 409
# AC-3: POST /orders/{id}/cancel + other user's order → 403
# AC-4: Cancelled order → 409 on re-cancel
# AC-5: Non-existent order → 404
# SCOPE: src/orders/router.py, src/orders/service.py, tests/orders/
```

```
@router.post("/orders/{order_id}/cancel")
async def cancel_order(  # Copilot дополняет отсюда
```

Это более трудоёмко, чем Claude Code или Cursor, но работает: Copilot использует комментарии как контекст для подсказок.

Границы области действия → контекст уровня файлов

Контекст Copilot — преимущественно открытый файл и связанные файлы. Управляйте областью действия:

- Держите нужные файлы открытыми во вкладках (Copilot читает открытые файлы)
- Держите несвязанные файлы закрытыми
- Используйте Copilot Chat с явными ссылками на файлы: «Following the pattern in `src/orders/router.py`, implement...»

База знаний → документация кодовой базы (правило 9.11)

Copilot обучается на вашей кодовой базе. Правило 9.11 (Полнота документации) напрямую улучшает результаты Copilot:

- Docstrings на каждой публичной функции → Copilot генерирует согласованные новые функции
- Аннотации типов повсюду → Copilot генерирует типизированный код
- `README.md` в каждом модуле → Copilot понимает назначение модуля

```
"""Order service — handles order lifecycle operations.

Patterns:
- All mutations go through service layer, never router
- Use get_or_404() for entity lookup
- Status transitions validated via Order.can_transition_to()
"""
```

Тупики → файлы ADR или документирующие комментарии

Используйте Architecture Decision Records в `docs/adr/`:

```
# ADR-007: Password Hashing Algorithm

## Status: Accepted

## Context
Need password hashing for auth module. bcrypt C extension fails on
Python 3.14.

## Decision
Use argon2-cffi with argon2id algorithm via shared
get_password_hash() utility.

## Consequences
- All password operations use one utility function
- argon2-cffi must remain in dependencies
- Do not add bcrypt as alternative
```

Copilot читает файлы ADR, когда они открыты, и Copilot Chat можно направить к ним явно.

QG-0 → ручная дисциплина (перед реализацией)

Ручная дисциплина — написать цель + критерии приёмки в файле трекера задач или issue до начала работы с AI. Copilot не имеет встроенного шлюза на этапе до реализации; используйте TASK_TEMPLATE.md или шаблоны issue как механизм контроля.

1. Написать цель + критерии приёмки в GitHub issue или `TASK_TEMPLATE.md` **до** открытия редактора
2. Если ловите себя на промптинге Copilot без критериев — остановитесь. Сначала напишите критерии.
3. Держите шаблон задачи (см. Стартовый набор ниже) и заполняйте каждый раз.

QG-2 → шаблон PR + CI-пайплайн + ревью PR

Поскольку Copilot интегрирован с GitHub, используйте шаблоны PR как пост-реализационный шлюз:

```
<!-- .github/PULL_REQUEST_TEMPLATE.md -->
## Task
```

```
- [ ] Goal defined
- [ ] Acceptance criteria listed below
- [ ] Scope boundaries stated

## Acceptance Criteria
1.
2.
3.

## Verification
- [ ] Each AC has a passing test
- [ ] mypy/tsc clean
- [ ] No new linting violations

## Knowledge
- Dead ends encountered:
- Decisions made:
```

PR нельзя открыть без заполнения. Рецензенты проверяют по нему.

Кроме того, естественный QG-2 Copilot включает CI-пайплайн:

1. GitHub Actions запускает тесты, чекер типов, линтер
2. Copilot Chat (режим ревью PR) проверяет diff
3. Рецензент-человек проверяет соответствие критериев тестам

Это менее оперативно, чем верификация внутри сессии Claude Code, но работает в сочетании с шаблоном PR.

Базовая база знаний (без инфраструктуры)

Вам не нужна база данных, вики или специализированный инструмент. Достаточно трёх файлов в репозитории:

```
docs/
  knowledge/
    dead-ends.md      — одна строка на тупик
    decisions.md      — один абзац на решение
    gotchas.md        — одна строка на подводный камень
```

dead-ends.md

```
# Dead Ends

Approaches that failed. Check here before trying something new.

- 2026-03-15: bcrypt — C extension fails on Python 3.14. Use argon2-cffi.
- 2026-03-18: Celery beat with async workers — event loop conflicts. Use APScheduler.
- 2026-03-20: pytest-asyncio auto mode — breaks fixtures with scope=session. Use mode=strict.
- 2026-03-21: SQLAlchemy lazy loading in async — not supported. Use selectinload().
- 2026-03-22: Redis Sentinel with asyncio — aioredis deprecated. Use redis-py >= 5.0.
```

Формат: `дата: что пробовали — почему не сработало. Что использовать вместо.`

Одна строка. Десять секунд на написание. Экономит часы.

decisions.md

```
# Decisions

Architectural and technical decisions with context.

## Password Hashing: argon2id
**Date:** 2026-03-15
**Context:** Need password hashing for auth module.
**Decision:** Use argon2-cffi with argon2id via `get_password_hash()` in `src/auth/utils.py`.
**Reason:** OWASP recommendation, Python 3.14 compatible, bcrypt fails (see dead-ends.md).

## Task Queue: APScheduler over Celery
**Date:** 2026-03-18
```

```
**Context:** Need periodic tasks (token cleanup, report generation).
**Decision:** Use APScheduler with AsyncIOScheduler.
**Reason:** Celery beat has event loop conflicts with async FastAPI.
APScheduler runs in-process.
```

Формат: заголовок, дата, контекст (одно предложение), решение (одно предложение), причина (одно предложение).

gotchas.md

```
# Gotchas

Things that are technically correct but surprising. Not bugs — just traps.

- SQLAlchemy: `session.refresh()` after commit or you get stale data in tests
- FastAPI: `Depends()` in test overrides must match the exact function, not just signature
- pytest: fixture scope=session + async requires `event_loop` fixture override
- Docker: `COPY requirements.txt .` before `COPY . .` — layer cache invalidation order matters
- PostgreSQL: `SERIAL` vs `GENERATED ALWAYS AS IDENTITY` — use IDENTITY for new tables
```

Формат: одна строка. Контекст: поведение. Это всё.

Почему это работает

AI-агенты читают ваш репозиторий. Когда `docs/knowledge/dead-ends.md` находится в репозитории, каждый промпт для AI имеет к нему доступ — будь то автоматическое чтение файлов Claude Code, `@file`-упоминания Cursor или контекст открытых вкладок Copilot.

База знаний версионируется, поддерживает поиск и растёт с каждой сессией. Никакого внешнего инструмента не требуется.

Стартовый набор SENAR

Готовые к копированию шаблоны для немедленного использования.

TASK_TEMPLATE.md

```
# Task: [slug]

## Goal
[Одно предложение: что должно быть сделано]

## Acceptance Criteria
1. [Первый критерий — тестируется независимо]
2. [Второй критерий]
3. [Третий критерий]

## Requirement Link
Story: [story-slug]
BR: [ID бизнес-требования, если применимо]

## Scope
- Change ONLY: [директории/файлы]
- Do NOT change: [защищённые файлы/директории]
- Follow patterns in: [файлы-образцы]

## Plan
1. [Первый шаг]
2. [Второй шаг]
3. [Третий шаг]

## Notes
[Контекст, тупики, которых следует избегать, связанные записи базы знаний]
```

.github/PULL_REQUEST_TEMPLATE.md

```
## Summary
[Одно предложение: что делает этот PR]

## Acceptance Criteria
- [ ] AC-1: [критерий из задачи]
- [ ] AC-2: [критерий из задачи]
- [ ] AC-3: [критерий из задачи]

## Verification
- [ ] All AC have corresponding tests
- [ ] Tests pass locally
- [ ] Type checker clean (mypy/tsc/pyright)
- [ ] Linter clean
- [ ] No new dependencies added (or justified below)

## Dead Ends Encountered
<!-- Подходы, которые не сработали при выполнении задачи. Перенести в
docs/knowledge/dead-ends.md -->
- None / [описание]

## Decisions Made
<!-- Архитектурные или технические решения. Перенести в
docs/knowledge/decisions.md -->
- None / [описание]

## Knowledge Entries
<!-- Подводные камни, паттерны или неочевидное поведение,
обнаруженное в процессе -->
- None / [описание]
```

docs/knowledge/README.md

```
# Knowledge Base

This directory contains accumulated project knowledge from SENAR
sessions.

## Files
```

```
| File | Purpose | Format |
|------|---------|--------|
| `dead-ends.md` | Approaches that failed | One line per entry |
| `decisions.md` | Architectural/technical decisions | One paragraph
per entry |
| `gotchas.md` | Surprising but correct behaviors | One line per
entry |

## How to Use

**Before starting a task:** Check `dead-ends.md` for the area you're
working in.

**During a task:** When an approach fails, add one line to `dead-
ends.md`.
When you make an architectural choice, add a paragraph to
`decisions.md`.
When you discover something surprising, add a line to `gotchas.md`.

**After a task:** Review what you learned. If any dead ends,
decisions, or
gotchas were captured in the PR template, move them to the
appropriate file.

## For AI Agents

If you are an AI coding agent reading this: check `dead-ends.md`
before
attempting any approach in a domain that has entries. Check
`gotchas.md`
for the libraries and frameworks you are about to use. Check
`decisions.md`
for architectural context.
```

Варианты базы знаний

SENAR не предписывает конкретный инструмент для базы знаний. Распространённые подходы по размеру команды:

- **Solo/Базовая**: Markdown-файлы в репозитории проекта (например, `docs/knowledge/`)
- **Начальная (1–3 пары)**: Вики проекта, Notion или структурированные YAML/JSON в репозитории
- **Командная (3–10)**: Специализированный инструмент с поиском — CouchDB+Meilisearch, Confluence, Linear docs
- **Корпоративная (10+)**: Федеративные базы знаний по проектам с кросс-проектным поиском

Ключевое требование: знания должны быть доступны для поиска AI-агентами в будущих сессиях.

Выбор инструмента

Практика SENAR	Claude Code	Cursor	GitHub Copilot
Цель + критерии приёмки	Текст промпта	Текст промпта	Структурированные комментарии
Границы области действия	CLAUDE.md + промпт	.cursorrules + @file	Открытые вкладки + промпт
База знаний	Память + MCP	.cursorrules + docs/	docs/ + ADR
Фиксация тупиков	Записи памяти	Файлы знаний	Файлы ADR
Контроль QG-0	навык /plan	Ручная дисциплина	Шаблон задачи / issue
Контроль QG-2	навык /review	Терминал + вид diff	Шаблон PR + CI-пайплайн
Передача сессии	Система памяти	Файл передачи	Описание PR
Сохранение контекста	Автоматическое (память)	Ручное (@file)	Ручное (открытые файлы)

Инструмент не создаёт процесс. Процесс делает инструмент полезным. Дисциплинированный Супервайзер с обычным текстовым редактором обойдёт Супервайзера с лучшим AI-инструментом, но без критериев приёмки.

Руководство SENAR: Конфигурация агентов на практике

Эта глава связывает требования раздела 5 Стандарта (Инструментирование агентов) с практикой. Как писать операционные скрипты, определять профили агентов, управлять изменениями скриптов и применять эти концепции в разных AI-инструментах.

Анатомия операционного скрипта

Операционный скрипт — структурированная инструкция на естественном языке. В отличие от кода, скрипты интерпретирует AI-агент, поэтому они должны быть однозначными, самодостаточными и проверяемыми.

Минимальная структура скрипта

```
Script: commit
Trigger: /commit command or explicit user request

Preconditions:
- Changes exist in the working tree
- No failing tests

Algorithm:
1. Run git status to see all changes
2. Run git diff to review staged and unstaged changes
3. Analyze changes and draft a commit message:
   - Summarize the nature of changes (feature, fix, refactor...)
   - Keep message under 72 characters for the subject line
4. Stage relevant files (prefer specific files over git add -A)
5. Create the commit
6. Run git status to verify success

Postconditions:
- Commit exists in local repository
- Working tree is clean for committed files
```

