TRIZLY · AI-АГЕНТ ИНЖЕНЕРНЫХ, ПРОДУКТОВЫХ И БИЗНЕС-РЕШЕНИЙ

Разрешение противоречия «мгновенно внутри ↔ изолированно между сервисами» в архитектуре платформы

Демонстрационный отчёт для IT-архитекторов: ТРИЗ-анализ архитектурной задачи, расчёты задержек и доступности, экономика решений и шаблоны для архитектурного ревью.

Тип задачи	Методология	Дата анализа	Формат
Инженерная / архитектура ПО	ТРИЗ: 9 фаз, 4 стратегии	28 сентября 2026	Демо-кейс для сайта

1. Резюме

Маркетплейс разделил монолит на 9 микросервисов, и оформление заказа превратилось в цепочку из 8 синхронных вызовов: p95 задержки вырос до 1,4 с, а сбой одной зависимости каскадно останавливает продажи. Команды хотят полной изоляции сервисов (независимые релизы, локализация отказов), а бизнес требует мгновенного ответа на чекауте. Обе попытки «выбрать сторону» провалились: возврат к общей базе вернул скорость, но убил независимость команд; полностью событийная схема дала изоляцию, но привела к перепродаже остатков. Это классическое ТРИЗ-противоречие: связь между компонентами должна быть тесной и слабой одновременно.

Агент Trizly прошёл полный девятифазный цикл анализа для инженерной задачи: сформулировал два технических противоречия (по обоим направлениям) и физическое противоречие, определил идеальный конечный результат (ИКР), провёл ресурсный аудит, сгенерировал решения по четырём стратегиям, объединил их, проверил «скептиком» (главный архитектор, SRE, руководитель платформы, финансовый директор) и ранжировал по составному критерию. Для архитекторов добавлены расчёты бюджета задержек и композитной доступности, оценка экономики и шаблоны (ADR, SLO, вопросы для ревью).

> **Итог:** из 5 проверенных решений приоритет — «защитные оболочки» (bulkhead, circuit breaker, деградация) и «локальные модели чтения» (данные, которые меняются редко, читаются локально и обновляются событиями); далее — предварительная подготовка на «клиентском времени». Расчёт: критический путь сокращается с 8 до 3 синхронных вызовов, p50 — с 640 до ≈240 мс, p95 — до ≈380 мс, доступность чекаута — с 99,2% до ≈99,93%.

2. Описание задачи (кейс)

Контекст. Маркетплейс, 9 продуктовых команд, ≈54 инженера. Взаимодействие сервисов — синхронный REST/JSON; брокер сообщений используется только для аналитики; есть API-шлюз, кэши на уровне отдельных сервисов и распределённая трассировка. Кейс учебный: числа приближены к типичным и служат для иллюстрации методики.

Заказов: 300 000 в месяц, средний чек 3 200 ₽, оборот ≈960 млн ₽/мес, валовая маржа 8%

Конверсия чекаута: 61,0% при p95 ≈400 мс → 57,9% при p95 1,4 с **Доступность каждого сервиса:** 99,9%

Каскадные инциденты: 3 в квартал, в среднем 47 минут

Кросс-командные релизы: 35%; срок поставки изменения (lead time) 14 дней

Шаг критического пути	Сервис	p50, мс	Как часто меняются данные	Критичность для заказа
1	Корзина	80	постоянно (у пользователя)	высокая
2	Цены	80	редко (≈3 раза в сутки на позицию)	средняя
3	Акции	80	редко	средняя
4	Остатки	80	очень часто	критичная
5	Антифрод	80	на каждый заказ	высокая
6	Платёж	80	на каждый заказ	критичная
7	Заказ	80	на каждый заказ	критичная
8	Лояльность	80	после заказа	низкая

Симптомы. Восемь обязательных зависимостей по 99,9% дают композитную доступность цепочки ≈99,2% (≈5,7 часа простоя в месяц). Задержки суммируются: p50 640 мс, p95 1,4 с, p99 3,2 с. Типичный инцидент: сервис «Акции» деградирует, вызовы висят до тайм-аута 30 с, пулы потоков в «Корзине» и «Заказе» исчерпываются, чекаут недоступен ≈47 минут.

