**АО «АЛМАТИНСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»**

Факультет инженерных и информационных технологий

Кафедра компьютерной инженерии

**ПРОЕКТИРОВАНИЕ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА**

## ЛЕКЦИЯ № 1

# Введение в проектирование систем искусственного интеллекта

Составитель: Алтыбай А.

Кафедра компьютерной инженерии

Алматы, 2026

# Аннотация и назначение лекции

Лекция 1 раскрывает тему «Введение в проектирование систем искусственного интеллекта» в контексте разработки промышленной AI-системы. Материал опирается на тематический план дисциплины и соединяет определения, принципы проектирования, математические оценки, проектные артефакты, практический код и аргументированный анализ рисков. Вопросы построены вокруг решений, которые предстоит принимать инженеру, а не только вокруг названий алгоритмов. [1, 2]

## План занятия

1. AI-система как инженерный продукт
2. Исследовательский прототип и production-сервис
3. Замкнутый жизненный цикл
4. Роли в команде
5. Понятие технического долга
6. Поток обучающих данных
7. Поток предсказания
8. Человекоцентричный контроль
9. Практические сценарии, контроль качества и проектные задания.

## Ожидаемые результаты обучения

По завершении студент способен:

1. объяснить и аргументированно применить принцип «AI-система как инженерный продукт» к инженерному решению;
2. объяснить и аргументированно применить принцип «Исследовательский прототип и production-сервис» к инженерному решению;

3. объяснить и аргументированно применить принцип «Замкнутый жизненный цикл» к инженерному решению;
4. объяснить и аргументированно применить принцип «Роли в команде» к инженерному решению;
5. объяснить и аргументированно применить принцип «Понятие технического долга» к инженерному решению;
6. объяснить и аргументированно применить принцип «Поток обучающих данных» к инженерному решению;
7. реализовать упрощённый проверяемый прототип и сформулировать критерии его приемки.
8. указать угрозы достоверности результатов и предложить проверки до эксплуатации.

**Сквозной кейс.** В течение лекции рассматриваем службу поддержки университета. Существенные условия: срочные обращения студентов, ошибки приоритетов. Основная проверяемая характеристика: долю обращений, вовремя получивших ответ. Числа в расчётных примерах учебные и не являются результатами реальных измерений.

# 1. Базовые теоретические положения из конспекта курса

Ниже представлены исходные положения учебно-методического конспекта, связанные с данной темой. Следующие разделы раскрывают их с инженерными деталями и проверяемыми примерами. [1, 2] Цель лекции. сформировать системное представление об AI-системе как о совокупности данных, моделей, программных сервисов, инфраструктуры и эксплуатационных процессов; показать отличие экспериментальной модели от промышленного AI-продукта.

Ключевые понятия. AI-система, ML-система, жизненный цикл, training, inference, model serving, feedback loop, технический долг, роли команды, production-ready.

## 1.1 AI-модель и AI-система

Модель машинного обучения — это функция, параметры которой оцениваются по данным. AI-система шире: она включает источники данных, правила подготовки и проверки данных, модель, сервис предсказаний, интерфейсы, хранилища, инфраструктуру, средства мониторинга, процессы обновления и управления рисками. Поэтому высокая accuracy модели сама по себе не гарантирует полезности и надежности продукта.

В инженерной практике основная сложность часто возникает не в выборе алгоритма, а в обеспечении стабильного потока качественных данных, согласованности training и inference, воспроизводимости экспериментов, контроле задержки и стоимости, а также безопасном обновлении модели.

▹ Экспериментальная модель отвечает на вопрос: «можно ли решить задачу на имеющихся данных?»

▹ Промышленная AI-система отвечает на вопрос: «можно ли решать задачу постоянно, надежно, экономично и безопасно?»

## 1.2 Жизненный цикл AI-системы

Жизненный цикл начинается с постановки проблемы и определения критериев успеха. Затем формируется и проверяется набор данных, строится baseline, выполняются эксперименты и оценка. После выбора кандидата модель упаковывается в сервис, развертывается, наблюдается в эксплуатации и периодически обновляется. В отличие от традиционной программы, поведение ML-компонента зависит не только от кода, но и от данных.