```
Outputs:
- Commit hash
- Summary of what was committed
```

Что делает скрипт хорошим

1. **Явные предусловия.** Если скрипт предполагает что-то истинным — укажите это. «Тесты проходят» — это предусловие, а не допущение.
2. **Нумерованные шаги с точками принятия решений.** Когда шаг 3 зависит от результата шага 2, скажите об этом. «Если тесты не прошли, остановиться и сообщить. Если тесты прошли, перейти к шагу 4.»
3. **Определённые результаты.** Что производит скрипт? Коммит? Отчёт? Смену статуса? Будьте конкретны.
4. **Задокументированные граничные случаи.** «Если изменений нет, сообщить об этом и выйти без создания пустого коммита.»
5. **Никаких платформоспецифичных вызовов.** Пишите «Запустить набор тестов», а не «Запустить `npm test`». Скрипт должен работать вне зависимости от стека проекта.

Антипаттерны

- **Расплывчатые инструкции:** «Убедиться, что всё хорошо» — непроверяемо
- **Неявные знания:** «Использовать стандартный подход» — какой стандартный?
- **Отсутствие обработки ошибок:** Нет указаний на случай провала шагов
- **Избыточная связность:** Один скрипт, который планирует, реализует и проверяет

Определение профилей агентов

Профиль агента ограничивает его действия. В этом и смысл: агент не выходит за рамки назначенной функции.

Профиль Generator (основная разработка)

```
Profile: Generator
Purpose: Implement code changes for a specific Task

Scripts available:
- implement: Write code to satisfy Task acceptance criteria
- debug: Investigate and fix failures
- commit: Create version-controlled commits
- test: Run and verify test results

Access:
- Read/write: source code, tests, configuration
- Read: task definitions, knowledge base
- Write: task status updates, knowledge entries
- NO access to: deployment, infrastructure, other projects' code

Constraints:
- Changes must stay within Task scope boundaries
- Commits must be atomic (one logical change)
- Must not modify files outside defined scope
```

Профиль Reviewer (независимая верификация)

```
Profile: Reviewer
Purpose: Independently verify code changes against acceptance criteria

Scripts available:
- review: Check code against acceptance criteria and standards
- security-audit: Scan for security issues
- report: Generate review findings

Access:
- Read-only: source code, tests, configuration, task definitions
- Write: review comments, findings
- NO access to: code modification, commits, deployment

Constraints:
- Cannot modify any code being reviewed
- Findings must reference specific acceptance criteria
```

```
- Must check for AI-specific issues (hallucinated APIs, self-validating
tests)
```

Минимально жизнеспособные профили

На уровне SENAR Core необходимо как минимум:

1. **Generator** — для разработки
2. **Reviewer** — для независимой верификации

Ключевая идея: даже если один AI-агент переключается между профилями, переключение создаёт когнитивную границу. Reviewer не может модифицировать код — значит, он оценивает, а не исправляет.

Управление изменениями скриптов

Изменение скрипта меняет поведение AI-агента в продакшене. Относитесь к этому так же строго, как к изменениям продакшен-кода.

Рабочий процесс изменения скрипта

```
1. Выявить потребность в изменении
   ↓
2. Составить черновик изменения с обоснованием
   ↓
3. Ревью (коллегиальное или самостоятельное, в зависимости от
конфигурации)
   ↓
4. Тестирование на изолированном проекте (Team+)
   ↓
5. Развёртывание в целевой проект
   ↓
6. Мониторинг эффективности (FPSR, частота ошибок)
   ↓
7. Запись решения в базу знаний
```

Тестирование изменений скриптов

Перед развёртыванием изменения скрипта во все проекты:

1. **Выберите типичную задачу** — подберите задачу, аналогичную тем, которые скрипт будет обрабатывать
2. **Выполните задачу со старым скриптом** — зафиксируйте FPSR и проблемы
3. **Выполните аналогичную задачу с новым скриптом** — сравните результаты
4. **Проверьте регрессии** — не сломало ли изменение существующее поведение?

Откат

Каждое изменение скрипта должно быть обратимым:

- Скрипты находятся под контролем версий — `git revert` работает
- Ведите идентификатор версии в каждом скрипте (дата или semver)
- Документируйте, на что обращать внимание после отката (задачи в процессе могут быть затронуты)

Примеры для конкретных инструментов

Claude Code

Claude Code использует `CLAUDE.md` для поведенческого контракта и `.claude/skills/` для операционных скриптов.

Поведенческий контракт (`CLAUDE.md`):

```
# Project Rules
- NEVER commit without explicit permission
- NEVER modify files outside src/ and tests/
- Run tests before every commit
- Ask before installing new dependencies
```

Операционный скрипт (`.claude/skills/review.md`):

```
# Review Skill
Trigger: /review command

Steps:
1. Read the current git diff
```

```
2. For each changed file, check:
   a. Does the change match the stated Task goal?
   b. Are there any hardcoded values that should be configurable?
   c. Are error cases handled?
   d. Do tests cover the acceptance criteria, not just code paths?
3. Check for AI-specific issues:
   a. Hallucinated APIs or methods
   b. Non-existent package references
   c. Self-validating test patterns
4. Report findings with specific line references
```

Переключение профилей: Claude Code не обеспечивает разрешения профилей на нативном уровне. Реализуйте через правила CLAUDE.md: «При выполнении ревью вы находитесь в режиме Reviewer. НЕ модифицируйте код. Только сообщайте о находках.»

OpenAI ChatGPT / GPT-4

Поведенческий контракт (системный промпт или пользовательские инструкции):

```
You are working in Generator mode. You may:
- Read and write source code and tests
- Create commits with descriptive messages
- Search the codebase

You may NOT:
- Modify deployment configuration
- Access external APIs
- Create files outside the project directory
```

Операционные скрипты (структурированные промпты или сохранённые рабочие процессы): ChatGPT / GPT-4 не имеет нативного механизма навыков/скриптов. Скрипты реализуются как структурированные шаблоны промптов, которые Супервайзер предоставляет в начале каждой задачи.

Cursor

Поведенческий контракт (`.cursorrules`):

```
Rules:
- Always run type checking before suggesting changes as complete
```

```
- Never modify files not mentioned in the current task
- Prefer editing existing files over creating new ones
```

Операционные скрипты (`.cursor/prompts/` или ручной вызов): Cursor поддерживает пользовательские промпты, работающие аналогично операционным скриптам. Храните их под контролем версий вместе с проектом.

Практический рабочий процесс: жизненный цикл изменения скрипта

Сценарий: Ваш скрипт «implement» порождает код, который часто содержит ошибки типов. Вы хотите добавить шаг «запуск проверки типов» перед отметкой реализации как завершённой.

Шаг 1: Задокументировать проблему

В базе знаний:

```
Type: observation
Title: Implementation script produces type errors
Body: In the last 5 tasks, 3 had type checking failures caught at CI
  (QG-2). Adding a type check step to the implement script should
  catch these earlier and improve FPSR.
```

Шаг 2: Составить черновик изменения

Добавить в скрипт implement после шага «написать код»:

```
5. Run the project's type checker
6. If type errors exist:
   a. Fix them
   b. Re-run type checker
   c. Repeat until clean
7. Continue to tests
```

Шаг 3: Ревью

Для Базовой (Core): самостоятельное ревью — имеет ли добавление смысл? Не конфликтует ли с другими шагами?

Для Командная+: коллегиальное ревью — другой Супервайзер проверяет изменение скрипта. Уточнить: ясен ли новый шаг? Обработаны ли граничные случаи (что если в проекте нет чекера типов)?

Шаг 4: Тестирование (Командная+)

Выполнить одну задачу с новым скриптом на некритичном проекте. Выполнился ли шаг проверки типов корректно? Обнаружил ли он ошибки типов до CI? Не добавил ли неразумного времени?

Шаг 5: Развёртывание

Закоммитить изменение скрипта. Обновить идентификатор версии.

Шаг 6: Мониторинг

На протяжении следующих 5–10 задач отслеживать:

- Снизились ли ошибки проверки типов на CI (QG-2)?
- Значительно ли изменилось время выполнения задач?
- Есть ли неожиданные побочные эффекты?

Шаг 7: Запись

