

Лекция 1: Системный подход

Курс «Основы инженерного творчества»

Прикладная математика и информатика

Направление «Искусственный интеллект и системная аналитика»

Что мы уже умеем

На вводной лекции мы собрали метод: **переформулировать задачу** → **найти ресурсы** → **собрать архитектуру** и договорились, что **ошибка** — это **данные** для следующего шага.

Что добавляет эта лекция

Первый инструмент архитектора — умение видеть **систему**: целое, части и связи между ними. Все последующие темы курса будут опираться на этот взгляд.

Введение в системное мышление для ИТ-специалистов

Системный подход как фундамент профессий

- Системный аналитик
- Бизнес-аналитик
- Архитектор решений

Отличие системного мышления от аналитического

- **Системное мышление:** целое vs части
- **Аналитическое мышление:** разложение на составляющие

Роль системного подхода

- Создание новых рынков
- Научные проекты
- Программные системы

Системный подход: определение

Что это

Системный подход — направление методологии научного познания, в основе которого лежит рассмотрение объекта как **системы**: целостного комплекса взаимосвязанных элементов (И. В. Блауберг, В. Н. Садовский, Э. Г. Юдин).

Другие формулировки

- совокупность взаимодействующих объектов (Л. фон Берталанфи);
- совокупность сущностей и отношений (А. Д. Холл, Р. И. Фейджин).

Суть

Объект рассматривается **одновременно** как система и как элемент более общей системы; системный подход — форма приложения теории познания и диалектики к природе, обществу и мышлению.

Пять принципов

- **Целостность:** система — единое целое и в то же время подсистема вышестоящего уровня.
- **Иерархичность:** минимум два уровня, подчинение низшего высшему (управляющая и управляемая подсистемы).
- **Структуризация:** поведение системы определяется не столько свойствами элементов, сколько свойствами **структуры**.
- **Множественность:** систему можно описывать множеством моделей (кибернетическими, экономическими, математическими).
- **Эмерджентность:** у целого есть признаки, которых нет у элементов (подробно — ниже).

Понятие	Определение
Система	Совокупность элементов, действующих вместе как целое и выполняющих определённую функцию
Структура	Способ взаимодействия элементов посредством связей (картина связей и их устойчивость)
Процесс	Динамическое изменение системы во времени
Функция	Работа элемента в системе
Состояние	Положение системы относительно других её положений
Системный эффект	Результат реорганизации элементов, когда целое становится больше простой суммы частей
Структурная оптимизация	Итерационный процесс получения серии системных эффектов в рамках заданных ограничений

Ключевые имена

- **А. А. Богданов** — тектология (всеобщая организационная наука)
- **Л. фон Берталанфи** — общая теория систем
- **А. Д. Холл, Р. И. Фейджин** — системотехника и системные исследования
- **У. Р. Эшби** — кибернетика и самоорганизация
- **Г. Саймон, Р. Акофф** — системный анализ, теория решений
- **П. Друкер, А. Чандлер** — системный взгляд в управлении
- **Дж. Форрестер, Д. Медоуз** — системная динамика

Истоки

Теоретическое обоснование метод получил в 1950–1960-е годы; сегодня системный принцип лежит в основе большинства наук и инженерных дисциплин.

Аксиомы системного подхода

- ❶ Системы существуют.
- ❷ Системное представление истинно.
- ❸ Системы взаимодействуют друг с другом и потому взаимосвязаны.
- ❹ Системы состоят из элементов, действующих как целое, — в этом их сущность.
- ❺ Любой элемент можно представить как систему; любую систему — как элемент вышестоящей.
- ❻ Мир выразим в понятиях системного представления.

Практический смысл

Раз любой элемент — тоже система, декомпозицию можно **останавливать там, где удобно**, и углубляться только по необходимости.