Жизненный цикл является итерационным. Мониторинг может выявить drift или деградацию метрик, что возвращает команду к этапу данных, признаков или обучения. Поэтому проектирование следует выполнять с учетом будущего переобучения уже на ранних этапах.

## 1.3 Роли в AI-проекте

Даже небольшая AI-система объединяет несколько типов компетенций. Data engineer отвечает за надежную доставку и качество данных; data scientist — за гипотезы, признаки, модели и эксперименты; ML engineer — за перенос модели в production и автоматизацию жизненного цикла; software/backend engineer — за интеграцию с продуктом; product owner — за связь с бизнес-ценностью; специалисты по безопасности и governance — за риски и требования.

В небольших командах роли могут совмещаться, но обязанности не исчезают. Полезная практика — заранее назначить владельцев данных, модели, сервиса и эксплуатационных метрик.

## 1.4 Главные инженерные компромиссы

Проектирование AI-системы всегда связано с trade-offs. Более сложная модель может повышать качество, но увеличивать задержку и стоимость inference. Частое переобучение повышает адаптивность, но усложняет контроль качества. Хранение большого числа признаков улучшает повторное использование, но повышает требования к governance.

Важный принцип — оптимизировать не отдельную метрику модели, а полезность всей системы при ограничениях по latency, стоимости, надежности, приватности и сопровождаемости.

## 1.5 Мини-пример: скоринг обращений

Пусть служба поддержки хочет автоматически определять приоритет обращения. Система должна получать текст, извлекать признаки, вызвать модель, вернуть вероятность критичности и сохранить решение для аудита. Дополнительно необходимо мониторить распределение тем обращений, частоту ошибок, latency и долю ручных исправлений оператором.

Таким образом, модель классификации — лишь один узел. Без API, журналирования, схемы данных и механизма обратной связи система не завер-

шена.

## 1.6 Выводы

▷ AI-система состоит из данных, модели, инфраструктуры, интерфейсов и эксплуатационных процессов.

▷ Жизненный цикл AI-системы итерационен и включает постоянную обратную связь.

▷ Качество решения оценивается одновременно по ML-метрикам и системным требованиям.

▷ Проектирование должно учитывать обновление, мониторинг и риски с самого начала.

## 1.7 Контрольные вопросы

▷ Чем AI-система отличается от отдельной ML-модели?

▷ Почему production-ready система требует мониторинга после развертывания?

▷ Какие роли необходимы в типичном AI-проекте?

▷ Приведите пример компромисса между качеством модели и эксплуатационной стоимостью.

▷ Почему жизненный цикл AI-системы является итерационным?

# 2. Систематическое изложение темы и инженерные решения

## 2.1 AI-система как инженерный продукт

В отличие от функции предсказания, продукт включает ввод данных, обработку ошибок, интерфейс, регистрацию решений и процедуру обновления.

Пользователь судит о системе по качеству сервиса, а не по значению метрики на тестовом наборе. [2]

**Проверяемый пример.** Для сортировки обращений на схеме должны присутствовать входной API, предобработка текста, модель, журнал решений и оператор, который может изменить приоритет.

**Вопрос для обсуждения.** Как проверить выполнение такого требования: Для сортировки обращений на схеме должны присутствовать входной API, предобработка текста, модель, журнал решений и оператор, который может изменить приоритет.

## 2.2 Исследовательский прототип и production-сервис

Прототип оценивает принципиальную применимость алгоритма при ограниченном наборе данных. Production-сервис несет ответственность за доступность, безопасность, воспроизводимость и стоимость каждого решения. [3]

**Проверяемый пример.** Сравните notebook с ручной загрузкой CSV и сервис с автоматическим источником событий, мониторингом и версионированием артефактов.

