

Лекция 6: Арсенал ТРИЗ
15+ методов разрешения ИТ-противоречий
Курс «Основы инженерного творчества»

Прикладная математика и информатика

- ❶ Блок 1: Введение. Противоречие как ядро проблемы.
- ❷ Блок 2: Детальный разбор 15+ приемов ТРИЗ на IT-кейсах.
- ❸ Блок 3: Стандарты и ресурсы. Интеграция с матмоделированием и ИИ.
- ❹ Блок 4: Полный цикл анализа. Интеграция в проектную работу.

Блок 1: Введение

Эволюция методологии в курсе:

- ➊ **Этап 1:** Системный подход (Границы, над/подсистемы)
- ➋ **Этап 2:** Заинтересованные стороны (Конфликт требований)
- ➌ **Этап 3:** ФСА (Дисбаланс функций и затрат)
- ➍ **Этап 4:** ЗРТС (Вектор развития системы)
- ➎ **Этап 5:** Диалектика Гегеля (Философский базис)
- ➏ **Этап 6:** ТРИЗ Альтшуллера (Инструментальное ядро)

Ключевой переход:

От констатации противоречий (диалектика) → к алгоритму их разрешения (ТРИЗ).

Противоречие — ядро проблемы. Три уровня

Административное противоречие

“Хочу, чтобы модель была точной и быстрой”
(Проблема формулировки)

Техническое противоречие (ТП)

“Увеличение точности ML-модели ведет к росту вычислительной сложности”
(Конфликт параметров системы)

Физическое противоречие (ФП)

“Модель **должна** быть сложной (для точности) и **должна** быть простой (для скорости)”
(Требование взаимоисключающих свойств)

Тезис: Сильное решение = разрешение ФП, а не компромисс на уровне ТП.

Общая схема работы с противоречиями

- 1 **Выявить** противоречие (из этапов 2, ФСА)
- 2 **Сформулировать** Физическое Противоречие (ФП)
- 3 **Подобрать** прием(ы) из арсенала ТРИЗ
- 4 **Найти** конкретное IT-решение
- 5 **Оценить** новый уровень системы (ЗРТС)

Цель сегодняшней лекции

Освоить шаги 3 и 4 — научиться применять арсенал ТРИЗ для генерации решений.

Блок 2: Арсенал ТРИЗ — 15+ приемов

Прием 1: Дробление

Суть: Разделить систему на независимые части.

Диалектика: Переход количества (частей) в новое качество (гибкость).

IT-примеры

- **Архитектура:** Монолит → Микросервисы
- **Данные:** Шардирование базы данных
- **Вычисления:** Распараллеливание алгоритма (MapReduce)
- **MLOps:** Разделение пайплайна на этапы (preprocessing, training, serving)

Прием 2: Вынесение

Суть: Выделить "мешающую" или, наоборот, "нужную" часть системы.

Диалектика: Разделение целого на противоположности для снятия конфликта.

IT-примеры

- **Кеширование:** Вынос кеша в Redis/Memcached
- **Аутентификация:** Вынос в отдельный сервис (Auth0, Keycloak)
- **Статика:** Вынос на CDN
- **Вычисления:** Вынос тяжелых задач в background workers

Прием 3: Местного качества

Суть: Разные части системы должны быть оптимизированы под разные функции.

Диалектика: Единство и борьба противоположностей внутри системы.

IT-примеры

- **Базы данных:** Master (запись) / Replica (чтение)
- **Хранение:** Горячие данные → SSD, Холодные данные → HDD/S3
- **ML:** Разные модели для разных сегментов данных (кластеризация + регрессия)
- **Архитектура:** CQRS (Command Query Responsibility Segregation)

Прием 4: Асимметрия

Суть: Переход от симметричных структур и процессов к асимметричным.

Диалектика: Нарушение симметрии как источник развития.

IT-примеры

- **Безопасность:** Асимметричное шифрование (RSA)
- **Сети:** Асимметричные алгоритмы балансировки нагрузки
- **Данные:** Несимметричные структуры (JSON-API: запрос vs ответ)
- **Архитектура:** Event Sourcing (асимметрия чтения и записи состояния)

Прием 5: Объединение

Суть: Объединить однородные или смежные операции, объекты.

Диалектика: Синтез как переход на новый уровень системности.

IT-примеры

- **API:** API-шлюз (Aggregation Pattern)
- **Данные:** Data Lake (объединение разнородных данных)
- **Инфраструктура:** Kubernetes (объединение нод в кластер)
- **ML:** Ансамблирование моделей (bagging, boosting)

Прием 7: "Матрёшка"

Суть: Один объект размещен внутри другого, который, в turn, находится внутри третьего.