Модель OSI/ISO как эталон системного проектирования

Три ключевых элемента системного проектирования

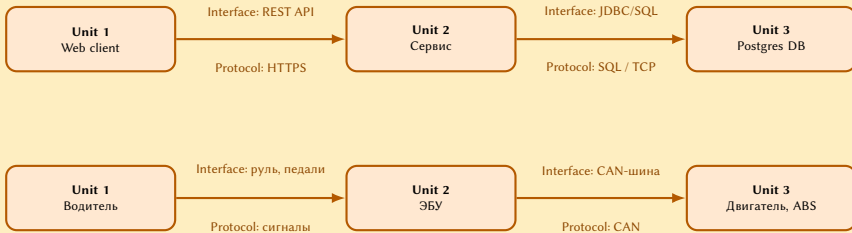
Элемент	Определение	Примеры в ИТ
Unit (Компонент)	Базовый элемент системы	Сервис, модуль, объект
Interface (Интерфейс)	Способ взаимодействия	API, контракт, endpoint
Protocol (Протокол)	Правила взаимодействия	REST, gRPC, сообщения

Связь с принципами SOLID

- **Single Responsibility:** Единая ответственность компонента
- **Interface Segregation:** Разделение интерфейсов
- **Dependency Inversion:** Инверсия зависимостей

Пример: Unit–Interface–Protocol (веб-система и автомобиль)

Три компонента и два канала связи: два примера

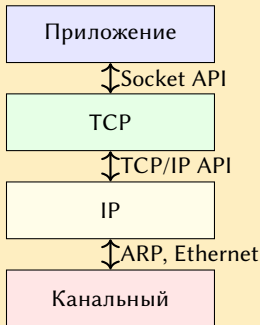


Как читать схему

- **Unit** — компонент, который можно разрабатывать и поставлять отдельно
- **Interface** — контракт: *что* можно вызвать у компонента
- **Protocol** — правила: *как* именно происходит обмен

Пример: Анализ сетевого стека TCP/IP

Применение модели Unit-Interface-Protocol



Практическое применение

- Проектирование микросервисной архитектуры
- Разработка API для внешних интеграций
- Создание плагинных систем

Пример проектирования крупной ERP-системы

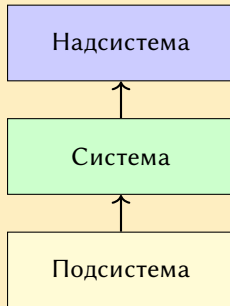
- **Проблема:** Разрозненные бизнес-процессы в компании
- **Системный подход:** Рассмотрение компании как единой системы
- **Решение:** Интеграция финансов, логистики, HR в единую платформу
- **Результат:** Снижение операционных затрат на 30%

Ключевые принципы

- Рассмотрение взаимосвязей между компонентами
- Учет влияния изменений на всю систему
- Баланс между детализацией и целостностью

Концепция Альтшуллера в контексте ИТ-архитектуры

Иерархия систем по Альтшуллеру



Практические примеры для архитекторов

- **Микросервис** (подсистема) → **Приложение** (система) → **Бизнес-экосистема** (надсистема)
- **Класс** (подсистема) → **Модуль** (система) → **Приложение** (надсистема)

Сквозной кейс курса: Короткометражный фильм как система

Постановка задачи

Студенческой команде нужно снять **5-минутный фантастический фильм**: бюджет ограничен, а визуальные эффекты должны быть качественными.

Иерархия по Альтшуллеру

Надсистема: киноиндустрия, платформы, зритель

Система: фильм как целостное произведение

Подсистемы

сцены, диалоги,
актёры, звук, эффекты

Системный взгляд меняет постановку задачи

«Снять фильм» → спроектировать **систему**, которая даёт зрителю нужное переживание при ограниченных ресурсах.

Как делить систему на подсистемы

Средний уровень: логическое проектирование

Критерии декомпозиции

- Основание деления: по бизнес-функциям, по данным или по процессам и командам
- Подсистема — **законченная единица** со своей ответственностью и интерфейсом
- Слабая **связанность** между подсистемами, сильная связность внутри
- **Элементы поставки**: то, что можно разрабатывать, тестировать и выкатывать независимо

Кино-пример

Съёмочный проект делится не «по сценам», а по зонам ответственности: сценарий → продакшн → пост-продакшн → дистрибуция; между ними — интерфейсы (материал, монтажный лист, мастер-копия).

Модели описания устройства и окружения системы

Средний уровень: методы моделирования и описания

Что моделируем

- Структурная схема: подсистемы и связи
- UML: диаграммы компонентов, классов, последовательности, развёртывания
- Интерфейсы и протоколы обмена

Что описываем дополнительно

- Границы системы и её **окружение**: смежные системы
- Входы и выходы, обмен данными с внешним миром
- Сценарии функционирования (кто и как пользуется)

Зачем

Без модели устройства и окружения невозможно понять, **что должна делать система** и какие у неё есть смежные зависимости (мост к анализу предметной области и требованиям).

