

Законы развития технических систем

Лекция 4: От анализа к прогнозированию развития

Курс «Основы инженерного творчества»

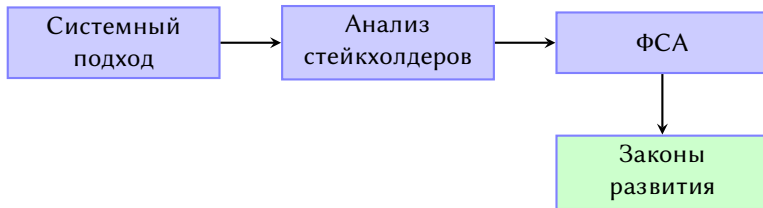
Прикладная математика и информатика

Что уже пройдено:

- Системный подход и границы систем
- Теория заинтересованных лиц
- Функционально-стоимостный анализ

Сегодня: Законы развития технических систем

- Почему системы развиваются закономерно
- Классификация законов развития
- Практическое применение в IT-проектах
- Анализ ваших школьных проектов



Методология инженерного творчества

- **Системный подход:** Понимаем границы и структуру системы
- **Анализ стейкхолдеров:** Определяем потребности и противоречия
- **ФСА:** Анализируем функции и их стоимость
- **Законы развития:** Прогнозируем и направляем эволюцию системы

Уровни развития технических систем

Уровень 1: Косметические улучшения

Исправление багов, улучшение интерфейса: 10-100 проб

Уровень 2: Значимые доработки

Добавление новых функций, оптимизация: 100-1,000

Уровень 3: Изменение архитектуры

Перепроектирование системы, смена технологий: 1,000-10,000

Уровень 4: Смена концепции

Принципиально новый подход к решению: 10,000-100,000

Уровень 5: Принципиально новое решение

Создание новой парадигмы, прорывная технология: 100,000+

Почему законы развития важны?

Проблема метода проб и ошибок

- Уровень 1: 10-100 проб
- Уровень 2: 100-1,000 проб
- Уровень 3: 1,000-10,000 проб
- Уровень 4: 10,000-100,000 проб
- Уровень 5: 100,000+ проб

Решение

Знание законов развития позволяет
целенаправленно создавать решения, а не искать их перебором

Классификация законов развития

Статические законы

Условия жизнеспособности системы

Кинетические законы

Законы развития во времени

Динамические законы

Законы качественных переходов

Классика Альтшуллера и обновления Петрова

Г. С. Альтшуллер — классика (ТРИЗ/ТРТС)

- Автор ТРИЗ и классических **законов развития технических систем**
- Три группы законов: **Статика, Кинематика, Динамика**
- Формулировки выросли из техники XX века

В. Петров — современные формулировки

- Специалист по ТРИЗ; работы по истории и развитию законов развития систем
- В лекции его формулировки отмечены как «**Обновление Петрова**»
- Добавляют потоки (вещество, энергия, информация), управляемость, «интеллектуальные поля»

Зачем два уровня

Классика даёт **каркас** законов, обновления Петрова приближают их к современным системам, включая ИТ.

Критерии жизнеспособности

Если эти законы нарушены — система неработоспособна

Закон 1. Полноты частей системы

Формулировка

- **Классика:** Чтобы система выжила, у неё должны быть четыре части: двигатель, трансмиссия, рабочий орган и орган управления
- **Обновление Петрова: Закон минимального обеспечения функциональности**
- Система должна обеспечивать сквозной проход как минимум трёх потоков: **вещества, энергии и информации**

IT-примеры

- Веб-приложение: данные (вещество) → обработка (энергия) → API (информация)
- Микросервис: запрос → бизнес-логика → ответ
- База данных: хранение → индексация → запросы

Закон 2. Энергетической проводимости системы

Формулировка

- **Классика:** Энергия должна свободно проходить через все части системы к рабочему органу
- **Обновление Петрова: Закон проводимости потоков**
- Развитие требует проводимости не только энергии, но и вещества (сырья) и информации

IT-примеры

- **Энергия:** Электричество в ЦОД → вычислительная мощность
- **Вещество:** Данные в хранилище → обработка в памяти
- **Информация:** API-запросы → бизнес-логика → ответы