Диалектика: Бесконечное развитие вглубь.

IT-примеры

- **Виртуализация:** Docker внутри VM внутри физического сервера
- **Программирование:** Вложенные виртуальные окружения (venv)
- **UI/UX:** Композитные компоненты (React/Vue)
- **Модели:** Нейросеть с внутренними автоэнкодерами

Прием 10: Предварительное действие

Суть: Заранее выполнить требуемое действие (полностью или частично).

Диалектика: Снятие противоречия во времени.

IT-примеры

- **Компиляция:** AOT-компиляция (Ahead-of-Time)
- **Веб:** Статическая генерация сайтов (SSG)
- **Данные:** Предварительный расчет и кеширование агрегатов
- **ML:** Претренинг базовых моделей (BERT, GPT)

Прием 13: "Наоборот"

Суть: Вместо прямого действия выполнить обратное. Сделать неподвижную часть подвижной.

Диалектика: Отрицание отрицания.

IT-примеры

- **Коммуникация:** Push-уведомления вместо Pull-запросов
- **Сети:** Server-Sent Events (SSE) / WebSockets вместо polling
- **Поиск:** Инвертированные индексы (Elasticsearch)
- **Архитектура:** Event-Driven вместо Request-Response

Прием 24: "Посредник"

Суть: Использовать промежуточный объект, передающий или осуществляющий действие.

Диалектика: Введение третьего элемента для снятия конфликта двух.

IT-примеры

- **Мессенджинг:** Брокеры сообщений (Kafka, RabbitMQ)
- **Интеграция:** Паттерн "Адаптер"
- **Безопасность:** Прокси-сервер, API Gateway
- **ML:** Слой абстракции для разных фреймворков (ONNX)

Прием 25: Самообслуживание

Суть: Объект должен обслуживать себя сам, используя побочные ресурсы.

Диалектика: Развитие самости системы, переход к автономности.

IT-примеры

- **Инфраструктура:** Автоскейлинг (K8s HPA)
- **Надежность:** Circuit Breaker, самовосстановление
- **Память:** Сборка мусора (Garbage Collection)
- **ML:** AutoML, нейросети с механизмами внимания (self-attention)

Прием 28: Замена механической схемы

Суть: Заменить механическую схему на использующую поля (эл/магн, оптич).

Диалектика: Переход на новый энергетический уровень.

IT-примеры

- **Инфраструктура:** Замена физических серверов на облачные (виртуализация)
- **Железо:** SSD (электрическая память) вместо HDD (механика)
- **Вычисления:** GPU (массовый параллелизм) вместо CPU (последовательность)
- **Сети:** Беспроводные сети (Wi-Fi) вместо проводных

Прием 31: Пористые материалы

Суть: Сделать объект пористым или добавить пористые элементы.

Диалектика: Введение проницаемости, управляемости.

IT-примеры

- **Безопасность:** Фаерволы (контролируемая "проницаемость")
- **Архитектура:** Middleware, Interceptors ("пористые" слои)
- **Расширяемость:** Система плагинов
- **Данные:** Поточковая обработка (stream processing)

Прием 35: Изменение агрегатного состояния

Суть: Изменить фазовое состояние системы или ее части.

Диалектика: Скачок, качественное преобразование.

IT-примеры

- **Виртуализация:** Снейпшоты VM ("заморозка" состояния)
- **Кеши:** TTL ("испарение" данных)
- **Контейнеры:** Оркестрация (масштабирование как "фазовый переход")
- **Версионность:** Git branch и merge ("плавление" и "кристаллизация" кода)

Прием 2: Предварительное антидействие

Суть: Заранее совершить антидействие, чтобы нейтрализовать вредный фактор.

Диалектика: Опыт отрицания будущего вреда в настоящем.

IT-примеры

- **Безопасность:** Санитизация входных данных (против инъекций)
- **Надежность:** Chaos Engineering (предварительное введение сбоев)
- **Данные:** Валидация схемы данных до обработки
- **ML:** Аугментация данных для борьбы с переобучением

Суть: Использовать тепловое расширение/фазовые переходы.

Диалектика: Использование внутренней энергии системы для ее преобразования.

IT-примеры (абстрактные)

- **Тестирование:** Нагрузочное тестирование ("нагрев" системы)
- **Релизы:** Canary-развертывание ("постепенное охлаждение" риска)
- **Данные:** "Отжиг" при оптимизации гиперпараметров моделей
- **Архитектура:** "Плавление" монолита в микросервисы

Блок 3: Стандарты, ресурсы и ИИ

Стандарты ТРИЗ: готовые рецепты для систем

Стандарт на синтез вепольной системы

Проблема: Нужно обеспечить управление объектом.