Две попытки «выбрать сторону». Тесная связь: корзина, цены и акции переведены на общую базу — p95 380 мс, но срок поставки вырос с 14 до 21 дня, а один инцидент затронул 6 доменов. **Полная асинхронность:** оформление заказа переведено на события — «принято» за 320 мс, но перепродажа остатков в 0,8% заказов (≈2 400 в месяц), задержка статуса до 6 с и расхождения по платежам, требующие ручной сверки.

Цель. p95 чекаута ≤400 мс и p99 ≤1 с, доступность ≥99,9%, ≤1 каскадного инцидента в квартал, доля кросс-командных релизов ≤10%, перепродажа ≤0,05%.

Кейс сконструирован как репрезентативный пример класса задач «скорость и консистентность против изоляции сервисов», типичного для e-commerce, финтеха, логистики и любых распределённых платформ — данные приближены к типичным порядкам величин и приведены для демонстрации работы агента.

3. Классификация типа задачи ФАЗА 0 — INTAKE+

ИНЖЕНЕРНАЯ ЗАДАЧА — архитектура ПО, распределённые системы, надёжность и производительность.

Границы системы: критический путь оформления заказа: шлюз, 8 сервисов, брокер, кэши, наблюдаемость

Главная полезная функция: мгновенно и корректно подтвердить заказ, не связывая судьбу сервисов друг с другом

Условия эксплуатации: пики нагрузки ×3 в распродажи; 9 независимых команд; деградация отдельных зависимостей

Ограничения: без остановки продаж; не расширять зону обработки платёжных данных (PCI DSS); бюджет ≈24 человеко-месяца за 4 месяца; без смены облака и брокера

4. Техническое противоречие ФАЗА 1

ТП-1: Улучшение изоляции сервисов и локализации отказов (аналог парам. 27 — надёжность) → рост задержки и снижение скорости отклика (аналог парам. 9 — скорость).
По аналогии с матрицей (27 × 9) справочник агента рекомендует принципы:

№35 Изменение параметров | №28 Замена механической схемы | №2 Извлечение | №1 Сегментация

ТП-2 (обратное): Улучшение скорости отклика (парам. 9) → снижение надёжности и изоляции (парам. 27).
По аналогии с матрицей (9 × 27) справочник агента рекомендует принципы:

№10 Предварительное действие | №28 Замена механической схемы | №19 Периодическое действие | №2 Извлечение

Для программных систем соответствие параметрам матрицы — аналогия (парам. 27 — устойчивость к отказам и независимость, парам. 9 — время отклика); принципы фиксируются в Фазе 1 и применяются в Фазе 4.

Цепочка «почему» (углубление до корня)

1. **Почему изоляция увеличивает задержку?** Изолированные сервисы общаются по сети: каждая проверка (цены, акции, остатки, антифрод) — отдельный синхронный запрос с сериализацией и ожиданием ответа; в цепочке из 8 обязательных вызовов задержки суммируются, а тайм-ауты и повторы раздувают «хвост» (p99 в 5 раз выше p50).
2. **Почему нельзя просто сделать всё асинхронным?** Часть данных требует немедленного согласованного ответа (остатки, платёж): при асинхронной обработке появляются окна неконсистентности (перепродажа в 0,8% заказов), а клиенту приходится ждать подтверждения. Граница «мгновенно и согласованно» проходит не по границам сервисов.
3. **Почему границы сервисов не совпадают с границами консистентности?** Монолит разрезали по слоям и командам (цены, акции, остатки, заказы), а не по границам транзакций и данных, нужных для одной пользовательской операции. Данные, которые нужны вместе, разнесены, и каждый заказ собирает их по сети «на лету» в критическом пути.

Физическое противоречие (Уровень 3)

Связь между компонентами должна быть **тесной** (мгновенный, согласованный ответ пользовательской операции) — функция 1,
и одновременно должна быть **слабой** (изоляция, независимые релизы, локализация отказов) — функция 2.

Зона конфликта: по месту — критический путь «Корзина → Цены → Акции → Остатки → Антифрод → Платёж → Заказ → Лояльность» и границы между сервисами; по времени — момент подтверждения заказа (нужен мгновенный ответ) в отличие от остальных шагов, которые можно отложить; по условию — пики нагрузки и деградация одной из зависимостей.

5. Идеальный конечный результат (ИКР) ФАЗА 2

ИКР-1 (функциональный): Ответ клиенту приходит мгновенно, а сбой или релиз одного сервиса не влияет на другие — без новой инфраструктуры и роста сложности.