Тренд развития

Система эволюционирует в сторону замены "дорогих" видов энергии (механической) на "дешёвые" (электромагнитные, полевые)

Закон 3. Согласования ритмики частей системы

Формулировка

- **Классика:** Части системы должны работать согласованно, чтобы не разрушать друг друга
- **Обновление Петрова: Закон динамического согласования**
- Согласование бывает статическим (раз и навсегда) и динамическим (подстраивающимся под режим работы)

IT-примеры

- **Статическое:** Фиксированные интервалы опроса БД
- **Динамическое:** Автомасштабирование по нагрузке
- **Рассогласование:** Асинхронная обработка через очереди

Современный тренд

Современные системы переходят к **динамическому согласованию и рассогласованию**

Как системы растут и развиваются независимо от внешних условий

Закон 4. Повышения идеальности

Формулировка

Система стремится к состоянию, когда функции выполняются, а самой системы нет

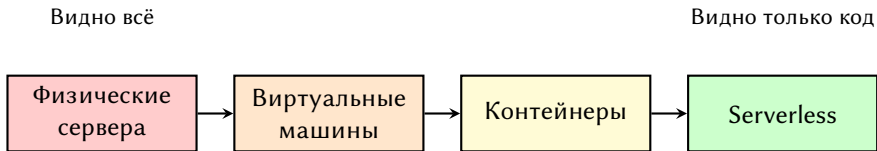
IT-примеры

- Физические сервера → Виртуальные машины → Контейнеры → Serverless
- Ручное развертывание → Автоматическое → GitOps
- Локальные файлы → Облачные хранилища → Поточковые сервисы

Суть

Чем ближе система к идеалу, тем меньше материальных компонентов требуется для выполнения её функций

Визуализация закона идеальности



Ваш проект

Как можно повысить идеальность вашего школьного проекта?

- Что можно убрать, сохранив функциональность?
- Какие процессы можно автоматизировать?
- Какие ресурсы можно виртуализировать?

Закон 5. Неравномерности развития частей системы

Формулировка

- **Классика:** Части системы стареют с разной скоростью, из-за чего возникают противоречия
- **Обновление Петрова: Закон перехода функции**
- Самый "узкий" элемент (слабое звено) со временем либо исчезает, передавая свою функцию соседнему элементу, либо переходит на микроуровень для ускорения работы

Закон 5. Неравномерности развития частей (примеры)

IT-примеры

- **Узкое место:** База данных не справляется с нагрузкой
- **Переход функции:** Кэширование переносит часть нагрузки на Redis
- **Микроуровень:** Шардирование БД на множество мелких инстансов

Связь с противоречиями

Этот закон объясняет, почему в системах возникают технические противоречия — разные части развиваются в разном темпе

Закон 6. Перехода в надсистему

Формулировка

- **Классика:** Исчерпав ресурсы, система объединяется с другими в надсистему
- **Обновление Петрова: Закон моно-би-поли-свертывание**
- Путь развития: Моносистема → Бисистема → Полисистема → Свернутая система

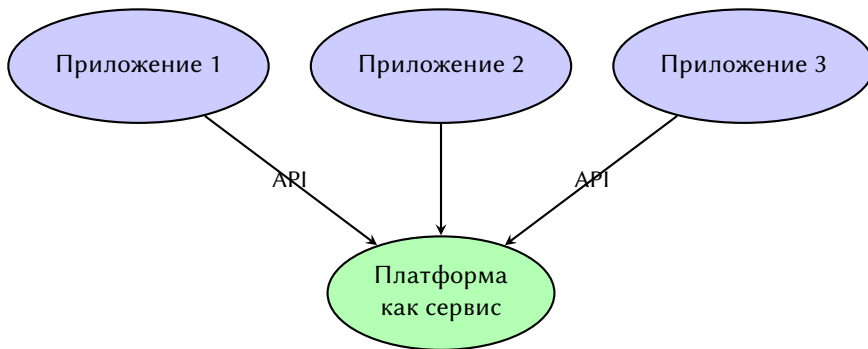
IT-примеры цикла развития

- **Моносистема:** Отдельный сервер с одним приложением
- **Бисистема:** Веб-сервер + база данных
- **Полисистема:** Микросервисная архитектура (10+ сервисов)
- **Свернутая система:** Serverless-функции (единая платформа)

