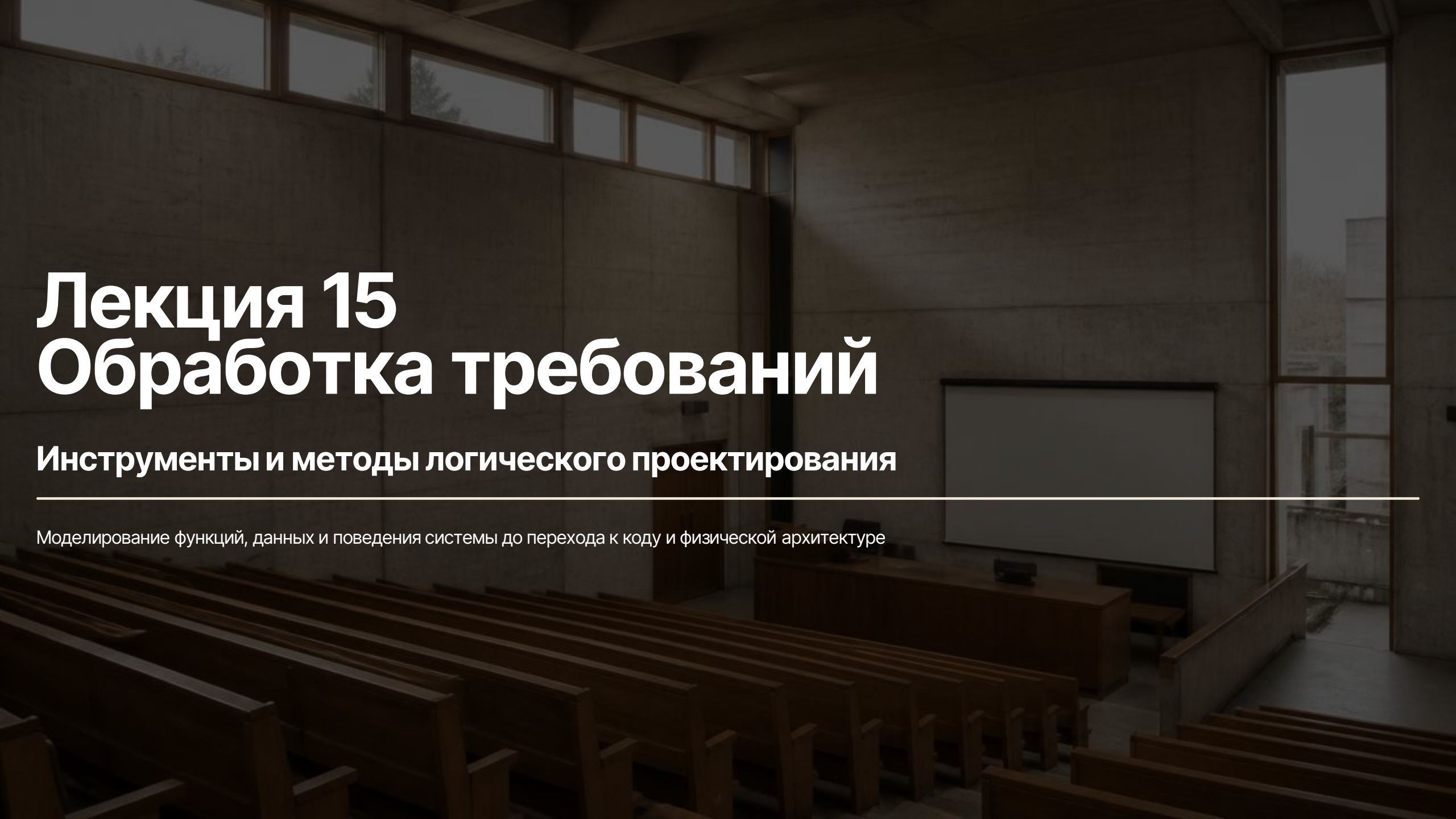
АО «АЛМАТИНСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»
https://atu.edu.kz/