Архитектурные шаблоны и выбор решения

Продвинутый уровень: обоснованный выбор архитектуры

Типовые решения (шаблоны)

- **Монолит** — просто, быстро для старта
- **Модульный монолит** — монолит с чёткими границами модулей
- **Микросервисы** — независимая поставка и масштабирование, но сложнее в управлении

Выбор по критериям

Скорость вывода функций, масштабируемость, стоимость владения, риски (технический долг), зрелость команды. Не усложняйте раньше времени: начинайте с простого решения и эволюционируйте.

Кино-аналогия

Фильм начинали как «камерный продакшн»; спецэффекты и команды подключают точно — там, где они дают зрителю максимум, а не «езде сразу».

Инструменты моделирования для системных аналитиков

UML-диаграммы для декомпозиции сложных задач

- **Диаграммы компонентов:** Визуализация архитектуры системы
- **Диаграммы развертывания:** Физическое размещение компонентов
- **Диаграммы последовательности:** Взаимодействие между компонентами
- **Диаграммы классов:** Структура данных и отношений

Преимущества системного подхода

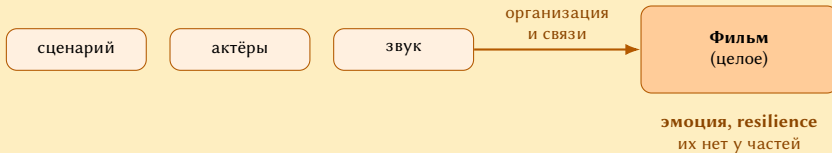
- Четкая декомпозиция сложных задач
- Упрощение анализа и проектирования
- Возможность параллельной разработки компонентов
- Легкость тестирования и отладки

Эмерджентные свойства: что это

Определение

Эмерджентное свойство — свойство системы, которое возникает у **целого** и отсутствует у компонентов по отдельности: оно не сводится к сумме свойств частей, а рождается из их **организации и взаимодействия**.

Кино-пример: сумма частей $\neq$ целое



Критерий проверки

Свойство эмерджентно, если оно **исчезает при «разборке»** системы на части и его нельзя получить суммированием свойств этих частей.

Resilience — пример эмерджентного свойства

Определение

Resilience (устойчивость, живучесть) — способность системы **сохранять свою функцию и восстанавливаться** при сбоях компонентов или внешних возмущениях.

Почему это эмерджентное свойство

- Устойчивость не бывает «у отдельного компонента» — она возникает только на уровне **целого**
- Она порождается **связями**: резервирование, обратная связь, разнообразие, буферы

Примеры

- **ИТ**: три сервера за балансировщиком — отказ одного не останавливает сервис; живучесть сервиса не равна сумме свойств серверов
- **Кино-кейс**: срыв съёмки или уход актёра не убивают проект — команда находит замену, сценарий адаптируется; живучесть — свойство **проекта**, а не отдельной сцены

Пример: масштабируемость

Ситуация

Один сервер держит 100 запросов/с. Добавили ещё 99 серверов и балансировку — система обслуживает 10 000 запросов/с, время отклика не выросло.

Проанализируйте

Это **эмерджентное** свойство или аддитивное? Свойственно ли оно **отдельному** серверу?

Разбор

Масштабируемость — свойство **целого**: у одного сервера его нет, оно рождается из **организации** (балансировка, шардинг, координация). Сто серверов без связей не масштабируются — значит, свойство не сводится к сумме частей.

Пример: синергия

Ситуация

Пять специалистов вместе выпускают продукт, который никто из них не сделал бы в одиночку: результат команды **больше** суммы личных вкладов ($1+1>2$).

Проанализируйте

Откуда берётся «лишнее»? Его можно получить суммированием вкладов каждого участника?

Разбор

Синергия — эмерджентное свойство **взаимодействия**: прибавка рождается из связей и разделения труда, а не из суммы индивидуальных умений.

Пример: сетевой эффект

Ситуация

Мессенджер: чем больше участников, тем ценнее он для каждого. Ценность сети растёт как n^2 (закон Меткалфа), а не как n .

Проанализируйте

Почему ценность сети **не аддитивна**? Кому принадлежит это свойство — узлу или сети?