Решение: Ввести поле, связывающее его с другим веществом.

IT-пример: Вместо прямого вызова сервиса → ввести брокер сообщений (поле).

Стандарт на разрушение вепольной системы

Проблема: Есть вредное взаимодействие между объектами.

Решение: Ввести 'ВЗ', нейтрализующий вред.

IT-пример: 'Злоумышленник (В1)' → 'Вредный запрос (П)' → 'Приложение (В2)'.

Вводим 'WAF (ВЗ)'.

Вещественно-полевые ресурсы (ВПР)

Идея: Использовать то, что уже есть в системе и ее окружении **бесплатно**.

Ресурсы вещества

- **Данные:** Логи, метаданные, кеш, "мусорные" данные
- **Код:** Старые версии, комментарии, документация

Ресурсы полей

- **Время:** Таймеры, крон-задачи, TTL
- **Информационные:** HTTP-заголовки, cookie, JWT-токены
- **Виртуальные:** Сессии, контексты исполнения, окружения

Ключевой вопрос архитектора

"Какие **бесплатные** ресурсы системы я еще не использовал?"

ТРИЗ задает *что* сделать, Матмодели/ИИ — *как* реализовать.

Связка ТРИЗ + AI/ML

- Прием **”Дробление”** + **ML**: Ансамбли моделей вместо одной монолитной.
- Прием **”Местного качества”** + **ML**: Кастомные модели для кластеров пользователей.
- Прием **”Самообслуживание”** + **AI**: Автоматический подбор гиперпараметров (AutoML).
- Прием **”Предварительное действие”** + **LLM**: Претренинг базовой модели для быстрой дообучения.
- Прием **”Посредник”** + **AI**: LLM как посредник для объяснения решений **”черного ящика”** (XAI).

Кино-кейс: противоречия в производстве фильма

Инженер и художник хотят «и этого, и того»:

- **Художественность**, но коммерческая успешность
- **Низкий бюджет**, но высокое качество эффектов
- **Быстрое производство**, но тщательная проработка
- **Инновационность**, но понятность для зрителя

Традиционный подход

Компромисс: ослабить одно требование ради другого («сделаем подешевле и попроще»).

ТРИЗ-подход

Разрешить противоречие **разделением**: разные части системы — разные цели. Например: ядро повествования — художественное, маркетинговая обёртка — коммерческая; ключевые сцены — качественные эффекты, второстепенные — дешёвые творческие заменители.

Приёмы ТРИЗ в кино: примеры из практики

- **Дробление (№1).** Не делать дорогим всё. «Парк Юрского периода»: динозавры на экране лишь несколько минут, остальное — напряжение, звук и монтаж. Погоня: одна дорогая деталь на 5 секунд (разбитая фара крупным планом), остальное — звук и быстрый монтаж.
- **Объединение (№5).** Один объект — много функций. Тот же лист: зелёный (лето) → тонирован жёлтым (осень) → вата и блёстки (зима). Три сезона за 0 рублей.
- **«Наоборот» (№13).** Не освещать темноту, а сделать её частью истории. Первая сцена «Пилы»: почти ничего не видно — только звуки и пятна света; страх дорисовывает сам зритель.
- **Предварительное действие (№10).** Сложное — заранее и отдельно. «Полёт по комнате»: фон снимается статично, актёры — на хромакее, склейка в монтаже. Проще: камера на офисном кресле = плавный проезд за 0 рублей.

Эффект

Смена вопроса: вместо «я не знаю, как снять» студент спрашивает «какой приём применить?» — интуиция и удача превращаются в систему и алгоритм.

ТРИЗ + LLM: приём превращает ассистента в эксперта

Без метода (LLM сам по себе):

- Запрос: «Как снять погоню без денег?»
- Ответ: стоковые кадры, снять на телефон, найти спонсора, упростить сценарий
- Результат: **общие советы**, склонность к галлюцинациям

С методом (приём «Дробление»):

- Запрос: «Примени приём “Дробление”: как снять погоню без денег?»
- Ответ: дорогие только 5 секунд (разбитая фара), остальное — звук и монтаж
- Результат: **конкретное рабочее решение**

Вывод

Знание приёмов ТРИЗ превращает LLM из «болтуна» в профессионального ассистента, а размер модели перестаёт быть решающим: локальная модель + метод = рабочий инструмент.

Блок 4: Интеграция в проект

Полный цикл на сквозном примере

Проблема (из ФСА): "Система рекомендаций должна быть персонализированной (сложные модели) и отдавать ответ <100мс (простая логика)".