Современный тренд

Современные IT-системы проходят полный цикл: от монолита к микросервисам и обратно к "свернутым" платформам

Визуализация закона перехода в надсистему



Примеры перехода

- Отдельные приложения → Единая платформа (AWS, Azure, GCP)
- Локальные сервисы → Облачная экосистема
- Изолированные системы → API-экономика

Дополнение (вне 3×3): согласование и рассогласование

Формулировка

Развитие системы происходит через циклы согласования и рассогласования компонентов

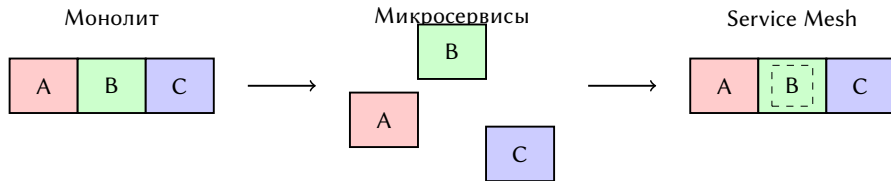
IT-примеры

- Монолитное приложение → Микросервисы → Service Mesh
- Жесткая схема БД → Гибкая схема → Schemaless
- Синхронная коммуникация → Асинхронная → Event-driven архитектура
- Жестко связанные компоненты → Слабая связность → Управляемая связность

Динамика развития

Каждый этап рассогласования создает противоречия, которые решаются переходом на новый уровень согласования

Дополнение: визуализация согласования-рассогласования



Эволюция связности

- **Жесткая связность** → **Слабая связность** → **Управляемая связность**
- Каждый этап решает проблемы предыдущего

Как системы подстраиваются под внешний мир на современном этапе

Закон 7. Перехода с макроуровня на микроуровень

Формулировка

- **Классика:** Рабочий орган дробится: от лопаты к струе воды, к лучу лазера, к отдельным атомам
- **Обновление Петрова: Закон повышения управляемости**
- Переход на микроуровень (поля, частицы) всегда сопровождается ростом **динамичности и управляемости** системы

Закон 7. Перехода с макроуровня на микроуровень (примеры)

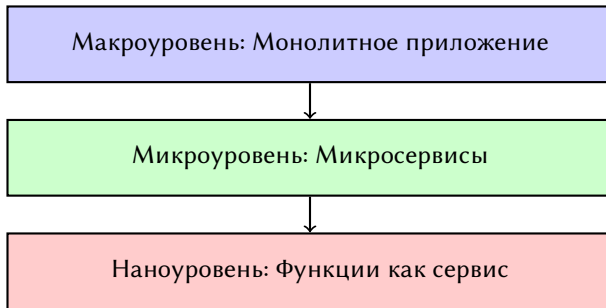
IT-примеры

- **Макро:** Физические сервера (управляем вручную)
- **Микро:** Виртуальные машины (управляем через гипервизор)
- **Нано:** Контейнеры (управляем оркестратором)
- **Пико:** Serverless-функции (управляем событиями)

Ключевой принцип

Мы управляем уже не лопатой, а магнитным полем. В IT: не серверами, а конфигурациями и политиками

Визуализация закона перехода на микроуровень



Примеры в IT

- Виртуальные машины → Контейнеры → Unikernels
- Монолитные БД → Шардирование → Distributed SQL
- Веб-приложения → SPA → Микро-фронтенды

Закон 8. Увеличения степени вепольности

Формулировка

- **Классика:** Невепольные системы (просто вещество) стремятся стать вепольными (вещество + поле)
- **Обновление Петрова:** Закон перехода к интеллектуальным полям
- Развитие идёт от грубых механических полей к сложным (электромагнитным, оптическим, биологическим)