Разбор

Сетевой эффект — эмерджентное свойство **надсистемы**: у отдельного пользователя его нет, оно возникает из числа и связности участников.

Пример: стайное поведение (муравейник)

Ситуация

Отдельный муравей не строит муравейник и не «управляет» колонией. Но колония как целое находит пищу, распределяет роли и регулирует температуру — без единого «начальника».

Проанализируйте

Есть ли у отдельного муравья план всего муравейника? Откуда берётся порядок?

Разбор

Колония — система, а её «интеллект» и самоорганизация — **эмерджентные** свойства: они возникают из **локальных правил** и взаимодействия (феромонные тропы). Ни у одного муравья нет карты целого.

Отрицательные эмерджентные свойства

Тоже свойства целого — но вредные

- **Пробка** на дороге: её не создал никто лично — она возникает из взаимодействия потоков
- **Каскадный отказ**: сбой одного узла обрушивает всю систему
- **Технический долг**: накапливается из множества мелких решений
- **«Big ball of mud»**: скрытая связанность, в которой уже никто не разберётся
- **Groupthink**: команда сообща принимает решение хуже, чем каждый по отдельности

Вопрос для анализа

Почему каждое из этих свойств **исчезает**, если «разобрать» систему на части?

Наводим порядок: эмерджентно или нет

Эмерджентные (свойства целого)

- **Эмоциональное воздействие** фильма (целостное переживание)
- **Resilience** — устойчивость/живучесть системы
- **Надёжность** резервированной системы (2 сервера надёжнее, чем каждый по отдельности)
- **Способность самолёта летать** — нет ни у двигателя, ни у крыла
- **Адаптивность, самоорганизация**

НЕ эмерджентные (аддитивные)

- **Масса, длина, стоимость/бюджет** — суммы по частям
- **Число сцен, строк кода, серверов** — счётные характеристики
- **Свойства отдельных компонентов** — цвет конкретного кадра, скорость одного сервера

Маркер-подсказка

Если свойство **просто складывается** из частей (аддитивно) — оно не эмерджентно; эмерджентность появляется только на уровне системы.

Системные модели разработки ПО

Водопадная модель

- Линейная детерминированная система
- Четкие этапы: требования → дизайн → реализация → тестирование → поддержка
- Подходит для проектов с фиксированными требованиями

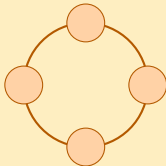
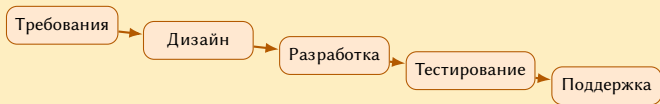
Спиральная модель Бозма

- Итеративная система с управлением рисками
- Циклы: планирование → оценка рисков → разработка → оценка
- Фокус на управлении рисками и прототипировании

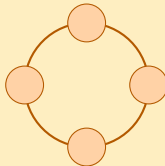
Модели разработки: водопад, спираль, Agile

Как выглядят модели

Водопад — линейно и детерминированно



Спираль Боэма
итерации и риски



Agile
спринты и обратная связь

Agile-методологии как адаптивные системы

Agile-подход

- Адаптивные живые системы
- Итеративная разработка с обратной связью
- Гибкость к изменениям требований

Выбор модели разработки

Тип проекта	Рекомендуемая модель
Государственные системы	Водопадная/спиральная
Стартапы	Agile/Scrum
Корпоративные системы	Гибридные подходы

Кейс: Применение спиральной модели в банковской системе

Проект: Разработка системы онлайн-банкинга

- **Вызов:** Высокие требования к безопасности и надежности
- **Решение:** Спиральная модель с фокусом на управлении рисками
- **Итерация 1:** Прототип авторизации и базовых операций
- **Итерация 2:** Интеграция с платежными системами
- **Итерация 3:** Мобильное приложение и push-уведомления
- **Результат:** Снижение рисков сбоев на 40%

Инварианты и тождественность в бизнес-системах

Бизнес-инварианты vs технические инварианты

Тип	Пример	Назначение
Бизнес-инвариант	«Баланс счета не может быть отрицательным»	Защита бизнес-правил
Технический инвариант	«Сумма всех транзакций должна сохраняться»	Обеспечение целостности данных

Три вида тождественности: схема

Тождественность подсистем

Подсистема **тождественна** другой, если её замена в системе **не меняет систему**. Три вида:

Одно и то же

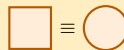


полная идентичность
(клоны, реплики)

Единый Собственно тождественный



есть общая часть
(shared-компонент)



разная форма —
тот же результат

Как это применять

Замену компонента безопасно делать тогда, когда новая подсистема **тождественна** старой по этому критерию.