❶ **ФП:** Модель **должна** быть сложной (качество) и **не должна** быть сложной (скорость).

❷ **Приемы:**

- **Вынесение:** Вынос инференса тяжелой модели в отдельный высокопроизводительный сервис.
- **Предварительное действие:** Предварительный расчет эмбедингов пользователей и товаров.
- **Местного качества:** Легкая модель (k-NN) для онлайн-выдачи, тяжелая (ANN) — для офлайн-пересчета.
- **Посредник:** Кеш-слой между API и моделями.

❸ **Решение:** Гибридная архитектура:
Офлайн-пайплайн (тяжелые модели) → Кеш эмбедингов → Онлайн-сервис (по кешу).

❹ **Уровень системы (ЗРТС):** Переход с уровня 2 (доработки) на уровень 3 (изменение архитектуры).

Интеграция с проектным циклом курса

- ❶ **Этап 2 (Стейкхолдеры):** Выявить конфликт требований → **ядро будущего ФП.**
- ❷ **Этап 3 (ФСА):** Найти самую дорогую/вредную функцию → **локализовать ФП.**
- ❸ **Этап 6 (Сейчас):** Сформулировать ФП → Применить арсенал ТРИЗ → Сгенерировать варианты решений.
- ❹ **Этап 4 (ЗРТС):** Оценить, на какой уровень развития переводит система выбранное решение.
- ❺ **Этап 5 (Диалектика):** Осознать выбранное решение как акт "снятия" противоречия.

Итог: Проектная работа становится полигоном для применения всего инструментария курса.

Задание

Применить арсенал ТРИЗ к вашему проекту для генерации архитектурных решений.

Шаги:

- 1 Выберите 1 ключевое противоречие из вашего ФСА или анализа стейкхолдеров.
- 2 Четко сформулируйте **Физическое Противоречие (ФП)**.
- 3 Используя таблицу приемов, выберите **3 наиболее релевантных приема**.
- 4 Для каждого приема предложите **конкретное архитектурное или техническое решение** для вашего проекта.
- 5 Оформите это как **1 слайд** в вашей итоговой презентации.

Критерий успеха: Не компромисс, а изящное разрешение ФП.

Типичные ошибки при применении ТРИЗ

Ошибка 1: Компромисс вместо решения

“Сделаем модель чуть проще и чуть быстрее”.

Правильно: Разделить противоречивые требования (Прием 3, 24).

Ошибка 2: Игнорирование ресурсов

“Разработаем сложный orchestration-сервис с нуля”.

Правильно: Использовать ВПР (кеш, заголовки, существующие сервисы).

Ошибка 3: Отрыв от диалектики

Рассматривать приемы как кулинарные рецепты.

Правильно: Понимать, что каждый прием — это акт диалектического “снятия”.

Заключение и выводы

- 1 **ТРИЗ** — это систематизированная эвристика, “таблица Менделеева” для инженера-творца.
- 2 Сильное решение рождается не из компромисса, а из **разрешения Физического Противоречия**.
- 3 Связка “**Диалектический анализ** → **Формулировка ФП** → **Применение приема ТРИЗ**” — ключевой навык для архитектора и руководителя НИОКР.
- 4 Арсенал ТРИЗ естественно интегрируется с современными подходами (**AI/ML, матмоделирование**), предоставляя для них **архитектурный каркас**.

Финальный тезис

Ваша ценность как будущего системного архитектора или руководителя — не в знании последнего фреймворка, а в способности видеть системные противоречия и владеть методологией их элегантного разрешения.

Блок 5: Полный перечень 40 приемов ТРИЗ

Полный перечень 40 приемов ТРИЗ (1/5)

- ❶ Прием 1: Дробление
- ❷ Прием 2: Вынесение
- ❸ Прием 3: Местного качества
- ❹ Прием 4: Асимметрия
- ❺ Прием 5: Объединение
- ❻ Прием 6: Универсальности
- ❼ Прием 7: "Матрёшка"
- ❽ Прием 8: Предварительное нагружение

- 9 Прием 9: Предварительное антидействие
- 10 Прием 10: Предварительное действие
- 11 Прием 11: "Заранее подложенная подушка"
- 12 Прием 12: Эквипотенциальность
- 13 Прием 13: "Наоборот"
- 14 Прием 14: Сфероидальность
- 15 Прием 15: Динамичности
- 16 Прием 16: Частичного или избыточного действия