Факультет «Информационные технологии»
Кафедра «Компьютерная инженерия»

Название курса: «Инструментальные средства разработки программ»



Зиятбекова Гулзат Зиятбеккызы – PhD, ассоциированный профессор
g.ziyatbekova@atu.edu.kz



Лекция 15
Обработка требований

Инструменты и методы логического проектирования

Моделирование функций, данных и поведения системы до перехода к коду и физической архитектуре

БАЗОВЫЕ ПОНЯТИЯ

Требования определяют, что и зачем мы строим

Требование — это документированное условие или возможность, необходимая пользователю для решения бизнес-задачи либо регламентированная стандартом.

Требования

Цели стейкхолдеров и бизнес-правила предметной области.

Логика

Абстрактная модель сущностей, функций и потоков данных.

Реализация

СУБД, языки, фреймворки и аппаратная инфраструктура.

Правило Боэма: дефект в требованиях, исправляемый на этапе эксплуатации, обходится в 50–100 раз дороже, чем исправление на этапе логического проектирования.



ЖИЗНЕННЫЙ ЦИКЛ

Обработка требований проходит пять шагов

Системный контур инженерии требований: от первого интервью до контроля версий спецификации

1. Выявление

Интервью, опросники, анализ регламентов и наблюдение за операторами на рабочих местах.

2. Анализ

Классификация, декомпозиция, устранение конфликтов интересов и приоритизация (MoSCoW).

3. Спецификация

Формализация в виде ТЗ, Use Cases, User Stories и схем логического проектирования.

4. Валидация

Экспертные ревью, прототипирование интерфейсов и согласование с заказчиком.

5. Управление

Трассируемость (RTM), версионирование и контроль влияния доработок на архитектуру.

Результат этапа: согласованная спецификация (SRS), пригодная для однозначной трансляции в логические схемы.

ФУНДАМЕНТАЛЬНЫЙ ПРИНЦИП

Логическое проектирование отвечает на вопрос «что», а не «как»

Функциональный срез

ЧТО СИСТЕМА ДЕЛАЕТ

Иерархия функций и процессов обработки без привязки к языку программирования или микросервисам.

Информационный срез

ЧТО СИСТЕМА ПОМНИТ

Сущности, их смысловые атрибуты и связи предметной области независимо от конкретной СУБД (Postgres, Oracle, NoSQL).

Поведенческий срез

КАК РЕАГИРУЕТ

Сценарии взаимодействия актеров, отклики на события и смена состояний бизнес-объектов во времени.

МЕТОДОЛОГИЯ

Как выбрать метод моделирования

Сопоставление четырёх базовых семейств моделей по их назначению, нотации и фокусу внимания

Семейство	Основной вопрос	Ключевые элементы нотации	Когда применять
IDEF0	Каковы функции и правила управления ими?	Функциональные блоки, стрелки ICOM (вход, управление, выход, механизм)	Моделирование сквозных бизнес-процессов компании верхнего уровня
DFD	Куда и как перемещаются потоки данных?	Процессы, накопители данных, внешние сущности, потоки сообщений	Проектирование контуров обработки информации и сервисных интерфейсов
ERD	Какова структура сохраняемых данных?	Сущности, атрибуты, первичные/внешние ключи, кардинальность связей	Проектирование логической реляционной схемы базы данных
UML	Кто взаимодействует с системой и в каких сценариях?	Актёры, прецеденты (Use Cases), дорожки, классы, сообщения	Объектно-ориентированный анализ, ТЗ для разработчиков и QA

Правило архитектора: ни один метод не является универсальным — реальный проект комбинирует срезы функций, данных и сценариев.

ФУНКЦИОНАЛЬНЫЙ СРЕЗ

IDEF0 раскладывает систему на функции

Методология структурного анализа SADT: жесткая семантика стрелок регламентирует окружение каждой операции



Принцип ICOM (Input, Control, Output, Mechanism)

Каждая грань функционального блока имеет фиксированный физический смысл: сверху правила, снизу ресурсы, слева сырье, справа результат.