Три вида тождественности

1. Одно и то же (полная идентичность)

- Клонированный микросервис
- Два идентичных веб-сервера за балансировщиком нагрузки
- Клонированные контейнеры Docker в Kubernetes

2. Единый (имеет общие части)

- Модули, использующие общую библиотеку аутентификации
- Сервисы, работающие с одной базой данных
- Компоненты с общим кэшем

Три вида тождественности (продолжение)

3. Собственно тождественный (функционально эквивалентен)

- Два разных алгоритма сортировки, дающие одинаковый результат
- Разные реализации одного API (REST vs gRPC)
- Миграция с одной СУБД на другую с сохранением функциональности
- Разные провайдеры облачных услуг с одинаковым API

Практическое применение

- Замена компонентов без остановки системы
- Миграция систем между платформами
- A/B-тестирование функциональности
- Резервирование и отказоустойчивость

Примеры из практики

Банковские системы

- **Инварианты:** Сохранение суммы при переводе, неотрицательный баланс
- **Тождественность:** Процессинговые модули разных банков

ERP-системы

- **Инварианты:** Двойная запись в бухгалтерии, баланс склада
- **Тождественность:** Модули учета для разных подразделений

От системного подхода к теории заинтересованных лиц

Система как сеть stakeholders

- Внутренние stakeholders: разработчики, тестировщики, менеджеры
- Внешние stakeholders: клиенты, партнеры, регуляторы
- Косвенные stakeholders: конкуренты, общество, экосистема

Подготовка к следующим лекциям

- **Лекция 2:** Заинтересованные стороны
- **Лекция 3:** Функционально-стоимостный анализ (ФСА)
- **Связь:** От технических компонентов к человеческим факторам

Практическое задание: ваш проект как система

Что сделать: разобрать свой проект как систему

Возьмите школьный (или новый учебный) проект из вводной лекции и посмотрите на него как на систему.

- 1 Определить систему, подсистемы и надсистему
- 2 Спроектировать компоненты (Unit) с интерфейсами (Interface)
- 3 Определить протоколы взаимодействия (Protocol)
- 4 Выявить ключевые инварианты системы
- 5 Проверить тождественность подсистем (три вида)

Подсказка

Если своего проекта нет — возьмите знакомый: учебный сервис, бота, сайт, игру. Пример: у приложения-напоминалки подсистемы «ввод задач / хранение / уведомления», интерфейс — API, протокол — REST/HTTPS.

Дополнительные задания

Задание 1: Анализ существующей ИТ-системы

- Выбрать знакомую ИТ-систему
- Проанализировать через призму Альтшуллера
- Определить подсистемы, систему и надсистему
- Оценить качество декомпозиции

Задание 2: Сравнение моделей разработки

- Проанализировать 2-3 проекта разного типа
- Сравнить применимость разных моделей разработки
- Определить критерии выбора модели
- Предложить оптимальный подход для каждого типа

Дополнительные задания (продолжение)

Задание 3: Инварианты и тождественность

- Выбрать знакомый бизнес-процесс
- Выявить ключевые инварианты
- Проанализировать тождественность компонентов
- Предложить варианты улучшения системы

Примеры для анализа

- Онлайн-магазин: процесс оформления заказа
- Банк: процесс кредитования
- Больница: процесс записи на прием
- Университет: процесс зачисления студентов

Восемь аспектов системного подхода

Чек-лист для разбора системы в задании

- ❶ **Элементный** — из каких элементов состоит система
- ❷ **Структурный** — внутренние связи и строение
- ❸ **Функциональный** — какие функции выполняет
- ❹ **Целевой** — зачем создана, цели и подцели
- ❺ **Ресурсный** — что требуется для работы
- ❻ **Интеграционный** — свойства, обеспечивающие целостность
- ❼ **Коммуникационный** — связи с внешней средой
- ❽ **Исторический** — возникновение, этапы, перспективы

Как пользоваться

Прогоните свой проект по всем восьми аспектам — это и есть разбор системы в практическом задании.