**Вопрос для обсуждения.** Как проверить выполнение такого требования: Сравните notebook с ручной загрузкой CSV и сервис с автоматическим источником событий, мониторингом и версионированием артефактов.

## 2.3 Замкнутый жизненный цикл

Сбор требований, исследование данных, обучение, оценка, внедрение и эксплуатация образуют итеративную петлю. Новые данные и изменение поведения пользователей могут сделать старую модель непригодной. [4]

**Проверяемый пример.** Постройте схему возврата от обнаруженного drift к экспертизе данных и переобучению с обязательным утвержде-

нием релиза.

**Вопрос для обсуждения.** Как проверить выполнение такого требования: Постройте схему возврата от обнаруженного drift к экспертизе данных и переобучению с обязательным утверждением релиза.

## 2.4 Роли в команде

Product owner формулирует ценность; data engineer строит потоки; data scientist проверяет гипотезы; ML engineer обеспечивает инференс; SRE и безопасность поддерживают эксплуатацию. В малой команде совмещаются должности, но не ответственность. [5]

**Проверяемый пример.** Создайте матрицу RACI для этапов «разметка», «утверждение метрики», «выкатка», «откат».

**Вопрос для обсуждения.** Как проверить выполнение такого требования: Создайте матрицу RACI для этапов «разметка», «утверждение метрики», «выкатка», «откат».

## 2.5 Понятие технического долга

Зависимость от нестабильных признаков, копирование логики предобработки и неявные потребители предсказаний образуют скрытый технический долг ML-систем. Ошибки часто проявляются лишь спустя месяцы. [6]

**Проверяемый пример.** Найдите, что сломается, если входной признак «тип обращения» переименуют в поставляющей системе.

**Вопрос для обсуждения.** Как проверить выполнение такого требования: Найдите, что сломается, если входной признак «тип обращения» переименуют в поставляющей системе.

## 2.6 Поток обучающих данных

Обучающий pipeline преобразует сырые записи в проверенный набор с метаданными происхождения, разбиением и фиксированной схемой. Должна быть известна версия обучающих данных. [7]

**Проверяемый пример.** Для каждой записи сохраняйте идентификатор, время события, источник метки и правила исключения дубликатов.

**Вопрос для обсуждения.** Как проверить выполнение такого требования: Для каждой записи сохраняйте идентификатор, время события, источник метки и правила исключения дубликатов.

## 2.7 Поток предсказания

Онлайн-инференс содержит десериализацию, валидацию, преобразование, вызов модели и сериализацию ответа. Время ответа складывается из этих операций и сети. [1]

**Проверяемый пример.** Разделите задержку 120 мс на сеть, преобразование и собственно вызов модели; предложите бюджет для каждой части.

**Вопрос для обсуждения.** Как проверить выполнение такого требования: Разделите задержку 120 мс на сеть, преобразование и собственно вызов модели; предложите бюджет для каждой части.

## 2.8 Человекоцентричный контроль

Высокорисковые или неуверенные предсказания передаются человеку. Канал исправлений необходим и для сервиса, и для получения качественной обратной связи. [2]

**Проверяемый пример.** Определите правило эскалации: низкая уверенность, недопустимое содержимое или ошибка входных данных.

**Вопрос для обсуждения.** Как проверить выполнение такого требования: Определите правило эскалации: низкая уверенность, недопустимое содержимое или ошибка входных данных.

## 2.9 Критерии завершённости

Definition of Done для AI-системы фиксирует не только accuracy, но и документированные ограничения, тесты, ресурсы, владельца и план инцидентов. [3]

**Проверяемый пример.** Подготовьте акт приемки, в котором отдельно представлены качество модели и доступность API.

**Вопрос для обсуждения.** Как проверить выполнение такого требования: Подготовьте акт приемки, в котором отдельно представлены качество модели и доступность API.