```
Type: decision
Title: Added type checking step to implement script
Body: After 3/5 tasks had type errors at CI, added explicit type check
  step to implement script. First 5 tasks after change: 0/5 type
  errors at CI. ~2 min added per task, saves ~10 min rework.
```

Итоги

Концепция	Базовая	Командная+
Профили агентов	Generator + Reviewer (РЕКОМЕНДУЕТСЯ)	Все 5 с контролем прав доступа (ОБЯЗАТЕЛЬНО)

Операционные скрипты	Под контролем версий, самостоятельное ревью	Рецензируются, тестируются, ведётся реестр
Изменения скриптов	Трактуются как изменения процесса	Тестируются на изолированном проекте
Откат	Контроль версий обеспечивает	Реестр + явная процедура отката
Аудиторский след	Записи в базе знаний	Формальный журнал изменений

Руководство SENAR: Онбординг новых членов команды

Порядок чтения

1. **День 1 (30 мин):** Прочитайте SENAR Core — изучите 8 правил, 2 шлюза качества, чеклист верификации
2. **День 1 (15 мин):** Прочитайте Руководство Quickstart — запомните 6 привычек
3. **День 2 (30 мин):** Прочитайте Руководство Worked Example — посмотрите полный цикл задачи от начала до конца
4. **День 2 (20 мин):** Прочитайте Руководство AI Review Checklist — ваш справочник по верификации
5. **Неделя 1:** Прочитайте Руководство Tool Guides — настройте AI-инструмент в соответствии с практиками SENAR

Протокол первой задачи

Ваша первая задача по SENAR должна быть:

- **Небольшая** — исправление бага или простая фича (< 1 часа)
- **В паре с опытным Супервайзером**, который проверяет ваши шлюзы
- **С полной церемонией** — выполняйте каждый шаг явно, даже если кажется медленным

Чеклист для первой задачи:

- [ ] Напишите цель и критерии приёмки до написания кода
- [ ] Определите, что входит в область задачи и что НЕ входит
- [ ] Выполните чеклист Стартового шлюза (QG-0)
- [ ] Работайте с AI, фиксируя тупики (> 15 мин)
- [ ] Верифицируйте каждый критерий приёмки с доказательствами
- [ ] Выполните чеклист Шлюза готовности (QG-2)
- [ ] Задокументируйте тупики, решения или подводные камни

График освоения

Неделя	Фокус	Ожидаемый результат
1	Правила Базовой конфигурации + 2–3 задачи под наблюдением	Понимание рабочего процесса
2	Самостоятельные задачи с ревью после	Уверенность в работе со шлюзами качества
3	Полная самостоятельность + интернализированный чеклист	Осознанное отслеживание FPSR
4	Пополнение базы знаний + взаимная проверка	Полноправный участник команды

Типичные ошибки новых участников

1. **Пропуск негативного сценария** — Правило 1 требует хотя бы одного "что НЕ должно происходить"
2. **Нефиксация тупиков** — Даже "Я попробовал X, и это не сработало" — ценная информация
3. **Отношение к чеклисту как к необязательному** — Всегда начинайте с уровня Standard
4. **Путаница между исследованием и реализацией** — Исследование свободно, но коммит требует задачи

О чём НЕ нужно беспокоиться

- Метрики (FPSR, Dead End Rate) — первые 10 задач — это калибровка, а не оценка
- Идеальные записи в базе знаний — любая запись лучше, чем ничего
- Скорость — процесс становится быстрее с практикой, а не за счёт пропуска шагов

Справочник SENAR: Полный глоссарий

Алфавитный перечень всех терминов SENAR. Основные термины нормативно определены в Стандарте SENAR, Section 3.

Термин	Определение
AI-агент	Программная система на базе LLM, генерирующая инженерные артефакты под руководством человека
Поставщик AI-модели	Внешний сервис, обеспечивающий работу AI-агента (например, Anthropic, OpenAI, Google); де-факто поставщик (Standard 3.25)
Версия AI-модели	Конкретная версия/поколение AI-модели; базовые показатели метрик зависят от версии (Standard 3.26)
Архитектор контекста	Роль: проектирует требования как структурированный вход для AI, ведёт прослеживаемость
Библиотека требований	Репозиторий проверенных, повторно используемых требований в базе знаний (Командная+)
Время выполнения (Cycle Time)	Время от начала задачи до завершения (started_at → completed_at)
Время исполнения (Lead Time)	Время от создания задачи до завершения
Доля ручного вмешательства (Manual Intervention Rate)	Метрика: % задач с ручным написанием кода
Запись знания (Knowledge Entry)	Зафиксированное решение, паттерн, подводный камень или тупиковый подход
Поток ценности (Value Stream)	Сквозной путь от запроса клиента до готового ПО
Инженер верификации	Роль: проверяет результаты AI на корректность и безопасность

Инженер знаний	Роль: собирает, курирует и поддерживает базу знаний
Инкремент	Рабочий цикл с фиксированным объёмом, целями и бюджетом
Исследование (Exploration)	Ограниченное по времени исследование без полной формальности задачи
Тупиковый подход (Dead End)	Задокументированный неудачный подход с объяснением, почему от него отказались
Коэффициент захвата знаний (Knowledge Capture Rate)	Метрика: записи знаний / выполненные задачи
Контекст	Информация, предоставляемая AI-агенту: цель, критерии приёмки, ограничения, знания, прослеживаемость
Контрольная точка (Checkpoint)	Действие по сохранению контекста в ходе сессии
Менеджер потока	Роль: задаёт ритм сессий, отслеживает затраты и метрики
Обзор поставки (Delivery Review)	Церемония: демонстрация ПО заинтересованным сторонам
Обход шлюза (Gate Bypass)	Задокументированное исключение при прохождении шлюза качества
Пара «Супервайзер+AI» (Supervisor+AI Pair)	Базовая производственная единица: один супервайзер + AI-агент(ы)
Паттерн требований (Requirement Pattern)	Повторно используемый шаблон критериев приёмки для типовых задач (CRUD, миграция, UI, интеграция)
Планирование инкремента (Increment Planning)	Церемония: определение целей, задач, бюджета, рисков
Предсказуемость затрат (Cost Predictability)	Метрика: фактические затраты / плановые затраты за инкремент
Пропускная способность (Throughput)	Метрика: задачи, выполненные за сессию
Качество на входе (Quality at Input)	Принцип: дефекты в требованиях каскадно распространяются на все последующие артефакты; качество закладывается на входе, а не проверяется на выходе

Ретроспектива инкремента (Increment Retrospective)	Церемония: анализ метрик инкремента с выводами
Сессия	Ограниченный по времени период управляемой работы AI
Синхронизация федерации (Federation Sync)	Церемония: координация нескольких пар по зависимостям
Стоимость задачи (Cost per Task)	Метрика: общие затраты / выполненные задачи, с разбивкой по сложности
Супервайзер	Человек, который направляет AI-агентов, проверяет результат и проводит работу через шлюзы качества
Требование	Задокументированное, проверяемое утверждение о потребности на уровне BR, SR или TR (Standard 3.17)
Требование — Бизнес (BR)	Потребность заинтересованных сторон в бизнес-терминах; источник всех последующих требований (Standard 3.18)
Требование — Системное (SR)	Что система должна делать; выводится из BR (Standard 3.19)
Требование — Задачи (TR)	Требование уровня реализации = цель задачи + критерии приёмки (Standard 3.20)
Иерархия требований (Requirement Hierarchy)	Цепочка декомпозиции: BR → SR → TR; глубина зависит от сложности (Standard 3.21)
Прослеживаемость (Traceability)	Двусторонняя связь: каждое TR ведёт вверх к BR; каждое BR раскрывается вниз до TR(s) (Standard 3.24)
Тип работы (Work Type)	Функциональная категория задачи (dev, arch, QA, docs)
Уровень зрелости (Maturity Level)	Степень внедрения SENAR в организации (L1–L5)
Доля успеха с первого раза (FPSR)	Метрика: % задач, выполненных корректно за один цикл
Доля утечки дефектов (DER)	Метрика: % дефектов, обнаруженных после отметки задачи как выполненной
Завершение сессии (Session End)	Церемония: фиксация метрик, запись передачи, сохранение знаний
Задача (Task)	Атомарная единица отслеживаемой работы с целью и критериями приёмки

История (Story)	Группировка задач, видимая заинтересованным сторонам как результат
Обзор качества (Quality Sweep)	Церемония: периодический глубокий аудит кодовой базы и базы знаний
Начало сессии (Session Start)	Церемония: загрузка контекста, выбор задач, проверка окружения
Федерация	Механизм координации нескольких пар «Супервайзер+AI»
Шлюз качества (Quality Gate)	Автоматизированная контрольная точка; блокирует работу, если критерии не выполнены
WSJF	Приоритизация: стоимость задержки / размер работы (из SAFe)

Справочник SENAR: Коэффициенты масштабирования

Приведённые коэффициенты ориентировочны. Организации ОБЯЗАНЫ адаптировать их под свою предметную область, возможности AI-инструментов и зрелость команды.

Распределение ответственностей по размеру команды

Пар	Супервайзер	Архитектор контекста	Инженер знаний	Менеджер потока	Инженер верификации
1–2	1 (все роли)	— (поглощена)	— (поглощена)	— (поглощена)	— (поглощена)
3–5	3–5	1 (АК+ИЗ)	(покрыта АК)	1 (МП+ИВ)	(покрыта МП)
6–10	6–10	1 выделенный	1 выделенный	1 выделенный	1 выделенный
10–20	10–20	2 (по доменам)	1	2	2
20–50	20–50	3–5	1–2	2–3 (Координаторы федерации)	3–5
50+	50+	5–10	2–3	5+ (вкл. Портфельного менеджера)	5–10

Роли корпоративного уровня

Роль	Соотношение	Когда необходима
Портфельный менеджер	1 на 20–50 пар	Несколько потоков ценности или бюджетный контроль

Главный супервайзер	1 на организацию	Архитектурное управление, стандарты шлюзов качества
Координатор федерации	1 на 10–20 пар	Управление межкомандными зависимостями
Специалист по нормативному соответствию	1 на организацию	Регулируемые отрасли (ISO, PCI, HIPAA)

Обеспечение обзоров качества

Пар	Покрытие инженерами верификации
1–2	Супервайзер проводит самоаудит
3–5	1 инженер верификации, обзоры ротируются между парами
6–10	1 выделенный инженер верификации, полный обзор каждый цикл
10+	1 инженер верификации на 5–10 пар, скоординированное расписание обзоров

Справочник SENAR: Модель эффективности

Методики оценки эффективности внедрения SENAR. Все модели используют **относительные коэффициенты и множители**, а не абсолютные денежные величины — организации подставляют свои числа в любой валюте.

A. Измерения эффективности

Эффективность SENAR измеряется по четырём направлениям:

Направление	Что измеряется	Ключевой показатель
Пропускная способность	Выработка на единицу	Задач на пару «Супервайзер+AI» vs задач на разработчика
Качество	Затраты на предотвращение и обнаружение дефектов	Стоимость раннего обнаружения дефекта vs позднего
Знания	Сохранение накопленного опыта	Доля повторно используемых знаний: записи, которые предотвращают повторение ошибок
Накладные расходы	Доля процессных затрат	Время на церемонии и шлюзы как % от продуктивного времени сессии

B. Модель пропускной способности

B.1 Сравнение производственных единиц

Модель	Производственная единица	Типичный состав
Традиционная	Команда разработки	5–9 инженеров + QA + PM
SENAR	Пара «Супервайзер+AI»	1 супервайзер + AI-агент(ы)

Множитель пропускной способности:

```
T_multiplier = Tasks_per_Pair_per_period / Tasks_per_Developer_per_period
```

Организации РЕКОМЕНДУЕТСЯ измерять этот коэффициент в пилотном проекте, чтобы установить базовую линию. Коэффициент сильно зависит от сложности предметной области, возможностей AI-инструментов, опыта супервайзера и качества контекста.

B.2 Эффективность масштабирования

Традиционное масштабирование: каждый новый разработчик даёт убывающую отдачу (закон Брукса — коммуникационные накладные расходы растут как n^2).

Масштабирование в SENAR: каждая новая пара «Супервайзер+AI» даёт почти линейную отдачу вплоть до предела координации федерации, потому что пары работают независимо, а зависимости отслеживаются программно.

```
Traditional: Effective_capacity = n × Developer_output × (1 -
communication_overhead(n))
SENAR:       Effective_capacity = n × Pair_output × (1 -
federation_overhead(n))
```

Где `federation_overhead(n)` растёт медленнее, чем `communication_overhead(n)`, потому что зависимости отслеживаются программно, а не через совещания.

C. Эффективность качества

C.1 Соотношение затрат на обнаружение дефектов

Чем раньше найден дефект, тем дешевле его исправить. Это соотношение устойчиво в разных организациях:

Точка обнаружения	Относительная стоимость
В процессе генерации AI (та же сессия)	**1×** (базовая линия)
В ходе обзора качества (периодический аудит)	**3–5×**
В ходе приёмочного тестирования	**5–10×**

В продакшене	**10–50×**

ROI шлюзов качества:

```
Gate_ROI = (Defects_caught × Avg_late_detection_cost) /
Gate_operation_cost
```

Организациям РЕКОМЕНДУЕТСЯ измерять количество дефектов на каждом этапе и рассчитывать собственные коэффициенты затрат.

C.2 Эффективность первого прохода

Более высокая доля успеха с первого раза (FPSR) означает меньше переделок:

```
Rework_cost = Tasks_total × (1 - FPSR) × Avg_rework_cost_per_task
Efficiency_gain = Rework_cost_before_SENAR - Rework_cost_after_SENAR
```

FPSR улучшается по мере повышения качества контекста (более чёткие критерии приёмки, богатая база знаний, задокументированные тупиковые подходы).

D. Эффективность знаний

D.1 Повторное использование тупиковых подходов

Каждый задокументированный тупик (Dead End) не даёт будущим супервайзерам повторить ту же ошибку.

```
Dead_End_ROI = Documented_dead_ends × Avg_times_would_be_repeated ×
Avg_exploration_cost
```

Хорошо задокументированные тупики используются повторно почти в 100% случаев — практически каждый из них предотвращает минимум одно повторение в организации.

D.2 Эффект накопления знаний

По мере роста базы знаний качество контекста улучшается, что повышает FPSR и сокращает переделки:

```
Session N:    FPSR = f(KB_size_at_N, Context_quality)
Session N+K:  FPSR' > FPSR  (if KB is maintained and growing)
```

Так возникает эффект сложного процента — каждый инкремент эффективнее предыдущего, пока не наступает плато, на котором большинство типовых паттернов и подводных камней уже задокументированы.

E. Модель накладных расходов

E.1 Коэффициент накладных расходов на процесс

```
Overhead_ratio = Time_on_ceremonies_and_gates / Total_session_time
```

Целевой показатель: накладные расходы < 15% времени сессии для Базовой/Начальной, < 20% для Командной.

Активность	Базовая/Начальная	Командная
Начало сессии	2–5 мин	2–5 мин
Завершение сессии	5–10 мин	5–10 мин
Проверки шлюзов качества	Автоматически (0 мин)	Автоматически (0 мин)
Обзор качества	Периодически (амортизировано)	Периодически (амортизировано)
Синхронизация федерации	Н/Д	5–10 мин на синхронизацию
Планирование инкремента	Амортизировано	1 сессия на инкремент
Ретроспектива	Амортизировано	30–60 мин на инкремент

E.2 Точка безубыточности накладных расходов

Накладные расходы SENAR окупаются, когда экономия от предотвращения дефектов превышает стоимость процесса:

```
Break_even: Gate_cost + Ceremony_cost < Defects_prevented × Avg_defect_cost
```

Организациям РЕКОМЕНДУЕТСЯ рассчитывать этот показатель по итогам 3 инкрементов с измеренными данными.

F. Система сравнения

F.1 Традиционная команда vs команда SENAR

Для сравнения эффективности поставки при одинаковом объёме:

Метрика	Традиционная команда	Команда SENAR	Как измерять
Численность	N разработчиков + QA + PM	M супервайзеров + роли поддержки	Подсчёт
Пропускная способность	Задач за период	Задач за период	Одинаковая гранулярность задач
Доля дефектов	Дефектов на 100 задач	Дефектов на 100 задач	Одинаковый метод подсчёта
Время исполнения	Требование → продакшен	Требование → продакшен	Одинаковые контрольные точки
Доля переделок	% задач, требующих переделки	% задач, требующих переделки (1 - FPSR)	Одинаковое определение
Сохранение знаний	Bus-фактор, время онбординга	Покрытие базы знаний, время онбординга	Измеряемое

F.2 Критерии принятия решения

SENAR эффективнее, когда:

- AI-инструменты способны генерировать большую часть артефактов реализации в данной области

- Множитель пропускной способности (B.1) превышает коэффициент накладных расходов (E.1) — чистый выигрыш в продуктивности
- Экономия от предотвращения дефектов (C.1) превышает затраты на шлюзы — чистый выигрыш в качестве
- Накопление знаний (D.2) даёт эффект сложного процента со временем

SENAR менее эффективен, когда:

- Предметная область плохо подходит для AI-генерации (исследования, строго регламентированные ручные процессы)
- Затраты на AI-инструменты превышают выигрыш от роста пропускной способности
- Организация не может вложиться в инструментальную инфраструктуру (трекер задач, CI/CD, база знаний)
- Команда слишком мала, чтобы процесс приносил пользу (1 разработчик на побочном проекте)

G. Рабочий лист оценки

Организации, оценивающие SENAR, РЕКОМЕНДУЕТСЯ измерять следующие показатели на пилоте (минимум 3 инкремента):

Метрика	Пилотное значение	Базовая линия (до SENAR)	Дельта
Задач на пару за сессию	___	___ (на разработчика)	×___
FPSR	___%	Н/Д (новая метрика)	—
Доля утечки дефектов (DER)	___%	___%	___%
Доля переделок	___%	___%	___%
Накладные расходы сессии (мин)	___	Н/Д	—
Создано записей знаний	___	0	+___
Задокументировано тупиковых подходов	___	0	+___

Правило принятия решения