Рекомендуемая литература

- ❶ О'Коннор Дж., Макдермотт И. «Искусство системного мышления»
- ❷ Медоуз Д. «Азбука системного мышления»
- ❸ Бераланфи Л. фон. «Общая теория систем»
- ❹ Садовский В.Н. «Системный подход и общая теория систем»
- ❺ Альтшуллер Г.С. «Творчество как точная наука»
- ❻ Брукс Ф. «Мифический человеко-месяц»
- ❼ Фаулер М. «Паттерны корпоративных приложений»

Инструменты моделирования

- **UML:** Unified Modeling Language - для визуализации архитектуры
- **BPMN:** Business Process Model and Notation - для моделирования процессов
- **ArchiMate:** Для enterprise-архитектуры

Стандарты и фреймворки

Международные стандарты

- **ISO/IEC 15288:** Системная инженерия
- **TOGAF:** Фреймворк enterprise-архитектуры
- **ITIL:** Управление ИТ-услугами
- **COBIT:** Управление и аудит ИТ

Профессиональные компетенции

- Анализ и проектирование систем
- Управление требованиями
- Оценка рисков и качества
- Коммуникация с stakeholders

Лекция 1 → Лекция 2

- Системный подход → Теория заинтересованных лиц
- От анализа системы к анализу ее окружения
- От технических компонентов к человеческим факторам

Лекция 1 → Лекция 3

- Системный подход → Функционально-стоимостный анализ (ФСА)
- От архитектуры системы к её функциям и стоимости
- От инвариантов к функциональной модели и поиску избыточных затрат

От декомпозиции к архитектуре

Лестница уровней: система → логическое проектирование → архитектура

Три ступени

Базовый (junior)

система, подсистемы,
границы, интерфейсы

Средний (middle)

логическое проектирование: деление
на подсистемы,
модели устройства

Продвинутый (senior / архитектор)

шаблоны, выбор
решения, требования
к подсистемам

Связь с вводной лекцией

Три ступени соответствуют движению **junior** → **middle** → **senior/архитектор**, о котором шла речь на вводной лекции.

От архитектуры к требованиям подсистем

Продвинутый уровень: фиксируем требования и согласуем роли

Что даёт архитектура

- Архитектурные принципы организации (стандарты, паттерны, безопасность) превращаются в **требования к подсистеме**
- Функциональные требования — что подсистема делает; нефункциональные — производительность, надёжность, масштабируемость
- Решение описывается на уровне, достаточном для постановки задачи на разработку или доработку подсистемы

Роли

Архитектурные решения **согласуются** системным аналитиком, архитектором и командой разработки; полный процесс фиксации требований — тема лекции 8 (ТЗ по ГОСТ 34.602-89).

Заключение

Ключевые выводы

- Системный подход - фундамент ИТ-профессий
- Декомпозиция сложных систем упрощает анализ и разработку
- Четкие интерфейсы и протоколы обеспечивают гибкость системы
- Инварианты и тождественность - основа надежности

Практическая значимость

- Подготовка к профессиональной деятельности
- Основа для дальнейшего обучения

Возвращаясь к вводной лекции

Системное мышление — это **линза** системного инженера из вводной лекции: структура и ресурсы там, где раньше был хаос.

Вопросы для обсуждения

Темы для дискуссии

- ❶ Какие преимущества дает системный подход в вашей будущей профессии?
- ❷ Как применить концепцию Альтшуллера к проекту, над которым вы работаете?
- ❸ Какие инварианты наиболее важны в банковских системах?
- ❹ Как обеспечить тождественность компонентов при миграции на новую платформу?
- ❺ Какие stakeholders наиболее важны для успеха ИТ-проекта?

Дополнительные ресурсы

- Онлайн-курсы по системному анализу
- Сообщества профессионалов (IIBA, IEEE)
- Конференции и вебинары
- Профессиональная литература и журналы

Ожидаемые результаты обучения

Знания и навыки

- Понимание системного подхода как профессионального инструмента
- Навыки декомпозиции сложных систем
- Умение проектировать системы с четкими интерфейсами и протоколами
- Способность анализировать тождественность компонентов
- Подготовка к работе с теорией заинтересованных лиц

Критерии оценки

- Глубина понимания системных принципов (40%)
- Качество системного разбора проекта (40%)
- Умение применять инструменты моделирования (20%)