ИКР-2 (ресурсный): Используя только существующие ресурсы (события об изменениях, которые уже публикуются для аналитики, кэши, шлюз, брокер, трассировку), критический путь читает всё нужное локально, а обмен между сервисами идёт асинхронно вне критического пути.

ИКР-3 (предельный — устранение элемента): Сетевых вызовов в критическом пути почти нет: всё нужное для подтверждения заказа находится «рядом», в одном малом контуре консистентности, а связь между контурами — события; сервисы изолированы, но пользователь этого не замечает.

6. Ресурсный анализ ФАЗА 3

Тип ресурса	Что доступно	Статус
Вещество (компоненты, данные)	9 сервисов; API-шлюз; брокер сообщений; кэши; базы сервисов; схемы контрактов	Брокер работает только на аналитику
Поле (потоки, сигналы)	Поток событий об изменении цен и остатков (уже публикуется); тайм-ауты и повторы; сетевой трафик	Поток не используется в операционном пути
Пространство	Возможность размещать данные рядом с потребителем; память под локальные копии; зоны доступности	Не используется
Время	Пока клиент вводит данные карты (≈20–40 с), система простаивает; шаги после подтверждения можно вынести из ответа	Не используется
Информация	Метрики p50/p95/p99, трассировки, профили рисков, история изменений цен и акций	Есть, но не управляет поведением системы
Функция/ свойство	Данные о ценах и акциях меняются редко и допускают ограниченную устарелость; шлюз и сервис заказа могут оркестрировать шаги	Не используется

7. Генерация решений — 4 параллельные стратегии ФАЗА 4

Ниже — необработанные решения по каждой стратегии до объединения (Фаза 5) и проверки (Фаза 6).

A. Принципы + матрица противоречий

- **№35 Изменение параметров** (ТП-1, по аналогии с матрицей) — менять параметры взаимодействия: тайм-ауты, ретраи, размеры пулов, режим согласованности (строгая → ограниченно устаревшая) в зависимости от типа данных.
- **№28 Замена механической схемы** (ТП-1 и ТП-2) — заменить синхронный вызов «запрос — ответ» локальным чтением из копии данных, обновляемой событиями.
- **№2 Извлечение** (ТП-1 и ТП-2) — вынести из критического пути всё, что не требует мгновенного ответа: лояльность, уведомления, аналитику, повторные проверки.
- **№1 Сегментация** (ТП-1) — перекроить границы сервисов по границам консистентности: разделить данные на «строго согласованные и малые» и «допускающие устарелость».
- **№10 Предварительное действие** (ТП-2) — заранее подготовить данные: материализованные представления цен и акций, резервирование остатков до оплаты, прогрев соединений.
- **№19 Периодическое действие** (ТП-2) — периодическое обновление копий и heartbeat вместо постоянных синхронных запросов; пакетная запись.
- **№3 Местное качество** (дополнительно) — разные режимы консистентности для разных данных: остатки и платёж — строго, цены и акции — с допустимой устарелостью до 5 с.
- **№11 Заранее подложенная подушка** (дополнительно) — резервные значения при сбое: последняя известная цена, отложенная проверка.
- **№15 Динамичность** (дополнительно) — адаптивные тайм-ауты, автоматическая деградация, circuit breaker.
- **№24 Посредник** (дополнительно) — шлюз или оркестратор, сокращающий число вызовов.

B. Су-поле + 76 стандартов

Модель: S1 (вызывающий сервис) — F (синхронный запрос) → S2 (вызываемый сервис). Поле связывает сервисы: полезное действие — быстрый ответ, вредное — «сцепление»: медленность и отказ S2 передаются S1.

- **Стандарт 2.1.1** — ввести S3, поглощающий вредный эффект между S1 и S2: circuit breaker, bulkhead (отдельные пулы), кэш последнего состояния, очередь.