```
IF throughput_multiplier > 1.0 + overhead_ratio
AND defect_escape_rate <= baseline_defect_rate
THEN SENAR is providing net efficiency gain → consider scaling
```

H. Тревожные сигналы

Признаки того, что SENAR снижает эффективность вместо того, чтобы повышать её:

Тревожный сигнал	Что это означает	Действие
Коэффициент накладных расходов > 30%	Процесс обходится дороже, чем приносит пользы	Упростить: свести к MVS, автоматизировать церемонии
FPSR снижается со временем	Качество контекста падает	Проверить базу знаний и качество критериев приёмки
Множитель пропускной способности < 1.0	Пары медленнее обычных разработчиков	Область не подходит для AI, или супервайзер недостаточно подготовлен
Доля обходов шлюзов > 20%	Шлюзы не соответствуют реальности	Перекалибровать критерии шлюзов
Записей знаний = 0	Фиксация знаний заброшена	Как минимум восстановить документирование тупиков
Сессии систематически превышают лимит	Нет дисциплины	Включить контрольные точки, разобраться в причинах

Справочник SENAR: Приложение по управлению и комплаенсу

Это приложение сопоставляет практики SENAR с требованиями управления, регулирования и комплаенса для организаций с AI-нативными командами разработки. Оно адресовано специалистам по нормативному соответствию, аудиторам, юристам и руководителям инженерных подразделений, которые оценивают внедрение SENAR в регулируемых средах.

Отказ от ответственности: Этот документ содержит рекомендуемые практики и руководство по сопоставлению с нормативными требованиями. Он не является юридической консультацией. Для интерпретации требований в конкретной юрисдикции обращайтесь к квалифицированным юристам.

A. Модель ответственности

A.1 Принцип подотчётности

В SENAR **супервайзер несёт ответственность за весь одобренный им результат AI**. Это не делегирование ответственности AI-агенту, а явное принятие ответственности человеком, который направляет, проверяет и одобряет работу.

Цепочка подотчётности следует чёткому принципу:

> ***AI генерирует. Человек одобряет. Человек несёт ответственность.***

Этот принцип встроен в SENAR через шлюзы качества. Когда супервайзер проводит задачу через QG-2 (шлюз реализации), он подтверждает:

- CI проходит, тесты проходят, типы чистые (автоматизированная верификация);
- Критерии приёмки выполнены (человеческое суждение);
- Уязвимости безопасности не обнаружены (с помощью инструментов);
- Результат пригоден для целевого назначения (профессиональное суждение).

Это подтверждение значимо для комплаенса. Оно фиксируется, снабжается временной меткой и привязано к конкретному лицу.

A.2 Ответственность по шлюзам качества

Шлюз	Кто несёт ответственность	Что подтверждает
QG-0 (Контекст)	Супервайзер / Архитектор контекста	Задача чётко определена с проверяемыми критериями приёмки
QG-1 (Требования)	Архитектор контекста	Бизнес-требование одобрено и корректно декомпозировано
QG-2 (Реализация)	Супервайзер	Вывод AI соответствует критериям приёмки, CI проходит, нет проблем безопасности
QG-3 (Верификация)	Инженер верификации / Рецензент	Код проверен, приёмочные тесты проходят, регрессий нет
QG-4 (Приёмка)	Заинтересованная сторона / Архитектор контекста	ПО соответствует бизнес-требованиям, готово к выпуску

Прохождение каждого шлюза — это задокументированное решение об одобрении, принятое конкретным человеком.

A.3 Ответственность при обходе шлюза

Стандарт SENAR Section 8.6(c) требует, чтобы обходы шлюзов включали обоснование, оценку рисков, план устранения и одобрение старшего сотрудника. С точки зрения комплаенса:

- **Лицо, одобрившее обход, принимает на себя риск** за все последующие последствия.
- Записи об обходах шлюзов ОБЯЗАНЫ сохраняться как аудиторские свидетельства.
- Организации РЕКОМЕНДУЕТСЯ отслеживать долю обходов шлюзов (Standard 8.6(d)) как метрику здоровья комплаенса.
- Аудиторам следует рассматривать повышенную долю обходов шлюзов как индикатор процессного давления, требующего расследования.

A.4 Пути эскалации

Ситуация	Путь эскалации
Супервайзер не уверен в корректности вывода AI	Эскалация к инженеру верификации или коллеге-супервайзеру для независимой

	проверки
Вывод AI затрагивает области высокого риска (безопасность, аутентификация, платежи, миграция данных)	ОБЯЗАНА запускать рецензирование согласно политике проверки по уровню риска (Standard 8.7)
Обнаружен дефект после релиза в AI-сгенерированном коде	Процесс реагирования на инцидент (Section F); отследить до исходной задачи и супервайзера
Супервайзер подозревает галлюцинацию AI или фабрикацию ссылок	Остановиться, проверить независимо, задокументировать как Dead End при отказе от подхода
Неясны нормативные или комплаенс-импликации	Эскалация к специалисту по нормативному соответствию (или эквивалентной роли) до продолжения работы

A.5 Совместная ответственность в конфигурациях Командная+

В конфигурациях Командная+ ответственность распределена между выделенными ролями:

- **Архитектор контекста** отвечает за качество требований и полноту прослеживаемости.
- **Инженер знаний** отвечает за точность и актуальность базы знаний.
- **Менеджер потока** отвечает за соблюдение процесса и сбор метрик.
- **Инженер верификации** отвечает за независимую верификацию и тщательность обзоров качества.
- **Супервайзер** по-прежнему отвечает за конкретный вывод AI, которым он руководит и который одобряет.

Такое распределение не размывает личную подотчётность — оно создаёт цепочку задокументированных ответственностей, каждая со своим аудиторским следом.

B. Требования к аудиторскому следу

B.1 Артефакты SENAR как аудиторские свидетельства

Структура SENAR порождает следующие артефакты, каждый из которых является аудиторским свидетельством:

Артефакт	Что содержит	Ценность для комплаенса
Запись задачи	Цель, критерии приёмки, ссылка на требование, тип работы, временные метки жизненного цикла, назначенный супервайзер	Подтверждает плановую, авторизованную работу с прослеживаемостью
Записи шлюзов качества	ID шлюза, прошёл/не прошёл, временная метка, одобрившее лицо, результаты автоматических проверок	Подтверждает применение контролей на каждом этапе
Записи обходов шлюзов	Обоснование, оценка рисков, план устранения, одобритель	Подтверждает управляемую обработку исключений
Логи сессий	Временные метки начала/конца, проработанные задачи, записи контрольных точек, заметки передачи	Подтверждает управляемое выполнение с ограниченной областью
Документы передачи	Итоги сессии, следующие шаги, предупреждения, открытые вопросы	Подтверждает преемственность наблюдения между сессиями
Записи знаний	Решения, паттерны, тупиковые подходы, подводные камни — с временными метками и категоризацией	Подтверждает организационное обучение и фиксацию обоснований
Записи планирования инкремента	Цели, пул задач, бюджет, реестр рисков	Подтверждает плановую поставку с управлением рисками
Записи ретроспективы	Обзор метрик, план vs факт, действия по улучшению	Подтверждает непрерывное совершенствование и управленческий обзор
Отчёты обзора качества	Область аудита, выводы, действия по устранению	Подтверждает периодическую независимую верификацию
Данные метрик	Пропускная способность, Lead Time, FPSR, DER, предсказуемость затрат и т.д.	Подтверждает управление процессом на основе измерений

B.2 Соответствие шлюзов качества аудиторским свидетельствам

Шлюз качества	Создаваемое свидетельство	Что подтверждает
QG-0 (Контекст)	Задача создана с целью, критериями приёмки, ссылкой на требование	Авторизация работы и определение области
QG-1 (Требования)	Одобренное требование, запись декомпозиции	Управление требованиями и одобрение
QG-2 (Реализация)	Результаты CI, результаты тестов, запись одобрения супервайзера	Верификация поставляемого результата, управление изменениями
QG-3 (Верификация)	Запись проверки, результаты приёмочных тестов, результаты сканирования безопасности	Независимая верификация, оценка безопасности
QG-4 (Приёмка)	Одобрение заинтересованной стороны, верификация на staging, запись обзора поставки	Авторизация выпуска, приёмочное тестирование

B.3 Демонстрация надлежащего наблюдения

Аудитор, оценивающий, был ли AI-сгенерированный код надлежащим образом контролирован, ОБЯЗАН проверить:

1. **Качество определения задачи.** Применялся ли QG-0? Были ли у задачи чёткие, проверяемые критерии приёмки до начала работы AI? Хорошо определённая задача свидетельствует о целенаправленном руководстве, а не об открытой генерации AI.

2. **Записи прохождения шлюзов.** Был ли пройден QG-2 с задокументированным одобрением супервайзера? Были ли выполнены и зафиксированы автоматические проверки (CI, тесты, линтинг, сканирование безопасности)? Автоматизированные записи шлюзов — более надёжное свидетельство, чем ручные чеклисты.

3. **Проверка, соответствующая уровню риска.** Был ли уровень риска классифицирован корректно? Прошли ли высокорисковые изменения рецензирование в соответствии с требованиями Standard 8.7? Последовательное применение проверки по уровню риска — ключевой индикатор.

4. **Дисциплина сессий.** Были ли сессии ограничены по длительности? Выполнялись ли контрольные точки с требуемой периодичностью?

Неограниченные сессии без контрольных точек свидетельствуют о недостаточном внимании.

5. **Захват знаний.** Были ли задокументированы решения? Были ли зафиксированы тупиковые подходы? Активный захват знаний указывает на рефлексивное наблюдение, а не пассивное принятие.
6. **Тренд метрик.** Стабильна или улучшается ли доля утечки дефектов (DER)? Растущий DER свидетельствует о деградации качества наблюдения.
7. **Доля обходов шлюзов.** Редки ли обходы и хорошо ли обоснованы? Частые обходы указывают на системные проблемы процесса.

B.4 Рекомендации по срокам хранения

Категория артефакта	Рекомендуемый минимальный срок хранения	Обоснование
Записи задач и шлюзов	Срок жизни ПО + нормативный период хранения	Основное свидетельство прослеживаемости
Логи сессий и передачи	3 года или нормативный минимум (что больше)	Свидетельство наблюдения
Записи знаний	Бессрочно (активное курирование)	Постоянная операционная ценность
Данные метрик	Минимум 3 года	Трендовый анализ и аудиторское свидетельство
Записи обходов шлюзов	Срок жизни ПО	Свидетельство принятия риска
Записи инцидентов	По нормативным требованиям (обычно 5–7 лет)	Пост-инцидентный анализ

C. Сопоставление с нормативными требованиями

C.1 ISO 9001:2015 — Системы менеджмента качества

Пункт ISO 9001	Краткое содержание требования	Артефакт SENAR	Как SENAR удовлетворяет
6.1 — Действия в отношении рисков и возможностей	Идентифицировать риски, влияющие на СМК, спланировать действия	Реестр рисков планирования инкремента; оценки рисков при обходе шлюзов; классификация проверки по уровню риска (Standard 8.7)	Каждый инкремент начинается с задокументированно идентификации рисков. Обходы шлюзов требуют явной оценки рисков Глубина проверки определяется уровне риска.
7.5 — Документированная информация	Создавать, обновлять и управлять документированной информацией	Записи задач, записи шлюзов, записи знаний, логи сессий, передачи	Все артефакты SENAI снабжены временными меткам атрибутированы и хранятся. Записи знаний курируются н актуальность (ответственность инженера знаний).
8.1 — Операционное планирование и контроль	Планировать и контролировать процессы предоставления продукции/услуг	Планирование инкремента с целями, пулом задач и бюджетом; QG-0 обеспечивает определение задачи до начала работы	Работа планируется н уровне инкремента, авторизуется на уровне задачи и контролируется чере автоматизированные шлюзы качества.
8.5 — Производство и предоставление услуг	Контролируемые условия производства	Дисциплина сессий (ограниченная длительность, контрольные точки); руководство AI-агентами со стороны супервайзера; автоматизированное обеспечение CI/CD	AI-производство осуществляется под управлением, в ограниченных по времени условиях с автоматизированны контролями. Ручные вмешательства отслеживаются.