- 17 Прием 17: Переход в другое измерение
- 18 Прием 18: Использование механических колебаний
- 19 Прием 19: Периодическое действие
- 20 Прием 20: Непрерывность полезного действия
- 21 Прием 21: Проскок
- 22 Прием 22: "Обратить вред в пользу"
- 23 Прием 23: Обратная связь
- 24 Прием 24: "Посредник"

- 25 Прием 25: Самообслуживание
- 26 Прием 26: Копирование
- 27 Прием 27: Дешевая недолговечность взамен дорогой долговечности
- 28 Прием 28: Замена механической схемы
- 29 Прием 29: Использование пневмо- и гидроконструкций
- 30 Прием 30: Использование гибких оболочек и тонких пленок
- 31 Прием 31: Применение пористых материалов
- 32 Прием 32: Принцип изменения окраски

Полный перечень 40 приемов ТРИЗ (5/5)

- 33 Прием 33: Принцип однородности
- 34 Прием 34: Принцип отброса и регенерации частей
- 35 Прием 35: Изменение агрегатного состояния
- 36 Прием 36: Применение фазовых переходов
- 37 Прием 37: Применение теплового расширения
- 38 Прием 38: Использование сильных окислителей
- 39 Прием 39: Применение инертной среды
- 40 Прием 40: Применение композитных материалов

Дополнительные приемы (более 40)

- 41 Прием 41: Разделение противоречивых свойств во времени
- 42 Прием 42: Разделение противоречивых свойств в пространстве
- 43 Прием 43: Разделение противоречивых свойств между частями системы
- 44 Прием 44: Переход к другому уровню системы
- 45 Прием 45: Использование физических эффектов
- 46 Прием 46: Использование химических эффектов
- 47 Прием 47: Использование геометрических эффектов
- 48 Прием 48: Использование биологических эффектов

Вопросы?

Следующая лекция
Когнитивная СППР

Контакты и материалы

- Материалы курса: <https://edu.irnok.net/doku.php?id=ec:main>
- Следующая встреча: 2026

Противоречие как инструмент анализа проблемы

Лестница уровней: приём → формулировка проблемы → выбор решения

Три ступени

- ❶ **Базовый:** противоречие, приёмы, ИКР (ядро лекции)
- ❷ **Средний:** от противоречия — к формулировке проблемы и требований
- ❸ **Продвинутый:** классификация проблем и выбор решения по близости к ИКР

Связка

ТРИЗ — это не только приёмы для инженеров, но и язык **анализа проблем**:
«сформулируй противоречие» = «пойми суть проблемы».

От противоречия к требованию

Средний уровень: проблема и требования — две стороны противоречия

Приём формулировки

- Плохое требование: «сделайте удобную кнопку» (решение, а не проблема)
- Хорошая постановка: **что мешает пользователю** и почему (противоречие)
- Требование — это «снятие» противоречия: ожидаемое свойство + ограничение

Откуда берутся требования

- Конфликт интересов стейкхолдеров → противоречие требований
- Функция, которую хочет пользователь, против затрат на неё
- ТЗ фиксирует требования; противоречия разрешаются методами курса (лекция 8 — как оформить ТЗ)

Бизнес-проблема или техническая проблема?

Продвинутый уровень: классифицируем и представляем проблему

Классификация проблем

- **Явная** — сформулирована заказчиком; **скрытая** — ещё не названа, её выявляют анализом
- **Бизнес-проблема** — на уровне целей и процессов; **техническая** — на уровне реализации
- Одна бизнес-проблема часто скрывает несколько технических противоречий (и наоборот)

Представление разным сторонам

Для менеджмента — бизнес-эффект и риски; для инженеров — противоречие и ограничения; для владельца продукта — ценность для пользователя. Формат зависит от аудитории.

Пример

«Падает конверсия» (бизнес) → противоречие «много шагов в оформлении / много данных для проверки» (техническое) → решение: разделить обязательные и необязательные шаги (приём «разделение противоречивых свойств»).

ИКР как критерий выбора решения

Продвинутый уровень: оцениваем варианты по идеальности

Идеальный конечный результат

- ИКР: функция выполняется сама, без затрат и вредных эффектов
- Варианты решения сравниваются по **близости к ИКР**: меньше ресурсов, меньше вреда, больше пользы
- Идеальность = польза / (затраты + вред) — созвучна ФСА (ценность против затрат)

Практический критерий отбора

Если несколько решений «работают», выбирают то, что ближе к идеалу и использует **имеющиеся ресурсы** (вещественно-полевые, информационные, временные) — а не добавляет новые сущности.

Связь с архитектурой

Выбор между монолитом и микросервисами — это выбор «идеальности» под текущие цели: минимально достаточная сложность, а не максимальная.