- **Стандарт 1.2.4** — заменить механическое (синхронное) поле другим — событийным (сообщения).
- **Стандарт 1.2.7** — разделить S2 на части, взаимодействующие независимо: критичные данные (остатки, платёж) и данные, допускающие устарелость (цены, акции), идут разными путями.
- **Стандарт 3.1.1** — перейти к двойной системе: синхронный малый контур для критичного и асинхронный контур для остального.
- **Стандарт 3.1.6** — сделать систему динамичной и адаптивной: адаптивные тайм-ауты, автомасштабирование, деградация.
- **Стандарты 4.1.3 и 4.1.4** — измерять динамику и периодически обнаруживать выход за границы: SLO, скорость расхода бюджета ошибок вместо разовых порогов.
- **Стандарт 5.1.2** — ввести «вещество» временно: резерв остатка с ограниченным временем жизни (TTL) исчезает сам.
- **Стандарт 5.2.1** — использовать поля, уже присутствующие в системе: уже публикуемые события изменения цен и остатков.

C. Аналогии физических эффектов

- **Конденсатор (буфер-накопитель)** — локальные копии и очереди сглаживают всплески и разрывают связь источника и потребителя по времени.
- **Предохранитель** — circuit breaker размыкает цепь при перегрузке и защищает остальную систему.
- **Гальваническая развязка** — передача сигнала без общей «земли»: обмен событиями без общих баз данных и разделяемых ресурсов.

D. ARIZ-85V

Не потребовалось — стратегии A–C дали достаточное покрытие решения; переход к Фазе 5 без полного цикла ARIZ.

Принципы разделения противоречивых свойств

- **По времени:** мгновенно — только подтверждение заказа; всё остальное — после ответа, асинхронно; пока клиент вводит карту, идут предварительные проверки.
- **По месту:** строго согласованный контур — маленький (остатки, платёж, запись заказа); всё прочее — за его границей, через события.
- **По условию:** режим взаимодействия зависит от критичности данных и состояния зависимости: строгая консистентность для остатков и платежей, ограниченная устарелость для цен, деградация при сбое.
- **Целое/часть:** ядро «расчёт корзины» — тесно связанная часть (один контекст и релиз внутри модульной структуры), периферия — слабо связанные сервисы.

8. Синтез решений ФАЗА 5

1. Границы по консистентности

Корзина, цены и акции объединяются в один контекст «Расчёт корзины» с модульной структурой внутри: вызовы в памяти, один релиз, чёткие внутренние контракты. Остатки и платёж остаются отдельными. Критический путь сокращается на 2 синхронных вызова. Принципы №1, №3; стандарт 1.2.7.

2. Локальные модели чтения

Цены и акции публикуют события об изменениях; сервис заказа и корзины держит локальную материализованную копию с допустимой устарелостью ≤5 с; при финализации заказа цена перепроверяется. Сетевые вызовы за редко меняющимися данными уходят из критического пути. Принципы №28, №2, №26; стандарты 1.2.4, 5.2.1.

3. Защитные оболочки (bulkhead, circuit breaker, деградация)

Для каждой зависимости — отдельный пул потоков, тайм-аут по SLO, circuit breaker и резервный сценарий (последняя известная цена, отложенная проверка); нерелевантные зависимости (лояльность, уведомления) выводятся из критического пути. Принципы №35, №11, №15; стандарты 2.1.1, 3.1.6.

4. Резервирование остатков и сага

Остатки резервируются синхронно с TTL (строгая консистентность в малом контуре); платёж и остальные шаги оркеструются сагой с компенсациями и идемпотентными операциями; клиент получает подтверждение после резерва и авторизации платежа, остальное выполняется асинхронно. Принципы №2, №10; стандарты 5.1.2, 3.1.1.

5. Предварительная подготовка на «клиентском времени»

Пока клиент вводит данные оплаты (≈20–40 с), система заранее считает корзину, запускает антифрод-скоринг, прогревает соединения и делает предавторизацию; финальный вызов только подтверждает результат. Принципы №10, №19.

9. Адверсарная проверка ФАЗА 6

Каждое решение проверено «скептиком» (главный архитектор, SRE, руководитель платформы, финансовый директор) по 4 критериям: логика, экономика, реализуемость, побочные эффекты.