8.6 — Выпуск продукции и услуг	Верифицировать выполнение	QG-4 (шлюз приёмки): одобрение	Выпуск требует задокументированно верификации и

	требований до выпуска	заинтересованной стороны, верификация на staging, QG-3 по-прежнему проходит	приёмки заинтересованной стороной.
9.1 — Мониторинг, измерение, анализ и оценка	Определить, что отслеживать и измерять	8 определённых метрик (4 обязательных, 4 рекомендуемых); ретроспектива инкремента с количественным обзором	Метрики собираются автоматически, проверяются на каждой ретроспективе, базовые линии устанавливаются за 3 инкремента.
10.1 — Улучшение: несоответствие и корректирующее действие	Реагировать на несоответствия, принимать корректирующие действия	Действия по улучшению из ретроспективы (конкретные, измеримые, ограниченные по времени, назначенные); выводы обзора качества и устранение	Ретроспективы порождают назначенные корректирующие действия. Обзоры качества выявляют несоответствия. Тупиковые подходы документируют неудачные попытки.

C.2 SOC 2 Type II — Критерии доверительных услуг

Критерий SOC 2	Краткое содержание требования	Артефакт SENAR	Как SENAR удовлетворяет
CC6.1 — Логический и физический контроль доступа	Ограничить доступ авторизованными пользователями и процессами	SENAR не предписывает инструменты контроля доступа, однако: назначение роли супервайзера контролирует, кто может одобрять прохождение шлюзов; назначение задач документирует авторизованных	Организации должны дополнять SENAR контролями доступа на уровне инфраструктуры. SENAR обеспечива записи авторизаци на уровне процесс

		работников; логи сессий документируют, кто и когда выполнял работу	
CC7.1 — Обнаружение несанкционированной или вредоносной деятельности	Мониторинг и обнаружение аномалий	Обзоры качества обнаруживают выход за рамки, несанкционированные изменения, дрейф конфигурации; правила контроля версий (Standard 10.6) требуют атомарных коммитов с обнаружением секретов; метрика DER отслеживает дефекты, прошедшие через шлюзы	Обзоры качества и автоматизированн сканирование обеспечивают обнаружение. Организации должны дополнять их мониторингом инфраструктуры.
CC7.2 — Мониторинг компонентов системы	Мониторинг компонентов системы для обнаружения аномалий	Логи сессий документируют все изменения под руководством AI; записи шлюзов документируют все одобрения; метрики отслеживают индикаторы здоровья процесса (FPSR, DER)	SENAR обеспечива мониторинг на уровне процесса. Организации должны дополнять его мониторингом на уровне системы (APM, логировани алертинг).
CC8.1 — Изменения инфраструктуры, данных, ПО и процедур	Управлять изменениями через контролируемый процесс	QG-0 через QG-4 образуют конвейер управления изменениями; записи задач документируют авторизацию изменений; записи шлюзов документируют верификацию изменений; обходы шлюзов документируют	Конвейер шлюзов качества SENAR является процесс управления изменениями. Каждое изменение авторизовано (задача), верифицировано (QG-2), независим проверено (QG-3 для Командная+) и выпущено (QG-4).

		контролируемые исключения	

C.3 GDPR — Рекомендации для AI-инструментов, обрабатывающих персональные данные

GDPR применяется, когда AI-инструменты для кодирования обрабатывают персональные данные. Это может происходить, когда:

- Исходный код содержит персональные данные (например, тестовые данные с реальными пользователями, захардкоженные учётные данные);
- Промпты AI включают персональные данные из продакшен-систем (например, отладка с реальными данными);
- Поставщики AI-инструментов обрабатывают и потенциально сохраняют данные промптов;
- Логи сессий содержат персональные данные, упомянутые в ходе разработки.

Аспект GDPR	Рекомендуемая практика SENAR
Правовое основание обработки	Организации должны установить правовое основание (обычно законный интерес или исполнение договора) для любых персональных данных, отправляемых в AI-инструменты. Задокументировать это в записях обработки данных.
Минимизация данных	Подготовка контекста QG-0 РЕКОМЕНДУЕТСЯ исключать персональные данные. Использовать синтетические или анонимизированные данные в промптах AI. Включить проверку классификации данных в область обзора качества.
Соглашения об обработке данных	Организации, обрабатывающие персональные данные ЕС через облачные AI-инструменты, ОБЯЗАНЫ иметь заключённые соглашения об обработке данных (DPA) с поставщиками AI-инструментов, охватывающие обработку и хранение данных промптов.
Право на удаление	Логи сессий, содержащие персональные данные, должны подлежать процедурам удаления. Организациям следует проектировать логирование сессий с минимизацией захвата персональных данных.
Оценка воздействия на защиту данных	Организациям РЕКОМЕНДУЕТСЯ проводить DPIA перед внедрением AI-инструментов для кодирования, которые будут обрабатывать персональные данные, особенно при использовании облачных моделей.

Трансграничная передача	Когда AI-инструменты обрабатывают данные за пределами ЕЭЗ, организации должны обеспечить наличие адекватных механизмов передачи (SCC, решения об адекватности).

C.4 Общие требования к аудиту ПО

Требование к аудиту	Артефакт SENAR	Как SENAR удовлетворяет
Прослеживаемость от требования до реализации	Связь задачи с требованием (требование QG-0); декомпозиция истории → задачи; цели инкремента → истории	Каждая задача связана с родительским требованием. Полная цепочка прослеживаемости от бизнес-цели до реализации.
Авторизация изменений	Создание задачи (авторизация на работу); прохождение QG-2 (авторизация на закрытие); прохождение QG-4 (авторизация на выпуск)	Изменения авторизуются на трёх уровнях: инициация работы, одобрение реализации и выпуск.
Разделение обязанностей	Супервайзер руководит AI; инженер верификации проверяет независимо (конфигурации Командная+); заинтересованная сторона одобряет выпуск	Конфигурации Командная+ обеспечивают разделение ролей. Конфигурации Базовая/ Начальная должны документировать компенсирующие контроли.
Свидетельство тестирования	QG-2 требует прохождения CI и тестов; QG-3 требует прохождения приёмочных тестов; автоматизированные результаты тестов фиксируются как свидетельства шлюзов	Выполнение тестов автоматизировано и фиксируется как часть прохождения шлюза.
Управление инцидентами	Отслеживание инцидентов через задачи; прослеживаемость от инцидента до исходного изменения кода; процесс пост-инцидентного анализа (Section F)	Инциденты прослеживаются через полную цепочку: инцидент → код → задача → требование → супервайзер.

Управленческий обзор	Ретроспектива инкремента с количественными метриками; отчёты обзора качества; записи обзора поставки	Регулярные, структурированные обзоры с задокументированными результатами и действиями по улучшению.
Непрерывное совершенствование	Действия по улучшению из ретроспективы; тренды DER и FPSR; рост базы знаний	Улучшение измеряется (метрики), документируется (действия) и верифицируется (обзор на последующей ретроспективе).

C.5 EU AI Act (Regulation 2024/1689)

AI-ассистенты для кодирования, как правило, попадают в категории «ограниченный риск» или «ИИ общего назначения». Ключевые обязанности:

- **Статья 50: Прозрачность** — пользователи должны быть проинформированы о взаимодействии с AI. Явная маркировка AI-сгенерированной работы в SENAR удовлетворяет этому требованию.
- **Статья 53: Обязанности поставщиков моделей GPAI** — актуально, когда организации дообучают или развёртывают модели.
- Организациям, работающим в ЕС, РЕКОМЕНДУЕТСЯ проверять использование AI-инструментов на соответствие требованиям Акта и вести документацию мер комплаенса.

C.6 NIST AI Risk Management Framework (AI RMF 1.0)

Практики SENAR соответствуют функциям NIST AI RMF:

- **Govern**: Роли (Section 4), Управление (настоящее приложение)
- **Map**: Уровни проверки по рискам (Section 8.7), Классификация данных (D.1)
- **Measure**: Метрики (Section 9), Обзоры качества (Section 7.4)
- **Manage**: Шлюзы качества (Section 8), Реагирование на инциденты (F.1)

Организации, подпадающие под требования федерального управления AI в США, РЕКОМЕНДУЕТСЯ документировать это сопоставление.

C.7 ISO/IEC 27001:2022

Контроли SENAR сопоставляются с Приложением A ISO 27001:

- **A.8.25** (Безопасный жизненный цикл разработки): Шлюзы качества, Чеклист верификации

- **A.8.28** (Безопасное кодирование): Правила 10.6, 10.15, Проверка вывода AI
- **A.8.9** (Управление конфигурацией): Правила 10.13, 10.14
- **A.5.12** (Классификация информации): Классификация данных (D.1)
- **A.8.15** (Логирование): Требования к аудиторскому следу (B.1)

Организации, претендующие на сертификацию ISO 27001, РЕКОМЕНДУЕТСЯ сопоставить артефакты SENAR со своим Заявлением о применимости.

D. Управление данными для AI-инструментов

D.1 Классификация данных для взаимодействия с AI

Перед отправкой любых данных в AI-инструмент для кодирования организации должны классифицировать эти данные и применить соответствующие контроли:

Классификация	Определение	Политика AI-инструмента	Примеры
Публичные	Информация, предназначенная для публичного раскрытия	Любой AI-инструмент (облачный или локальный)	Открытый исходный код, публичная документация, опубликованные API
Внутренние	Информация для внутреннего использования, низкая чувствительность	Облачные AI-инструменты с DPA; локальные AI-инструменты	Внутренние архитектурные документы, нечувствительная бизнес-логика, внутренний инструментарий
Конфиденциальные	Чувствительная бизнес-информация	Предпочтительны локальные AI-инструменты; облачные только с явным DPA, шифрованием и гарантиями отсутствия хранения	Проприетарные алгоритмы, код клиентских функций, конкурентные преимущества

Ограниченного доступа	Высшая чувствительность; нормативные или договорные обязательства	Только локальные AI-инструменты; запрет облачной передачи	Персональные данные (GDPR), данные платёжных карт (PCI DSS), медицинские записи (HIPAA), секреты аутентификации, ключи шифрования

D.2 Критерии выбора AI-инструментов

Организации должны оценивать AI-инструменты для кодирования по следующим критериям, взвешенным с учётом требований классификации данных:

Критерий	Вопросы для ответа
Хранение данных	Хранит ли поставщик промпты или сгенерированный вывод? Как долго? Можно ли отключить хранение?
Отказ от обучения	Использует ли поставщик данные клиентов для обучения модели? Можно ли исключить это договором?
Местоположение данных	Где обрабатываются и хранятся данные? Соответствует ли это применимым требованиям к местоположению данных?
Шифрование	Зашифрованы ли данные при передаче (TLS 1.2+) и при хранении? Кто владеет ключами шифрования?
Контроль доступа	Кто у поставщика имеет доступ к данным клиентов? При каких обстоятельствах?
Право аудита	Включает ли договор право аудита или отчёты аудита третьей стороной (SOC 2, ISO 27001)?
Субобработчики	Использует ли поставщик субобработчиков? Раскрыты ли они и связаны ли договорными обязательствами?
Уведомление об инцидентах	Каковы обязательства поставщика по уведомлению о нарушениях и сроки?

D.3 Локальный vs облачный AI: система принятия решения

Фактор	Облачный AI допустим	Локальный AI необходим

Классификация данных	Публичные или Внутренние	Конфиденциальные или Ограниченного доступа
Нормативная среда	Нет требований к местоположению данных	Суверенитет данных или требования к местоположению
Договорные обязательства	Нет ограничений клиента на облачную обработку	Договоры с клиентами запрещают облачную AI-обработку
Толерантность к рискам	Организация принимает риски облачного поставщика	Нулевая толерантность к внешнему раскрытию данных
Компромисс стоимость-производительность	Стоимость облака приемлема за возможности	Стоимость локального решения оправдана требованиями комплаенса