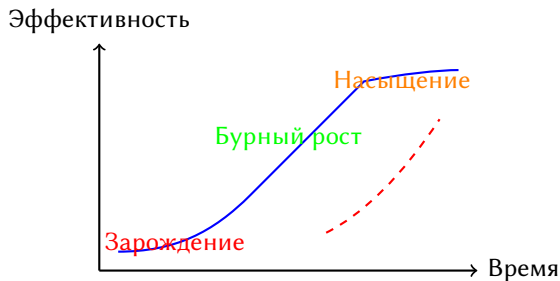
IT-примеры эволюции полей

- **Механическое:** Физические переключатели, кнопки
- **Электрическое:** Сенсорные экраны, мыши
- **Электромагнитное:** Беспроводные сети, Bluetooth
- **Интеллектуальное:** Голосовые помощники, жестовое управление

Современный тренд

Системы учатся "чувствовать" окружающую среду и адаптироваться к ней через интеллектуальные поля (AI/ML)

Закон 9. S-образного развития



IT-примеры

- Технологии: jQuery → Angular → React → Next.js
- Базы данных: SQL → NoSQL → NewSQL
- Архитектура: Монолиты → Микросервисы → Serverless

☒ Бонус: Тренды современного развития (по Петрову)

1. Повышение динамичности

- От жёсткого к гибкому и меняющему форму
- **IT-пример:** Монолиты → Микросервисы → Serverless

2. Управление потоками

- Система учится управлять веществом, энергией и информацией одновременно
- **IT-пример:** Data pipelines, ETL-процессы, потоковая аналитика

3. Автоматизация и вытеснение человека

- Человек только ставит задачу, система выполняет
- **IT-пример:** CI/CD, автотестирование, AI-ассистенты в разработке

Итог

Современные IT-системы развиваются по всем девяти законам (3×3: Статика, Кинематика, Динамика) одновременно, создавая интеллектуальные, самоуправляемые экосистемы

Технические и физические противоречия

Техническое противоречие (ТП)

Улучшение одной части системы ухудшает другую: «если усилить X, то ослабнет Y».

- Быстрая проверка ИИ ↔ глубина экспертной оценки (кейс ЛР по БД)
- Качество визуальных эффектов ↔ ограниченный бюджет (кино-кейс)

Физическое противоречие (ФП)

К одному элементу предъявляются противоположные требования: «элемент должен быть X, чтобы ..., и не-X, чтобы ...».

- Проверка должна быть **автоматической** (чтобы быстро) и **не автоматической** (чтобы сохранить контроль эксперта)

Как разрешать

ТП и ФП не сводятся к компромиссу «поделить недостатки» — их **снимают** (диалектика — лекция 5, приёмы ТРИЗ — лекция 6), получая новое качество.

Уровни развития технических систем

Уровень 1: Косметические улучшения

Исправление багов, улучшение интерфейса

Уровень 2: Значимые доработки

Добавление новых функций, оптимизация

Уровень 3: Изменение архитектуры

Перепроектирование системы, смена технологий

Уровень 4: Смена концепции

Принципиально новый подход к решению

Уровень 5: Принципиально новое решение

Создание новой парадигмы, прорывная технология

Шаблон анализа вашего проекта

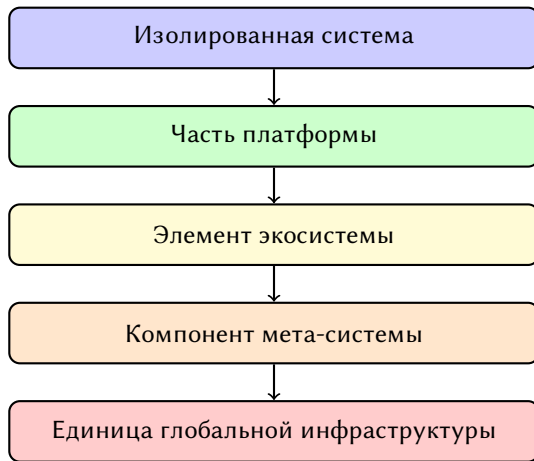
Определите уровень вашего проекта

- Какой уровень соответствует текущему состоянию?
- Какие характеристики это подтверждают?
- Каков потенциал развития?