Решение	Статус	Реализуемость	Трудоёмкость	Ключевые риски
1. Границы по консистентности	С ДОРАБОТКОЙ	3,5/5	Высокая (≈8 чел.-мес.)	Потеря независимости команд «Цены» и «Акции» (закон Конвея); нужны внутренние контракты и модульные границы; миграция данных и оргизменения
2. Локальные модели чтения	С ДОРАБОТКОЙ	4/5	Средняя (≈6 чел.-мес.)	Клиент может увидеть устаревшую цену — нужны перепроверка при финализации и метрика лага; порядок и дубликаты событий требуют идемпотентности
3. Защитные оболочки	ПОДТВЕРЖДЕНО	4,5/5	Низкая (≈3 чел.-мес.)	Неверные пороги дают ложные срабатывания; резервные значения могут быть устаревшими; проверка chaos-тестами
4. Резервирование и сага	С ДОРАБОТКОЙ	3,5/5	Средняя–высокая (≈5 чел.-мес.)	Сложность компенсаций, «зависшие» резервы, гонки; нужны идемпотентность, мониторинг саг и опыт команды
5. Подготовка на клиентском времени	С ДОРАБОТКОЙ	3,5/5	Средняя (≈2 чел.-мес.)	Лишняя нагрузка на брошенных корзинах (до 40% сессий не завершаются); утечка сигналов антифрода — нужны лимиты, кэш и ограничение частоты

10. Ранжирование по составному критерию ФАЗА 7

Composite = Инновационность×0,30 + Реализуемость×0,30 + Стоимость×0,20 + Универсальность×0,20

№1 · Защитные оболочки — 0.74

Инновационность 0.40 · Реализуемость 0.90 · Стоимость 0.90 · Универсальность 0.85

№2 · Локальные модели чтения — 0.72

Инновационность 0.65 · Реализуемость 0.75 · Стоимость 0.65 · Универсальность 0.85

№3 · Подготовка на «клиентском времени» — 0.70

Инновационность 0.75 · Реализуемость 0.65 · Стоимость 0.85 · Универсальность 0.55

№4 · Резервирование остатков и сага — 0.65

Инновационность 0.60 · Реализуемость 0.70 · Стоимость 0.55 · Универсальность 0.75

№5 · Границы по консистентности — 0.63

Инновационность 0.75 · Реализуемость 0.55 · Стоимость 0.40 · Универсальность 0.80

Рекомендуемая связка по горизонтам. Быстрый эффект (1–2 месяца): защитные оболочки и локальные модели чтения. Средний срок (2–4 месяца): подготовка на «клиентском времени» и резервирование остатков с сагой. Стратегически: перекройка границ по консистентности — только после измерения эффекта первых шагов.

11. Анализ эволюции и системный оператор ФАЗА 8

	Прошлое	Настоящее	Будущее
Надсистема	Монолитная платформа и единая команда	Множество команд и сервисов с синхронными связями	Платформа с событийным ядром, контрактами и автоматическим управлением надёжностью
Система	Монолит: быстро и связанно	Синхронная цепочка сервисов → конфликт скорости и изоляции	Малые контуры консистентности + асинхронная периферия; модульные границы по данным
Подсистема	Вызов функции в памяти	Сетевой вызов с тайм-аутом и ретраями	Локальное чтение из копии, обновляемой событиями; самонастраивающиеся защитные политики

Закон эволюции: переход от жёсткой связи к динамической и адаптивной (от синхронных вызовов к событиям и локальным копиям); переход на микроуровень (границы по данным и транзакциям); рост степени управляемости (SLO и автоматическая деградация вместо ручных реакций на инциденты).

Следующее вероятное новое противоречие: «свежесть данных и простота модели ↔ независимость и масштабируемость»: чем больше локальных копий, тем сложнее отслеживать их согласованность, порядок событий и владельцев данных; вырастут стоимость наблюдаемости и когнитивная нагрузка на команды.

12. Образец расчётов для архитектурного ревью

Методический образец: значения задержек и коэффициентов приняты для иллюстрации. В реальном проекте их берут из трассировок и нагрузочных тестов и проверяют экспериментом.