Организации в регулируемых отраслях (финансы, здравоохранение, государственный сектор, оборона) должны по умолчанию использовать локальные AI-инструменты для всех непубличных данных, если конкретная оценка рисков не обосновывает облачное использование.

D.4 Сроки хранения данных взаимодействия с AI

Тип данных	**Рекомендация по хранению**	**Примечания**
Логи сессий (обезличенные)	Согласно политике хранения аудиторского следа (Section B.4)	Удалить персональные данные перед долгосрочным хранением
Исходные промпты AI	Минимальное хранение; удаление после закрытия сессии, если нет нормативных требований	Промпты могут содержать чувствительный контекст
Сгенерированный AI вывод	Хранится как часть исходного кода под управлением версий	Применяются стандартные политики хранения кода
Записи задач и шлюзов	Срок жизни ПО + нормативный период	Основные артефакты комплаенса
Метрики использования AI-инструментов	Минимум 3 года	Отслеживание затрат, аудиторские свидетельства

D.5 Право на удаление

Критический вопрос для организаций, подпадающих под GDPR или аналогичные нормы конфиденциальности: если персональные данные были включены в промпт AI, можно ли устранить их влияние на модель?

Текущая реальность: Для облачных AI-моделей организации, как правило, не могут гарантировать удаление влияния обучения из весов модели, даже если поставщик удаляет сохранённые данные промптов. Это фундаментальное ограничение текущих архитектур больших языковых моделей.

Рекомендуемые практики:

1. **Предотвращать, а не исправлять.** Классификация данных и подготовка контекста QG-0 должны исключать персональные данные из промптов AI. Предотвращение надёжнее постфактумного удаления.
2. **Договорные оговорки об отказе от обучения.** Обеспечить, чтобы договоры с AI-инструментами явно исключали данные клиентов из обучения моделей.
3. **Удаление данных промптов.** Обеспечить, чтобы договоры включали обязательства по удалению данных промптов, и верифицировать исполнение.
4. **Локальные модели для данных ограниченного доступа.** Когда необходимы гарантии удаления, использовать локальные модели, где организация контролирует полный жизненный цикл данных.
5. **Документировать ограничение.** Если организация не может гарантировать удаление влияния обучения, задокументировать это как известный риск в DPIA и применить компенсирующие контроли (минимизация данных, синтетические данные).

E. Интеллектуальная собственность и лицензирование

Примечание: Вопросы ИС AI-сгенерированного кода — это развивающийся правовой ландшафт со значительными различиями по юрисдикциям. Ниже представлены рекомендуемые практики на момент написания. Организациям следует привлекать квалифицированных юристов для своей конкретной юрисдикции.

E.1 Право собственности на AI-сгенерированный код

Правовой статус собственности на AI-сгенерированный код различается по юрисдикциям и во многих из них остаётся неопределённым:

Подход юрисдикции	Текущая позиция	Последствия для SENAR
Требование человеческого авторства	В ряде юрисдикций для авторского права требуется творческий вклад человека. Чисто машинный вывод может не охраняться авторским правом.	Модель супервайзера в SENAR укрепляет претензии на ИС: супервайзер обеспечивает творческое руководство (цели задач, критерии приёмки, архитектурные решения) и применяет суждение при отборе и одобрении.
Работа по найму	Во многих юрисдикциях работодатель владеет работой, созданной сотрудниками в рамках трудовой деятельности.	Вывод AI-инструмента, направляемый супервайзерами-сотрудниками, вероятно, покрывается существующими соглашениями об ИС при найме, но организациям следует это верифицировать.
Условия поставщика AI-инструмента	Большинство поставщиков AI-инструментов передают права на вывод пользователю, но условия различаются.	Организации должны проверять условия предоставления услуг AI-инструментов на предмет положений о передаче ИС и обеспечить их совместимость с требованиями организации.

Рекомендуемые практики:

1. **Обновить трудовые и подрядные соглашения**, явно охватив код, созданный с помощью AI и AI-сгенерированный код.
2. **Проверить условия поставщика AI-инструмента** для подтверждения передачи прав собственности на вывод.
3. **Документировать творческий вклад человека** через записи задач SENAR (цель, критерии приёмки, архитектурные решения, заметки проверки супервайзера). Эта документация укрепляет претензии на авторство.
4. **Вести записи руководства супервайзера** — предоставленный AI контекст, выбор и модификация вывода AI, суждения, применённые при проверке.

E.2 Риск лицензионного заражения

AI-модели, обученные на открытом исходном коде, могут генерировать вывод, воспроизводящий или близко напоминающий код под копилефт-лицензиями (GPL, AGPL, LGPL). Если такой вывод включается в проприетарное ПО без соблюдения условий, возникает риск лицензионного заражения.

Факторы риска:

- AI-модели, обученные на публичных репозиториях кода без фильтрации по лицензиям;
- Сгенерированный код, близко совпадающий с существующими реализациями открытого кода;
- Недостаточная проверка сгенерированного кода на паттерны, обременённые лицензиями.

Рекомендуемые практики:

1. **Автоматизированное сканирование лицензий.** Включить инструменты сканирования лицензий в конвейер CI (принудительно на QG-2). Сканировать как прямые зависимости, так и сгенерированный код на известные паттерны, обременённые лицензиями.
2. **Аудит зависимостей на QG-3.** Для конфигураций Командная+ включить аудит лицензий зависимостей в область верификации QG-3.
3. **Покрытие обзором качества.** Включить лицензионный комплаенс в область обзора качества (аудит здоровья зависимостей).
4. **Выбор AI-инструмента.** Оценивать политики обучающих данных поставщиков AI-инструментов и любые предлагаемые ими гарантии возмещения убытков по претензиям об ИС.
5. **Документация происхождения кода.** Для кода высокой ценности или высокого риска документировать, был ли он сгенерирован AI, написан человеком или является комбинацией.

E.3 Требования к документации для комплаенса ИС

Организации, стремящиеся продемонстрировать комплаенс ИС для AI-сгенерированного кода, должны вести:

Документ	Назначение	Источник в SENAR
Инвентарь AI-инструментов	Перечень используемых AI-инструментов, их условия обслуживания, положения об ИС	Организационная политика (вне области SENAR)
Записи задач с атрибуцией супервайзера	Документирование творческого руководства и суждения человека	Записи задач, записи шлюзов, логи сессий
Результаты сканирования	Подтверждение отсутствия лицензионного заражения	Автоматизированные результаты QG-2 и QG-3

лицензий		
Записи аудита зависимостей	Документирование лицензионного комплаенса всех зависимостей	Отчёты обзора качества
Записи человеческого вклада	Укрепление претензий на авторство	Записи подготовки контекста, заметки проверок, записи знаний о проектных решениях

F. Реагирование на инциденты с дефектами AI-сгенерированного кода

F.1 Процесс реагирования

При обнаружении продакшен-инцидента, связанного с AI-сгенерированным кодом, в дополнение к стандартному процессу реагирования организации применяется следующий процесс:

1. **Сортировка и сдерживание.** Стандартное реагирование на инцидент: оценка серьёзности, сдерживание воздействия, восстановление сервиса.
2. **Отслеживание происхождения.** Использование цепочки прослеживаемости SENAR:
    - Продакшен-инцидент → изменение кода (контроль версий);
    - Изменение кода → запись задачи (коммит ссылается на задачу);
    - Запись задачи → требование (ссылка QG-0 на требование);
    - Запись задачи → супервайзер (назначение задачи и одобрение QG-2);
    - Запись задачи → сессия (лог сессии с временными метками и контекстом).
3. **Оценка эффективности шлюзов.** Для каждого шлюза качества, через который прошёл дефектный код:
    - Был ли шлюз выполнен? (Проверить записи шлюзов.)
    - Прошли ли автоматические проверки? (Были ли проверки адекватны для данного типа дефекта?)
    - Была ли проведена проверка человеком? (Соответствовала ли она уровню риска согласно Standard 8.7?)
    - Был ли задействован обход шлюза? (Если да, была ли оценка рисков обхода точной?)

4. **Классификация первопричины.** Использование таксономии первопричин, специфичных для AI (Section F.2).

5. **Устранение.** Исправить непосредственный дефект. Создать задачу на любое системное улучшение.

6. **Ротация учётных данных.** При раскрытии учётных данных или секретов (зафиксированы в VCS, включены в контекст AI или залогированы) организация ОБЯЗАНА выполнить ротацию затронутых учётных данных в течение 24 часов и провести аудит логов доступа на предмет несанкционированного использования.

7. **Пост-инцидентный разбор.** Провести структурированный разбор, охватывающий:

    - Было ли наблюдение адекватным для уровня риска данного изменения?
    - Были ли шлюзы качества эффективны? Следует ли усилить критерии шлюзов?
    - Был ли контекст AI-агента достаточным? Следует ли создать записи знаний?
    - Следует ли повысить классификацию риска для данного типа изменений?

Организации РЕКОМЕНДУЕТСЯ определить сроки реагирования на AI-сгенерированные уязвимости в зависимости от серьёзности. Рекомендуется: Критический — 4 часа, Высокий — 24 часа, Средний — 72 часа, Низкий — следующий инкремент.

F.2 Таксономия первопричин, специфичных для AI

К AI-сгенерированному коду применяются стандартные категории первопричин (логическая ошибка, ошибка интеграции, проблема производительности). Следующие дополнительные категории специфичны для AI-нативной разработки:

Категория первопричины	Описание	Индикатор	Превентивный контроль
Галлюцинация	AI фабрикует API, библиотеку или поведение, которых не существует	Код ссылается на несуществующие функции, пакеты или конфигурации	Автоматические проверки QG-2 (компиляция, тесты); верификация супервайзером внешних ссылок

Пробел в контексте	AI не располагал необходимым контекстом для корректного вывода	Код противоречит существующей архитектуре, дублирует функциональность или нарушает недокументированные ограничения	Улучшение подготовки контекста QG-0; записи знаний об ограничениях и конвенциях
Устаревший контекст	AI использовал устаревшую информацию (deprecated API, изменённое требование)	Код использует устаревшие паттерны или API; поведение соответствует старым требованиям	Поддержка актуальности базы знаний (инженер знаний); обновление контекста при начале сессии
Обход шлюза	Шлюз качества был обойдён, и дефект находился в области обхода	Запись обхода шлюза существует для соответствующего шлюза	Пересмотр процесса одобрения обходов; снижение доли обходов
Недостаточные критерии шлюза	Шлюз качества пройден, но критерии были неадекватны для обнаружения дефекта	Записи шлюзов показывают прохождение; тип дефекта не покрывался автоматическими проверками	Усилить критерии шлюзов; добавить тестовое покрытие для данного типа дефектов
Выход за рамки (Scope creep)	AI сгенерировал изменения за пределами области задачи, которые породили дефект	Коммит включает изменения, не связанные с критериями приёмки задачи	Проверка области QG-2 (Standard 10.6(c)); принудительная атомарность коммитов
Пробел в наблюдении	Супервайзер одобрил без надлежащей проверки	Нет свидетельств содержательной проверки; паттерн штамповки в записях шлюзов	Ограничения длительности сессий; метрики качества проверки; аудиты инженера верификации

Эффект накопления	Множество индивидуально корректных изменений AI во взаимодействии создают системную проблему	Ни один отдельный коммит не является дефектным; проблема возникает из комбинации	Обзоры качества (проверка архитектурного соответствия); интеграционное тестирование на QG-3

F.3 Цепочка прослеживаемости

Полная цепочка прослеживаемости для расследования инцидентов:

```
Production Incident
  └─► Code Change (commit hash, diff, timestamp)
        └─► Task Record (goal, AC, work type, Supervisor)
              ├─► Requirement (Story / business requirement)
              ├─► QG-0 Record (context quality at task start)
              ├─► QG-2 Record (implementation verification)
              │     ├─► Automated check results (CI, tests, lint,
security)
              │     └─► Supervisor approval (who, when)
              ├─► QG-3 Record (if applicable: review, acceptance tests)
              ├─► Session Log (when, duration, checkpoints, context)
              └─► Knowledge Entries (decisions, Dead Ends, patterns)
```

Эта цепочка позволяет аудиторам и расследователям инцидентов восстановить полную историю решений от бизнес-требования через реализацию до развёртывания в продакшен.

G. Что не покрывает SENAR

SENAR — это методология процесса управляемой AI-нативной разработки. Следующие области, релевантные для комплаенса, находятся за пределами области SENAR и требуют дополнительных организационных контролей:

Область	Почему вне области SENAR	Что необходимо организации

Контроль доступа к инфраструктуре	SENAR предписывает процессные роли, а не системные разрешения	Политики IAM, RBAC, MFA, управление привилегированным доступом
Сетевая безопасность	SENAR агностичен к инструментам; сетевая архитектура специфична для реализации	Межсетевые экраны, VPN, сегментация сети, защита от DDoS
Стандарты шифрования данных	SENAR не предписывает криптографические контроли	Шифрование при хранении, шифрование при передаче, управление ключами
Непрерывность бизнеса / аварийное восстановление	SENAR охватывает процесс разработки, а не операционную устойчивость	Планы BCP/DR, процедуры резервного копирования, целевые RTO/RPO
Физическая безопасность	SENAR — методология разработки ПО	Контроль доступа в помещения, контроль окружающей среды
Проверка биографий сотрудников	SENAR определяет ответственности, а не HR-процессы	Проверка биографии, процедуры допуска к секретной информации
Управление поставщиками (помимо AI-инструментов)	SENAR охватывает управление данными AI-инструментов	Программа управления рисками третьих сторон
Конфиденциальность по проектированию	SENAR поддерживает минимизацию данных в промптах AI, но не заменяет архитектуру конфиденциальности	Оценка воздействия на конфиденциальность, картирование данных, управление согласиями
Управление AI-моделями	SENAR управляет использованием AI в разработке, а не обучением или развёртыванием AI-моделей	Управление рисками моделей, тестирование предвзятости, мониторинг моделей (актуально для организаций, обучающих или дообучающих модели)
Нормативная отчётность	SENAR производит аудиторские свидетельства, но не автоматизирует нормативные подачи	Рабочие процессы отчётности, нормативные календари, процедуры подачи

Организации, внедряющие SENAR, должны интегрировать его в более широкую систему управления, рисков и комплаенса (GRC), а не рассматривать как самостоятельное решение для комплаенса.

G.1 Компенсирующие контроли для конфигураций Базовая/Начальная

Конфигурации SENAR Базовая (1 пара) и Начальная (1–3 пары) совмещают множество ответственностей в меньшем числе людей. Это создаёт разрыв в разделении обязанностей, который может вызвать вопросы у аудиторов:

Разрыв	Компенсирующий контроль
Супервайзер проверяет сам себя (нет независимого инженера верификации)	Автоматизированное обеспечение шлюзов (автоматические проверки QG-2 не зависят от супервайзера); периодические обзоры качества; рецензирование для высокорисковых изменений
Супервайзер совмещает роль архитектора контекста	Обеспечение QG-0 гарантирует минимальное качество определения задачи независимо от того, кто выполняет роль
Нет выделенного менеджера потока	Автоматизированный сбор метрик снижает зависимость от ручного контроля; ограничения длительности сессий могут быть обеспечены инструментально

Организации в регулируемых отраслях, работающие в масштабе Базовой или Начальной конфигурации, должны задокументировать эти компенсирующие контроли и оценить их адекватность для своих нормативных обязательств. Переход к Командной конфигурации (Team) может быть необходим для удовлетворения требований разделения обязанностей.

H. Чеклист внедрения

Организации, внедряющие SENAR в регулируемой среде, должны решить следующие вопросы:

- ☐ Установить политику классификации данных для взаимодействия с AI-инструментами (Section D.1)
- ☐ Оценить и выбрать AI-инструменты на основе классификации данных и требований комплаенса (Section D.2)
- ☐ Заключить соглашения об обработке данных с облачными поставщиками AI-инструментов (Section D.3)

- [ ] Обновить трудовые и подрядные соглашения об ИС для AI-сгенерированного кода (Section E.1)
- [ ] Интегрировать сканирование лицензий в конвейер CI на QG-2 (Section E.2)
- [ ] Настроить автоматизацию шлюзов качества для создания записей аудиторского уровня (Section B.1)
- [ ] Установить процесс одобрения обхода шлюзов с требованиями к документации (Section A.3)
- [ ] Определить политику хранения данных для артефактов SENAR (Section B.4)
- [ ] Провести DPIA, если AI-инструменты будут обрабатывать персональные данные (Section C.3)
- [ ] Задокументировать компенсирующие контроли при работе в конфигурации Базовая/Начальная (Section G.1)
- [ ] Установить категории первопричин, специфичных для AI, в процессе управления инцидентами (Section F.2)
- [ ] Включить комплаенс AI-инструментов в область обзора качества (Section D, E)
- [ ] Обучить супервайзеров классификации данных и обязательствам комплаенса, специфичным для AI
- [ ] Интегрировать аудиторский след SENAR в существующий инструментарий и отчётность GRC

Справочник SENAR: Требования к инструментарию

Что должны уметь инструменты для работы по SENAR. Требования разделены на два уровня:

- **Базовая/Начальная** — минимум для конфигураций SENAR Базовая (1 пара) и Начальная (1–3 пары), уровень зрелости 2
- **Командная+** — полные требования для конфигураций SENAR Командная и Корпоративная (3+ пар, уровень зрелости 3+)

SENAR не привязан к конкретным инструментам. Здесь описаны требования к возможностям, а не рекомендации продуктов.

---

1. Трекер задач

Трекер задач — это система записи для всех единиц работы. Каждая задача, история и инкремент живут здесь.

1.1 Обязательные поля

Поле	Тип	Базовая/ Начальная	Командная+	Приме
`goal`	Текст	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Что за выпол сформ как ре
`acceptance_criteria`	Текст	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Прове услови заверш
`status`	Перечисление	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Мини состоя `plann` `activ` `block`

`work_type`	Перечисление	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Значе `arch` , `infra`
`complexity`	Перечисление	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Значе `trivi` `moder` `compl`
`story_id`	Ссылка	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Связь родите истори требо
`requirement_link`	Ссылка	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Ссылк просл к BR/S (Stand
`requirement_level`	Перечисление	—	ОБЯЗАТЕЛЬНО	Значе `TR` — связан требо
`requirement_status`	Перечисление	—	ОБЯЗАТЕЛЬНО	Значе `appro` `verif` `depre`
`requirement_parent`	Ссылка	—	ОБЯЗАТЕЛЬНО	Ссылк родите требо навига иерар
`attempt_count`	Целое число	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Число реали расчёт
`created_at`	Временная метка	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Автом создан
`started_at`	Временная метка	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Автом перех `activ`

`completed_at`	Временная метка	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Автом перех
`session_id`	Ссылка	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Какая заверш задачу
`supervisor_id`	Ссылка	—	ОБЯЗАТЕЛЬНО	Кто на этой з
`bypass_reason`	Текст	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Обосн какой- качест обойд
`knowledge_refs`	Список	—	РЕКОМЕНДУЕТСЯ	Ссылк релева записи знаний
`cross_deps`	Список	—	ОБЯЗАТЕЛЬНО	Ссылк межпр зависи
`cost`	Числовое	—	РЕКОМЕНДУЕТСЯ	Стоим токено этой з

1.2 Обязательные автоматизации

Автоматизация	**Базовая/ Начальная**	**Командная+**	**Описание**
Валидация полей QG-0	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Блокировать переход в `active`, если `goal` и `acceptance_criteria` пусты
Временные метки переходов состояний	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Автоматическая фиксация `started_at`, `completed_at` при смене состояния
Правила перехода состояний	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Обеспечение допустимых переходов: `planning` -> `active` -> `done`; `blocked`

			может входить/выходить из любого состояния
Инкремент счётчика попыток	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Увеличение `attempt_count` при каждом возврате задачи из `done` в `active`
Свёртка завершения историй	—	ОБЯЗАТЕЛЬНО	Автоматический расчёт статуса истории из статусов дочерних задач
Отслеживание прогресса инкремента	—	ОБЯЗАТЕЛЬНО	Дашборд с процентом завершения инкремента
Отслеживание обходов	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Запись, какой шлюз был обойдён, кем и с каким обоснованием
Обнаружение дубликатов	—	РЕКОМЕНДУЕТСЯ	Предупреждение, когда цель новой задачи близко совпадает с существующей
Обнаружение осиротевших требований	—	РЕКОМЕНДУЕТСЯ	Выявление требований без задач реализации и задач без требований
Анализ влияния изменения требования	—	РЕКОМЕНДУЕТСЯ	При изменении BR/SR — список всех затронутых нижестоящих артефактов

1.3 Обязательные отчёты

Отчёт	**Базовая/ Начальная**	**Командная+**	**Описание**
Пропускная способность	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Задач, завершённых за сессию, с трендом во времени
Распределение Lead Time	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Гистограмма `completed_at - created_at`, с разбивкой по сложности
FPSR	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	% задач, завершённых с `attempt_count = 1`

DER	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	% задач с дефектами, обнаруженными после статуса `done`
Сводка статусов задач	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Количество задач по статусам, с фильтрацией по инкременту/сессии
Cycle Time	—	ОБЯЗАТЕЛЬНО	Распределение `completed_at - started_at`
Стоимость задачи	—	ОБЯЗАТЕЛЬНО	Общая стоимость / выполненные задачи, с разбивкой по сложности и типу работы
Предсказуемость затрат	—	ОБЯЗАТЕЛЬНО	Факт vs план по стоимости за инкремент
Доля обходов	—	ОБЯЗАТЕЛЬНО	Обходы шлюзов / всего оценок шлюзов, по типу шлюза
Нагрузка супервайзера	—	РЕКОМЕНДУЕТСЯ	Задач на супервайзера за сессию, с учётом сложности
Статус кросс-зависимостей	—	ОБЯЗАТЕЛЬНО	Статус всех межпроектных зависимостей

---

2. Конвейер CI/CD

Автоматизированное обеспечение QG-2 (шлюз реализации) и поддерживающая инфраструктура для QG-3 и QG-4.

2.1 QG-2: Шлюз реализации (автоматизированный)

Все проверки выполняются при каждом коммите или мерж-реквесте. Конвейер должен блокировать мерж при провале любой проверки.

Проверка	**Базовая/ Начальная**	**Командная+**	**Примечания**
Юнит-тесты проходят	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Требуется 100% прохождение

Проверка типов проходит	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	В зависимости от языка: TypeScript strict, mypy и т.д.
Линтер проходит	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Политика нулевых предупреждений (не только ошибок)
Сканирование безопасности (зависимости)	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Обнаружение известных CVE в зависимостях
Сканирование безопасности (SAST)	—	РЕКОМЕНДУЕТСЯ	Статический анализ на антипаттерны безопасности
Сборка успешна	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Чистая сборка с нуля, без кэшированного состояния
Порог покрытия тестами	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Настраиваемый минимум (рекомендация: 70%+)
Обнаружение изменений зависимостей	—	РЕКОМЕНДУЕТСЯ	Пометка новых зависимостей для проверки
Обнаружение breaking changes	—	РЕКОМЕНДУЕТСЯ	Проверка совместимости API для общих интерфейсов

Требования к конфигурации конвейера:

Требование	**Базовая/ Начальная**	**Командная+**
Конвейер запускается автоматически при push/MR	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
Конвейер блокирует мерж при провале	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
Результаты конвейера видны супервайзеру	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
Без `allow_failure` — каждая задача либо работает, либо удаляется	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
Время выполнения конвейера < 10 минут	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО

Конвейер предоставляет ясные сообщения об ошибках	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО
Конвейер версионируется (pipeline-as-code)	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО

2.2 QG-3: Шлюз верификации (ревью)

Требование	Базовая/ Начальная	Командная+	Примечания
Защита ветки main/production	—	ОБЯЗАТЕЛЬНО	Запрет прямого push
Мерж-реквест обязателен	—	ОБЯЗАТЕЛЬНО	Все изменения через MR
Минимум 1 одобрение	—	ОБЯЗАТЕЛЬНО	Одобряющий должен отличаться от супервайзера, создавшего MR
Интеграция чеклиста ревью	—	РЕКОМЕНДУЕТСЯ	Чеклист проверки AI-вывода (Guide 02) встроен в шаблон MR
Одобрение не может быть от автора MR	—	ОБЯЗАТЕЛЬНО	Обеспечение независимой верификации
Инвалидация устаревшего одобрения	—	РЕКОМЕНДУЕТСЯ	Повторное одобрение после новых коммитов

2.3 QG-4: Шлюз приёмки (развёртывание)

Требование	Базовая/ Начальная	Командная+	Примечания
Staging-окружение существует	—	ОБЯЗАТЕЛЬНО	Зеркалирует конфигурацию продакшена
Развёртывание на staging автоматизировано	—	ОБЯЗАТЕЛЬНО	Тот же конвейер, что и для продакшена, другая цель

Этап верификации на staging	—	ОБЯЗАТЕЛЬНО	Ручной или автоматизированный приёмочный тест на staging
Развёртывание в продакшен требует явного одобрения	—	ОБЯЗАТЕЛЬНО	Человеческий шлюз — не авто-деплой при мерже