Примените законы ЗРТС для анализа

- **Статика (1-3):** Есть ли все необходимые потоки (вещество, энергия, информация)?
- **Идеальность (4):** Что можно убрать/автоматизировать?
- **Неравномерность (5):** Где самое слабое звено в системе?
- **Надсистема (6):** Как стать частью большего целого?
- **Микроуровень (7):** Можно ли перейти на более глубокий уровень управления?
- **Вепольность (8):** Как добавить "интеллектуальные поля" в систему?
- **S-кривая (9):** На какой стадии зрелости система и что придёт ей на смену?

Закон перехода в надсистему: Иерархия интеграции



Примеры перехода в надсистему

Эволюция интеграции

- **Изолированная система:** Мое standalone приложение
- **Часть платформы:** Приложение с API в облаке AWS/Azure
- **Элемент экосистемы:** Сервис в микросервисной архитектуре
- **Компонент мета-системы:** Часть глобальной data mesh архитектуры
- **Единица глобальной инфраструктуры:** Элемент интернета вещей или edge computing сети

Куда дальше?

После глобальной инфраструктуры система может стать частью **межпланетной коммуникационной сети** или **искусственного интеллекта планетарного масштаба**

Пример анализа: Мобильное приложение учета финансов

Текущее состояние

- **Уровень:** 2-3 (базовые функции + аналитика)
- **Основные функции:** учет операций, категоризация, отчеты
- **Проблемы:** ручной ввод, изолированность

Анализ по ЗРТС

- **Статика (1-3):** Есть потоки данных, но нет автоматической синхронизации
- **Идеальность (4):** автоматическое категорирование через ИИ
- **Неравномерность (5):** ручной ввод - самое слабое звено
- **Надсистема (6):** интеграция с банковскими API и финансовыми экосистемами
- **Микроуровень (7):** переход к микросервисной архитектуре для масштабирования
- **Вепольность (8):** добавление AI-аналитики и прогнозирования

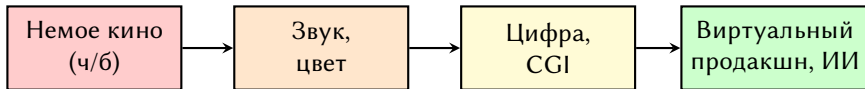
Проектирование перехода на следующий уровень



Ключевые вопросы

- Какие изменения необходимы для перехода?
- Какие ресурсы потребуются?
- Какие препятствия могут возникнуть?
- Какой ожидается эффект?

Кино-кейс: эволюция кинотехнологий по законам ЗРТС



Движущие законы ЗРТС

- **S-кривая:** каждая кинотехнология проходит цикл «становление → расцвет → насыщение»
- **Идеальность:** от дорогой плёнки и громоздких камер к цифре и пост-обработке на компьютере
- **Переход в надсистему:** отдельный фильм → киноиндустрия, стриминг, экосистемы контента
- **Микроуровень:** от трюков и макетов к CGI и генеративному ИИ

Кино-кейс: прогноз развития для студенческого фильма

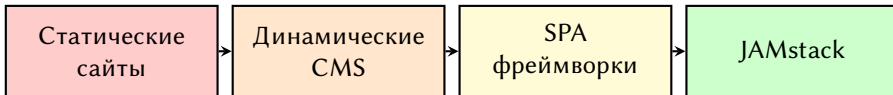
Текущие тренды (по Петрову, 2020-е)

- **Виртуальный продакшн:** LED-стены и real-time рендеринг заменяют натурные съёмки
- **ИИ-генерация:** черновики сцен, фонов и даже спецэффектов «за 0 рублей»
- **Иммерсивность:** от пассивного просмотра к интерактивным и AR/VR-форматам

Вопросы для применения к фантастическому фильму

- Какие из этих технологий **уже сегодня доступны** при ограниченном бюджете?
- Где лежит «**верхняя точка**» **S-кривой** текущей технологии, и что придёт ей на смену?
- Как заложить **потенциал развития** (переход на следующий уровень), не переделывая проект?