## 2.10 Компромисс качество–стоимость

Точность может расти одновременно с требованиями к GPU, энергопотреблению и времени ответа. Выбирают допустимую Парето-эффективную конфигурацию, а не автоматически самую сложную. [4]

**Проверяемый пример.** Сравните две модели с F1=0.88 и F1=0.90 при p95 40 и 350 мс соответственно.

**Вопрос для обсуждения.** Как проверить выполнение такого требования: Сравните две модели с F1=0.88 и F1=0.90 при p95 40 и 350 мс соответственно.

## 2.11 Проверяемость предположений

Каждое обещание системы должно иметь способ испытания: метрика, тестовый набор, протокол измерения, владелец и допустимый порог. [5]

**Проверяемый пример.** Фразу «система работает быстро» замените требованием «p95 инференса < 200 мс при 50 запросах/с».

**Вопрос для обсуждения.** Как проверить выполнение такого требования: Фразу «система работает быстро» замените требованием «p95 инференса < 200 мс при 50 запросах/с».

## 2.12 Документация и трассируемость

Связь бизнес-цели, датасета, эксперимента, версии модели и релиза позволяет объяснить происхождение любого результата. Недокументированная модель трудно воспроизводима. [6]

**Проверяемый пример.** Отследите ошибочный ответ от request_id через версию модели до commit и снимка данных.

**Вопрос для обсуждения.** Как проверить выполнение такого требования: Отследите ошибочный ответ от request_id через версию модели до commit и снимка данных.

# 3. Математические инструменты и количественные критерии

Для перехода от неформального решения к проверяемой постановке введём следующие величины.

$$\mathcal{S} = \big(\mathcal{D}, \mathcal{P}, f_\theta, \mathcal{A}, \mathcal{O}\big) \tag{1}$$

$$\mathrm{Cost}_{\mathrm{system}} = \mathrm{Cost}_{\mathrm{data}} + \mathrm{Cost}_{\mathrm{training}} + \mathrm{Cost}_{\mathrm{serving}} + \mathrm{Cost}_{\mathrm{operations}} \tag{2}$$

Здесь данные, pipeline, обученная функция, приложение и эксплуатационные процессы образуют систему. Оптимизация только $f_\theta$ не гарантирует минимум системной стоимости.

При выборе числового критерия нужно отличать обучающую функцию потерь, продуктовую ценность и эксплуатационные ограничения. Первую можно оптимизировать дифференцированием, вторая определяется процессом принятия решения, а третьи проверяются нагрузочными испытаниями, политиками безопасности и мониторингом.

## 3.1 Пошаговый расчёт на учебных данных

Рассмотрим две конфигурации одной системы. Вариант A имеет условную оценку качества $q_A = 0.86$ и задержку $t_A = 90$ мс; вариант B — $q_B = 0.89$ и $t_B = 280$ мс. Предположим, что требование заказчика задаёт $q \geq 0.85$ и $p95 \leq 150$ мс. Тогда A удовлетворяет обоим условиям, тогда как B нарушает второе. Следовательно, выбор B только потому, что его offline-метрика выше, не является обоснованным. Числа иллюстративны; реальный вывод требует измерений при одинаковой нагрузке и одинаковом наборе данных.

$$\mathrm{accept}(A) = \mathbf{1}\{0.86 \geq 0.85 \wedge 90 \leq 150\} = 1,$$
$$\mathrm{accept}(B) = \mathbf{1}\{0.89 \geq 0.85 \wedge 280 \leq 150\} = 0.$$

## 3.2 Как интерпретировать число, а не только вычислить его

Сначала следует проверить единицы измерения, состав выборки и смысл агрегирования. Например, p95 на спокойном стенде нельзя переносить на рабочую нагрузку с иным числом одновременных запросов. Затем исследуют чувствительность к изменениям входных данных и неопределённость оценок. Наконец, фиксируют последствия порога: какую ошибку он предотвращает, а какую может увеличить. Этот подход делает количественный критерий частью инженерной аргументации, а не украшением отчёта. [1, 2]

# 4. Практическое программирование на Python

Пример предназначен для демонстрации ключевого инженерного принципа. Он не претендует на обработку реального production-трафика. Запускать код следует в изолированном окружении и фиксировать версии библиотек, если они используются. Вывод зависит от указанных входных данных; для стохастических алгоритмов он может меняться.