12.1. Бюджет задержек критического пути

Путь	Синхронные вызовы	p50	p95	p99
Сейчас	8 (по 80 мс)	640 мс	1 400 мс	3 200 мс
После: резерв остатка, авторизация платежа, запись заказа	3 (по 80 мс)	240 мс	≈380 мс (множитель хвоста 1,6)	≈900 мс (оценка)

$p_{50} = n \cdot t = 8 \cdot 80 = 640$ мс → $3 \cdot 80 = 240$ мс
$p_{95} \approx k \cdot p_{50} = 1{,}6 \cdot 240 \approx 384$ мс ($k = 1{,}6$ принято для пути с защитными оболочками; сейчас $k = 1\,400 / 640 \approx 2{,}2$)
Бюджет тайм-аутов (p99): резерв остатка 150 мс + авторизация платежа 400 мс + запись заказа 200 мс = 750 мс < 1 000 мс; повтор — только идемпотентных операций и не более одного

12.2. Композитная доступность цепочки

Сейчас: $A = 0{,}999^8 \approx 0{,}9920$ (99,2%); простой ≈ $(1 - 0{,}9920) \cdot 720$ ч ≈ 5,7 ч в месяц
Три обязательные зависимости без запасных сценариев: $0{,}999^3 \approx 99{,}70\%$ (≈2,2 ч в месяц) — недостаточно
С запасными сценариями (второй платёжный провайдер, оптимистичная приёмка при сбое резерва для малорисковых позиций, запись заказа в локальный outbox): $A \approx 0{,}9999 \cdot 0{,}9999 \cdot 0{,}9995 \approx 99{,}93\%$ (≈30 минут в месяц)
Допущения: независимость отказов; эффективная доступность зависимостей с запасным сценарием 99,99%; сервис заказа 99,95%

12.3. Допустимая устарелость локальных копий

Цена позиции меняется в среднем раз в 8 ч = 28 800 с; лаг доставки события p99 ≤ 2 с
P(устаревшая цена при чтении) ≈ лаг / интервал = 2 / 28 800 ≈ $7 \cdot 10^{-5}$
Заказов в месяц с устаревшей ценой ≈ 300 000 · $7 \cdot 10^{-5}$ ≈ 21 на одну позицию; при 2,4 позиции в заказе ≈ 50 (≈0,017%)
Такие случаи перехватываются перепроверкой цены при финализации заказа

12.4. Ограничения модели

- Не учтены корреляция отказов (общий брокер, общая сеть), «эффект стада» при повторах и рост нагрузки при деградации; их проверяют chaos-тестами и нагрузочными испытаниями.
- Значения множителя хвоста, доступности зависимостей и лага событий приняты для иллюстрации; результат чувствителен к ним.

13. Экономика и целевые KPI (оценочная модель)

Показатель	Сейчас	Цель
p50 / p95 / p99 чекаута	640 / 1 400 / 3 200 мс	≤250 / ≤400 / ≤1 000 мс
Обязательных синхронных вызовов в критическом пути	8	3
Доступность чекаута	99,2% (≈5,7 ч/мес)	≥99,9% (≈43 мин/мес)
Каскадные инциденты	3 в квартал по ≈47 мин	≤1 в квартал, ≤10 мин
Кросс-командные релизы	35%	≤10%
Срок поставки изменения (lead time)	14 дней	≤7 дней
Перепродажа остатков	0,8% в асинхронном пилоте	≤0,05%
Конверсия чекаута	57,9%	≥60%

Сценарий	Потери и расходы, млн ₽/мес	Комментарий
А. Сейчас (синхронная цепочка)	≈4,5	Потеря маржи из-за конверсии ≈4,1 (≈16 000 заказов) + инциденты ≈0,4 (простой и разбор)
Б. Тесная связь (общая база)	≈0,8 + не монетизировано	Скорость восстановлена; инциденты в 2 раза дороже (принято); срок поставки 14 → 21 дня не оценён в деньгах
В. Полная асинхронность	≈2,6	Перепродажа ≈2,2 (2 400 заказов × 900 ₽) + ручная сверка ≈0,3 + инциденты ≈0,1
Г. Целевой гибрид	**≈0,8–2,1**	Остаток потерь по конверсии (доля 10–40%) + инциденты ≈0,1–0,15 + эксплуатация инфраструктуры ≈0,35

Статья	Оценка
Выигрыш Г против А	≈2,3–3,7 млн ₽/мес
Разовые затраты: 24 чел.-мес. × ≈0,3 млн ₽ (≈7,2) + инфраструктура и инструменты (≈1,8)	≈9,0 млн ₽
Срок окупаемости	≈2,5–3,8 месяца
Эффект на скорость поставки (lead time 14 → ≤7 дней)	не монетизирован: потенциал сверх оценки

Допущения модели (для иллюстрации): оборот 960 млн ₽/мес, маржа 8%, восстанавливается 60–90% потерь из-за падения конверсии с 61,0% до 57,9%; стоимость инцидентов: потерянная маржа за простой (≈0,08 млн ₽/мес) и разбор (120 инженеро-часов на инцидент × 2 500 ₽ ≈ 0,3 млн ₽/мес); инциденты сокращаются на 60–80%; стоимость одной перепроданной позиции 900 ₽ (возврат, компенсация, обработка). Для реальной платформы модель нужно пересчитать по собственным данным.