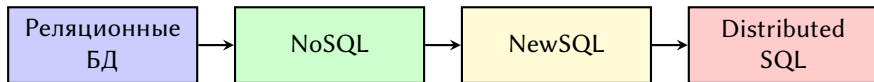
Эволюция веб-разработки по законам ЗРТС



Применение законов

- **Идеальность:** от серверного рендеринга к статической генерации
- **Надсистема:** интеграция с CDN и headless CMS
- **Микроуровень:** компонентный подход → микро-фронтенды

Эволюция баз данных



Шардирование

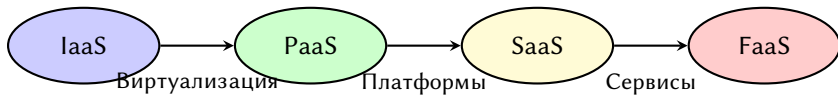
Репликация

Кэширование

Движущие законы

- **S-кривая:** Каждая технология проходит полный цикл
- **Согласование:** ACID vs BASE, Consistency vs Availability
- **Идеальность:** Managed services, авто-масштабирование

Развитие облачных технологий



Проявление законов

- **Идеальность:** От управления инфраструктурой к написанию кода
- **Надсистема:** Интеграция в экосистемы (AWS, Azure, GCP)
- **Микроуровень:** Контейнеры → функции → nanoservices

Сквозной пример: Автоматизация проверки ЛР по БД

Исходное состояние

- **Уровень:** 2 (ручная проверка с шаблонами)
- **Процесс:** преподаватель → студент (1-3 дня ожидания)
- **Проблемы:** высокая нагрузка, субъективность, задержки

Анализ по ЗРТС

- **Идеальность:** проверка есть, а времени преподавателя нет
- **Надсистема:** интеграция в образовательную платформу
- **Согласование:** баланс между автоматизацией и качеством

Эволюция системы проверки ЛР



Проявление законов развития

- **Идеальность:** преподаватель освобождается от рутины
- **Переход в надсистему:** ИИ становится частью процесса обучения
- **Согласование:** баланс скорости и качества проверки

Конкретные улучшения через ЗРТС

Закон идеальности в действии

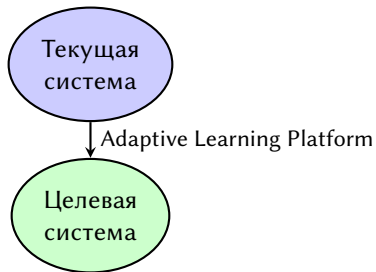
- **Было:** преподаватель тратит 45 мин на проверку
- **Стало:** ИИ проверяет за 5 мин + эксперт 10 мин
- **Идеал:** мгновенная проверка с гарантией качества

Закон перехода в надсистему

- **Было:** изолированная проверка работ
- **Стало:** часть образовательной платформы
- **Будет:** интеграция с системами тестирования и рейтинга

Проектирование перехода на уровень 4

Уровень 3



Уровень 4

Концепция уровня 4: "Adaptive Learning Platform"

- Персонализированные задания на основе пробелов
- Непрерывная оценка компетенций
- AI-тьютор с индивидуальными траекториями
- Интеграция с промышленными кейсами

Выявленные противоречия для следующей лекции

Технические противоречия

- **Скорость vs Качество:** быстрая проверка ИИ vs глубина экспертной оценки
- **Автоматизация vs Контроль:** автономность системы vs управление процессом
- **Стандартизация vs Индивидуальность:** единые критерии vs персональный подход

Для домашнего задания

Сформулируйте аналогичные противоречия для вашего проекта, мешающие переходу на следующий уровень

Законы развития как инструмент выбора и прогноза

Лестница уровней: закон → применение → стратегия

Три ступени

- 1 **Базовый:** законы и S-кривая (ядро лекции)
- 2 **Средний:** зрелость технологии и момент её внедрения
- 3 **Продвинутый:** анализ возможностей автоматизации и прогноз развития для архитектуры

Связка

Законы развития — не «теория ради теории», а инструмент практических решений: какую технологию взять, когда и зачем.