Иерархическая декомпозиция

От контекстной диаграммы A-0 (система как один черный ящик) до диаграмм уровней A1, A2, A3 (3–6 блоков на каждом листе).

Область применения

Анализ текущего состояния предприятия («As-Is») и целевой регламент работы подразделений («To-Be»).

ПОТОКОВЫЙ СРЕЗ

DFD показывает, как данные текут по системе

Диаграммы потоков данных концентрируются на движении пакетов информации между преобразователями и накопителями

ИСТОЧНИК / СТОК

Внешняя сущность

Человек, сторонний сервис или внешняя система вне границ нашего управления (клиент, платежный шлюз банка).

ДЕЙСТВИЕ

Процесс

Операция трансформации входных потоков данных в выходные. Обязательно нумеруется для декомпозиции (1.1, 1.2).

ХРАНИЛИЩЕ

Накопитель данных

Пассивная память системы (таблица БД, файл, картотека), где данные ожидают последующей выборки.

КАНАЛ

Поток данных

Именованный поток сообщений или структур данных, связывающий процессы, сущности и хранилища.

⚠ **Золотое правило балансировки DFD:** данные не могут появляться из ниоткуда («чудо») и исчезать в никуда («черная дыра»). Накопитель не может связываться напрямую с другим накопителем без процесса-посредника.

ИНФОРМАЦИОННЫЙ СРЕЗ

ER-диаграмма фиксирует данные и связи

Модель сущность-связь (Entity-Relationship) служит математическим мостом к проектированию реляционных баз данных

Сущность (Entity)

Класс однотипных объектов реального мира (Студент, Договор, Товар). Превращается в реляционную таблицу.

Атрибут и Первичный ключ (PK)

Свойства сущности. Ключ уникально идентифицирует каждый экземпляр (ID_студента, Номер_паспорта).

Типы кардинальности связей (Cardinality)

1 : 1 (Один к одному) — Паспорт гражданина ↔ Гражданин

1 : N (Один ко многим) — Кафедра ↔ Преподаватели

M : N (Многие ко многим) — Студенты ↔ Учебные курсы

Реализация связи M:N

В реляционных базах данных связь многие-ко-многим не поддерживается напрямую.

На логическом этапе M:N всегда распадается на две связи 1:N через промежуточную ассоциативную сущность («Студент_Курс»).

ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ ПОДХОД

UML описывает требования через сценарии и взаимодействие

Унифицированный язык моделирования объединяет аналитиков, программистов и тестировщиков через единый визуальный стандарт

USE CASE DIAGRAM

Диаграмма вариантов использования

Определяет границы системы, внешних актеров (пользователей, сервисы) и их бизнес-цели (прецеденты).

Связи: *«include»* (обязательное включение подпроцесса) и *«extend»* (условное расширение сценария).

DYNAMIC DIAGRAMS

Динамические диаграммы логики

Уточняют, как именно реализуются прецеденты во времени и между объектами.

- **Activity:** алгоритмы ветвления и параллельные потоки бизнес-логики.
- **Sequence:** хронология обмена сообщениями между компонентами.

Главное преимущество UML: сквозная трассируемость от бизнес-сценария пользователя до конкретного метода в программном коде класса.

ИТОГИ И ПРАКТИКА

CASE-средства и проверка качества модели

Инструменты автоматизации проектирования (Computer-Aided Software Engineering) поддерживают репозиторий моделей и проверку целостности.

Критерии качества требований:

- **Полнота:** отсутствие белых пятен и неспецифицированных ветвей условий.
- **Непротиворечивость:** модели функций и данных не конфликтуют между собой.
- **Проверяемость:** каждое требование можно покрыть тестом или метрикой.

Золотое правило инженера: выбирайте метод моделирования под конкретный вопрос стейкхолдера, а не подгоняйте систему под любимую нотацию.