Возможность отката	—	ОБЯЗАТЕЛЬНО	Возможность откатиться к предыдущей версии за считанные минуты
Отслеживание развёртываний	—	ОБЯЗАТЕЛЬНО	Запись что развёрнуто, когда, кем
Smoke-тесты после развёртывания	—	РЕКОМЕНДУЕТСЯ	Автоматическая верификация успеха развёртывания
Feature flags	—	РЕКОМЕНДУЕТСЯ	Возможность отключить функции без повторного развёртывания

---

3. База знаний

База знаний — это долговременная память проекта. У AI-агентов нет памяти между сессиями — база знаний сохраняет знания.

3.1 Основные требования

Требование	**Базовая/ Начальная**	**Командная+**	**Примечания**
Полнотекстовый поиск	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Супервайзер и AI-агент должны иметь возможность поиска по ключевым словам
Категоризация по типу записи	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Минимум типов: `decision`, `pattern`, `gotcha`, `dead_end`, `observation`

Запись имеет заголовок и тело	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Заголовок для беглого просмотра, тело для деталей
Запись имеет временную метку создания	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Когда это знание было создано
Запись имеет тег проекта/области	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	К какому проекту или области относится это знание
Версионирование записей	—	ОБЯЗАТЕЛЬНО	Отслеживание изменений записей знаний во времени
Отслеживание актуальности	—	ОБЯЗАТЕЛЬНО	Временная метка `last_reviewed`, обновляемая при подтверждении актуальности
Статус записи	—	ОБЯЗАТЕЛЬНО	Значения: `current`, `needs_review`, `deprecated`
Перекрёстные ссылки между записями	—	РЕКОМЕНДУЕТСЯ	Связь между связанными записями знаний
Массовый экспорт	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Экспорт всех записей в переносимом формате (JSON, markdown)
Отслеживание автора записи	—	ОБЯЗАТЕЛЬНО	Кто создал или последним обновил запись

3.2 Типы записей

Тип	Назначение	Пример
`decision`	Архитектурный или проектный выбор с обоснованием	«Мы используем UUID для всех ID сущностей, потому что...»

`pattern`	Повторно используемый подход для типовых задач	«Обработка ошибок в API-эндпоинтах следует такой структуре...»
`gotcha`	Неочевидное поведение, вызывающее проблемы	«CouchDB bulk_docs молча игнорирует конфликты, если не...»
`dead_end`	Подход, который был опробован и отвергнут, с обоснованием	«Пробовали WebSockets как замену SSE — не сработало, потому что...»
`observation`	Эмпирический вывод, заслуживающий запоминания	«Время сборки увеличивается в 2 раза при запуске TypeScript strict mode на...»
`template`	Повторно используемый шаблон требований/КП для типовых задач	«КП REST API эндпоинта: 1. Возвращает 200 при валидном вводе. 2. Возвращает 401 без аутентификации...»

3.3 Интеграция с AI-агентом

Требование	Базовая/ Начальная	Командная+	Примечания
AI-агент может искать в БЗ во время сессии	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Через вызов инструмента/функции или инъекцию контекста
AI-агент может создавать записи БЗ во время сессии	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Тупиковые подходы и подводные камни, обнаруженные в ходе работы
Релевантные записи БЗ инъектируются в контекст сессии	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	На основе области задачи, проекта или явных ссылок
AI-агент может обновлять существующие записи БЗ	—	ОБЯЗАТЕЛЬНО	Исправление или расширение существующих знаний
Результаты запросов к БЗ структурированы	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Заголовок, тип, тело, актуальность — не сырой текстовый дамп

Доступ к БЗ ограничен по области	—	ОБЯЗАТЕЛЬНО	Агент видит знания релевантного проекта, а не всё подряд
Создание записи БЗ запускает очередь на проверку	—	РЕКОМЕНДУЕТСЯ	Человек проверяет записи знаний, созданные AI

3.4 Управление актуальностью

Требование	Базовая/ Начальная	Командная+	Примечания
Записи старше N инкрементов помечаются для проверки	—	ОБЯЗАТЕЛЬНО	Настраиваемый порог (рекомендация: 3 инкремента)
Отчёт по актуальности доступен	—	ОБЯЗАТЕЛЬНО	Распределение возрастов записей, количество по статусам
Устаревшие записи исключены из контекста AI	—	ОБЯЗАТЕЛЬНО	Устаревший контекст хуже отсутствия контекста
Обзор качества включает аудит актуальности БЗ	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Часть чеклиста церемонии

---

4. Управление сессиями

Сессии — основной ритм работы по SENAR. Инструменты должны поддерживать полный жизненный цикл сессии.

4.1 Жизненный цикл сессии

Требование	Базовая/ Начальная	Командная+	Примечания

Начало сессии с временной меткой	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Фиксация начала сессии
Завершение сессии с временной меткой	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Фиксация окончания сессии
Сессия имеет уникальный идентификатор	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Для привязки задач, метрик, передач
Сессия привязана к супервайзеру	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Кто провёл эту сессию
Сессия привязана к проработанным задачам	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Какие задачи были активны в этой сессии
Передача предыдущей сессии доступна при старте	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Преемственность контекста

4.2 Возможности контрольных точек

Требование	**Базовая/ Начальная**	**Командная+**	**Примечания**
Контрольная точка в ходе сессии	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Сохранение состояния контекста без завершения сессии
Контрольная точка фиксирует текущий статус задач	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Что в работе, что выполнено
Напоминание о контрольной точке	—	РЕКОМЕНДУЕТСЯ	Оповещение с настраиваемым интервалом (например, каждые 40–60 вызовов инструментов)
Контрольная точка восстанавливает контекст при сбое сессии	—	РЕКОМЕНДУЕТСЯ	Восстановление после неожиданного завершения сессии

4.3 Документы передачи

Требование	Базовая/ Начальная	Командная+	Примечания
Документ передачи создаётся при завершении сессии	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Итоги сделанного и план следующих шагов
Передача включает завершённые задачи	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Список задач, выполненных в этой сессии
Передача включает незавершённые задачи	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Состояние задач, не завершённых
Передача включает блокеры/риски	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Что может помешать успеху следующей сессии
Передача включает захваченные знания	—	ОБЯЗАТЕЛЬНО	Записи БЗ, созданные или обновлённые в ходе сессии
Передача хранится и доступна для поиска	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Исторические передачи доступны для справки
Передача загружается автоматически при следующем старте сессии	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Без ручного поиска последней передачи

4.4 Сбор метрик

Требование	Базовая/ Начальная	Командная+	Примечания
Количество вызовов инструментов за сессию	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Опережающий индикатор сложности сессии и утомления
Длительность сессии	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Вычисляется из временных меток начала/конца

Задач, завершённых за сессию	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Вход для расчёта пропускной способности
Стоимость сессии (токены/затраты на API)	—	ОБЯЗАТЕЛЬНО	Для расчёта стоимости задачи
Количество ошибок/повторов	—	РЕКОМЕНДУЕТСЯ	Сколько раз AI требовалась коррекция

---

5. Интеграция AI-агента

Требования к настройке, мониторингу и управлению AI-агентами в рамках SENAR.

5.1 Инъекция контекста

Требование	**Базовая/ Начальная**	**Командная+**	**Примечания**
Структурированные инструкции проекта	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Контекст уровня проекта, загружаемый при старте сессии (например, CLAUDE.md, .cursorrules)
Инъекция контекста задачи	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Цель, КП и релевантные ограничения предоставлены агенту
Инъекция записей БЗ	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Релевантные записи знаний доступны в ходе работы
Конвенции кодовой базы	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Стандарты кода, архитектурные паттерны, конвенции именования
Границы области	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Явные инструкции «не изменять»
Инъекция истории/ передачи сессии	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Контекст предыдущей сессии для преемственности

5.2 Захват вывода

Требование	Базовая/ Начальная	Командная+	Примечания
Изменения кода фиксируются в контроле версий	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Стандартный git-процесс
Логи сессий доступны для проверки	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Что AI делал, что супервайзер одобрил
Рассуждения AI видимы	РЕКОМЕНДУЕТСЯ	РЕКОМЕНДУЕТСЯ	Почему AI принял конкретные решения (для верификации)
Вывод привязан к задаче	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Какая задача породила какие изменения кода

5.3 Отслеживание стоимости и токенов

Требование	Базовая/ Начальная	Командная+	Примечания
Использование токенов за сессию	—	ОБЯЗАТЕЛЬНО	Входящие + исходящие токены
Стоимость API за сессию	—	ОБЯЗАТЕЛЬНО	Денежная стоимость, вычисленная из использования токенов
Стоимость задачи (вычисляемая)	—	ОБЯЗАТЕЛЬНО	Стоимость сессии / завершённые задачи
Тренд стоимости во времени	—	ОБЯЗАТЕЛЬНО	Процесс становится дороже или дешевле?
Бюджетные предупреждения	—	РЕКОМЕНДУЕТСЯ	Оповещение при приближении стоимости сессии/инкремента к бюджету
Стоимость по модели/поставщику	—	РЕКОМЕНДУЕТСЯ	При использовании нескольких моделей —

			отдельное отслеживание стоимости

5.4 Мультиагентная оркестрация (Командная+)

Требование	Базовая/ Начальная	Командная+	Примечания
Изоляция рабочего пространства	—	РЕКОМЕНДУЕТСЯ	Агенты работают в изолированных средах (worktrees, контейнеры)
Коммуникация агент-агент	—	РЕКОМЕНДУЕТСЯ	Делегирование подзадач специализированным агентам
Параллельное выполнение агентов	—	РЕКОМЕНДУЕТСЯ	Несколько агентов работают одновременно под наблюдением
Агрегация вывода агентов	—	РЕКОМЕНДУЕТСЯ	Сбор результатов от нескольких агентов в единый обзор
Профили возможностей агентов	—	РЕКОМЕНДУЕТСЯ	Разные агенты для разных типов задач (код, тесты, документация)

5.5 Управление конфигурацией агентов

Требования к управлению профилями агентов и операционными скриптами (Standard Section 5).

Требование	Базовая/ Начальная	Командная+	Примечания
Хранение скриптов в контроле версий	ОБЯЗАТЕЛЬНО	ОБЯЗАТЕЛЬНО	Скрипты хранятся вместе с кодом проекта
Версионирование скриптов (история изменений)	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Отслеживание кто, что, когда и зачем изменил
Хранение определений профилей	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Профили агентов хранятся как структурированная конфигурация

Поддержка переключения профилей	—	РЕКОМЕНДУЕТСЯ	Возможность смены контекста агента между профилями в рамках сессии
Ограничение прав по профилю	—	ОБЯЗАТЕЛЬНО	Обеспечение границ чтения/записи для каждого профиля агента
Механизм распространения скриптов	—	ОБЯЗАТЕЛЬНО	Распространение изменений скриптов по множеству проектов
Возможность отката скриптов	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Откат любого изменения скрипта к предыдущей версии
Реестр активных скриптов	—	ОБЯЗАТЕЛЬНО	Инвентарь активных скриптов с идентификаторами версий
Аудиторский след изменений скриптов	—	ОБЯЗАТЕЛЬНО	Лог всех модификаций скриптов с обоснованием
Инвентарь инструментов по профилю	РЕКОМЕНДУЕТСЯ	ОБЯЗАТЕЛЬНО	Документация, какие инструменты доступны каждому профилю
Движок выполнения хуков	—	РЕКОМЕНДУЕТСЯ	Автоматические действия по событиям (пост-сессия, пре-коммит)
Метрики эффективности скриптов	—	РЕКОМЕНДУЕТСЯ	Отслеживание FPSR и доли ошибок по версии скрипта

---

6. Требования к интеграции

Как компоненты соединяются в работающую инструментальную цепочку SENAR.

6.1 Интеграции Базовая/Начальная

Интеграция	Описание
Трекер задач <-> Управление сессиями	Сессии ссылаются на задачи; задачи ссылаются на сессии
Трекер задач <-> CI/CD	Результаты конвейера привязаны к задачам и мерж-реквестам
База знаний <-> AI-агент	Агент может искать, читать и создавать записи БЗ во время сессий
Управление сессиями <-> AI-агент	Старт/завершение сессии управляются или координируются с жизненным циклом агента

6.2 Интеграции Командная+

Интеграция	Описание
Трекер задач <-> База знаний	Задачи ссылаются на релевантные записи БЗ; записи БЗ ссылаются на породившие их задачи
CI/CD <-> Трекер задач	Провалы конвейера автоматически обновляют статус задачи в `blocked`
Управление сессиями <-> Метрики	Метрики сессии автоматически заполняются из данных сессии
База знаний <-> CI/CD	Записи БЗ об архитектурных ограничениях обеспечиваются в конвейере (перспективное)
Федерация <-> Трекер задач	Статус межпроектных зависимостей синхронизируется
Федерация <-> База знаний	Межпроектные знания доступны с надлежащей областью видимости

---

7. Руководство по выбору

При оценке вариантов инструментария приоритизируйте:

1. **Автоматизация важнее ручного процесса** — если шаг можно автоматизировать, автоматизируйте. Ручные шаги — точки отказа (см. Guide 06, PF-3: Нормализация обхода шлюзов).

2. **Переносимость важнее функций** — данные должны экспортироваться. Привязка к поставщику (Guide 06, OF-3) — известный режим отказа.

3. **Доступность для AI важнее удобства для человека** — каждое хранилище данных, которым пользуется супервайзер, должно быть доступно AI-агенту. Если AI не может к нему обратиться — для SENAR оно не существует.

4. **Простота важнее полноты** — трекер задач с 5 хорошо работающими полями лучше трекера с 50 полями, которые никто не заполняет. Начните с Базовая/Начальная и добавляйте Командная+ по мере надобности.

5. **Композируемость важнее монолитности** — выбирайте инструменты с API-интеграцией, а не универсальные платформы. Компоненты SENAR развиваются с разной скоростью; замена одного не должна тянуть за собой остальные.