S-кривая и выбор технологии

Средний уровень: оцениваем зрелость и момент внедрения

Стадии S-кривой

- **Зарождение:** дорого, нестабильно, высокий риск — выгодно для лидеров, дорого для остальных
- **Рост:** технология дешевеет, появляются стандарты — лучший момент для большинства
- **Зрелость:** эффективность растёт медленно — новые решения вытесняют старые

Применение к ИИ-решениям

Задачи ИИ тоже находятся на своих S-кривых: прежде чем внедрять, оценивают **зрелость** технологии и данных (осуществимость), а не только «модность».

Законы развития и возможности автоматизации

Средний/продвинутый уровень: находим, что упрощать и автоматизировать

Как работают законы

- **Идеальность:** система стремится к состоянию, когда функция выполняется «сама» → кандидат на автоматизацию
- **Свёртывание** и переход функций к надсистеме → ищем, что можно вынести или объединить
- **Динамизация** → гибкость вместо жёсткости (процессам — адаптивность)

Вопрос для анализа процессов

Где функция выполняется вручную, повторяется и требует ресурсов? Такие места — первичные кандидаты на оптимизацию или автоматизацию программными средствами.

Пример курса

Эволюция веб-разработки (лекция) — наглядная S-кривая: от статичных страниц к адаптивным платформам, где часть функций «свёрнута» в готовые сервисы.

Прогноз развития как аргумент для архитектуры

Продвинутый уровень: перспективы развития закладываем заранее

Приём

- По законам развития прогнозируем, **куда** пойдёт система (моно → би → поли → свёртывание; рост идеальности)
- «Перспективы развития и модернизации» фиксируются в ТЗ и архитектуре — чтобы не переделывать «с нуля» через год
- Пример: сразу закладываем возможность масштабирования и интеграций, ожидая роста (см. лекции 1 и 8)

Вывод

Умение видеть траекторию развития системы — аргумент в споре об архитектуре и технологиях: решения обосновываются не вкусом, а закономерностями.

- **Г. С. Альтшуллер** — изобретатель, автор ТРИЗ и классических законов развития технических систем; выделил три группы законов: **Статика, Кинематика, Динамика**.
- **В. Петров** — современные формулировки законов и история их разработки:
<http://www.trizminsk.org/e/23111.htm>

Что мы узнали

- Технические системы развиваются по объективным законам — в трёх блоках:
Статика, Кинематика, Динамика
- Законы объясняют эволюцию: идеальность, неравномерность развития, переход в надсистему и на микроуровень, вепольность
- **S-кривая** показывает стадии зрелости технологии и момент её внедрения
- Законы универсальны — работают и в кино, и в IT (кино-кейс, проверка ЛР по БД)

Практическая ценность

- Прогноз развития вместо перебора: что взять, когда и зачем
- Обоснование архитектуры и технологий закономерностями, а не вкусом
- Кандидаты на автоматизацию: где функция делается вручную и повторяется
- Понимание природы противоречий — мостик к следующей лекции

Что мешает развитию проекта дальше?

Мы знаем КУДА развиваться, но...

- **Закон идеальности** говорит: "убери лишнее"
- **Но на практике:** убрав один элемент → ломается другой
- **Пример:** Хотим убрать ручную проверку (идеальность)
→ но теряем контроль качества (противоречие!)

Ключевой вопрос следующей лекции

Как преодолевать технические противоречия, которые возникают при развитии систем по законам ЗРТС?

Уже сейчас подумайте:

Какие **противоречия** мешают переходу вашего проекта на следующий уровень?

Домашнее задание

Анализ вашего школьного проекта

- Определите текущий уровень развития (1-5)
- Примените **все девять законов ЗРТС (3×3: Статика, Кинематика, Динамика)** с учётом современных уточнений
- Спроектируйте переход на следующий уровень
- Сформулируйте основные противоречия

Формат отчета

- Описание проекта и текущего состояния
- **Анализ по блокам:** Статика, Кинематика, Динамика
- Концепция развития с применением законов 1–9
- Выявленные противоречия и пути их разрешения

Дедлайн

Следующий семинар: готовность к обсуждению вашего анализа

Вопросы?

Следующая лекция
Феномен противоречия



Контакты

- План работ: <https://edu.irnok.net/doku.php?id=ec:project>
- Следующая встреча: *23 октября 2026*