14. Дорожная карта внедрения

Этап	Действие	Срок	Эффект
1	Измерения: карта зависимостей критического пути, SLI/SLO чекаута, базовые p50/p95/p99, реестр инцидентов, конверсия по задержкам	2 недели	Базовая линия и целевые метрики
2	«Защитные оболочки»: пулы, тайм-ауты, circuit breaker, деградация; вывод лояльности и уведомлений из критического пути	3–5 недель	Локализация каскадных отказов
3	«Локальные модели чтения» для цен и акций: события, идемпотентность, метрика лага, перепроверка при финализации	6–8 недель, параллельно	Сокращение критического пути на 2–3 вызова
4	«Подготовка на клиентском времени»: предрасчёт корзины, антифрод-скоринг, прогрев, предавторизация; лимиты нагрузки	4 недели после этапа 3	Сокращение финального вызова до подтверждения
5	«Резервирование остатков и сага»: резерв с TTL, оркестрация шагов, компенсации, мониторинг саг	8–10 недель	Согласованность без перепродаж
6	Оценка перекройки границ по консистентности (контекст «Расчёт корзины») по результатам измерений	по готовности	Стратегическое упрощение

15. Как использовать подход в архитектурной работе

Образец ADR (запись архитектурного решения)

Раздел	Содержание
Название	ADR-014: локальные модели чтения для цен и акций в критическом пути чекаута
Контекст	Критический путь из 8 синхронных вызовов: p95 1,4 с, доступность 99,2%; полностью асинхронная схема приводит к перепродаже 0,8% заказов
Решение	Цены и акции читаются из локальной материализованной копии с устарелостью ≤5 с; остатки и платёж остаются в синхронном малом контуре; цена перепроверяется при финализации
Альтернативы	Общая база (отвергнуто: связь релизов); полностью асинхронный заказ (отвергнуто: перепродажа); кэш в шлюзе (частично: ограничение по инвалидации)
Последствия	Плюс: меньше сетевых вызовов и зависимостей. Минус: нужна идемпотентность обработки событий, метрика лага, дополнительное хранилище
Как проверим	Нагрузочный тест и chaos-тест: p95 ≤400 мс при отказе сервиса «Акции»; лаг событий p99 ≤2 с

Вопросы для архитектурного воркшопа

- Какие данные нужны для подтверждения заказа «здесь и сейчас», а какие могут подождать? Какова допустимая устарелость каждого?
- Где границы транзакций и консистентности и совпадают ли они с границами сервисов?
- Что произойдёт, если каждая зависимость откажет или замедлится в 10 раз? Какой запасной сценарий у неё есть?
- Сколько синхронных вызовов в критическом пути и каков вклад каждого в p95 и p99?
- Какие события уже публикуются и какие данные можно читать локально?
- Как расходуется бюджет ошибок и кто принимает решение о замораживании релизов?
- Какие ограничения нельзя нарушать (платёжные данные, аудит, регуляторные требования)?

Типичные ошибки: «распределённый монолит» (сервисы независимы формально, но связаны синхронными цепочками и общей базой); двойная запись в базу и брокер без outbox; повторы без идемпотентности; слепое доверие к eventual consistency там, где нужна строгая; выбор решения до измерения p99 и доли влияния каждого вызова. **Ограничения метода:** ТРИЗ-анализ структурирует поиск решений и даёт направления; выбор нужно подтверждать измерениями, нагрузочными и chaos-тестами.

16. Приложение: инструменты ТРИЗ, использованные в анализе

Матрица противоречий (по аналогии для ПО) | 40 принципов изобретательства | Су-польный анализ | 76 стандартов | Принципы разделения (время/место/условие/целое-часть) | Аналогии физических эффектов | ИКР (3 уровня) | Ресурсный чек-лист (6 типов) | Системный оператор | Законы эволюции систем | Адверсарная верификация | Многокритериальное ранжирование