## 4.1 Основной воспроизводимый пример

```python
from dataclasses import dataclass
from typing import Callable
@dataclass
class Prediction:
    label: str
    confidence: float
    model_version: str

def predict_message(text: str, model: Callable) -> Prediction:
    if not text.strip():
        raise ValueError("empty input")
    label, confidence = model(text)
    return Prediction(label, float(confidence), "baseline-v1")

def baseline(text):
    return ("urgent" if "urgent" in text.lower() else "normal", 0.6)

print(predict_message("urgent: server down", baseline))
```

**Проверка результатов.** Сначала убедитесь, что код завершился без исключений; проверьте формы массивов, схему результата и отсутствие недопустимых значений. Если используются внешние данные, включите в отчёт способ их получения и идентификатор версии. Измените один входной параметр и объясните, почему наблюдаемый результат изменился или остался прежним.

## 4.2 Компактная проверка контракта или метрики

```python
def validate_record(obj):
    assert isinstance(obj.get("request_id"), str)
    assert obj.get("latency_ms", -1) >= 0
    assert obj.get("model_version")

sample = {"request_id": "r-1", "latency_ms": 23, "model_version": "v1"}
validate_record(sample)
print("contract validated")
```

При самостоятельном повторении предложите минимум один негативный тест: некорректное значение, пустой вход, несовместимую версию схемы либо нарушение SLO. Переход от успешного демонстрационного запуска к надежной системе требует именно таких проверок.

# 5. Разбор трёх инженерных ситуаций

## 5.1 Ситуация 1: AI-система как инженерный продукт

Заказчик использует службу поддержки университета. Новое требование заставляет рассмотреть принцип «AI-система как инженерный продукт». Основная неопределённость состоит в том, как одновременно обеспечить долю обращений, вовремя получивших ответ, не допустить ошибки приоритетов и сохранить возможность изменения модели.

**Постановка.** В отличие от функции предсказания, продукт включает ввод данных, обработку ошибок, интерфейс, регистрацию решений и процедуру обновления. Пользователь судит о системе по качеству сервиса, а не по значению метрики на тестовом наборе.

**Действие инженера.** Для сортировки обращений на схеме должны присутствовать входной API, предобработка текста, модель, журнал решений и оператор, который может изменить приоритет.

**Альтернативы.** Сравните три подхода: (A) простое правило без ML, (B) модульное ML-решение с отдельной проверкой входных данных и (C) интегрированный сервис со сложной моделью. Для каждого укажите потребность в данных, ожидаемые инженерные риски, возможность тестирования и минимальную инфраструктуру. Таблица ниже задаёт форму отчёта, а не готовый рейтинг.

| Вариант | Данные и зависимости | Проверка | Основной риск |
|---|---|---|---|
| Правила | Фиксированные условия | Набор тест-кейсов | Ограниченная гибкость |
| ML-модуль | Версионированные признаки | Offline и API tests | Drift и skew |
| Сложный сервис | Несколько компонентов | End-to-end и нагрузка | Каскадный отказ |

**Проверка решения.** Подготовьте две позитивные и две негативные пробы; задайте метрику успеха, базовый сценарий и критерий остановки внедрения. Обсудите, какие допущения останутся непроверенными даже по-

сле удачного теста.

## 5.2 Ситуация 2: Поток обучающих данных

Заказчик использует службу поддержки университета. Новое требование заставляет рассмотреть принцип «Поток обучающих данных». Основная неопределённость состоит в том, как одновременно обеспечить долю обращений, вовремя получивших ответ, не допустить ошибки приоритетов и сохранить возможность изменения модели.

**Постановка.** Обучающий pipeline преобразует сырые записи в проверенный набор с метаданными происхождения, разбиением и фиксированной схемой. Должна быть известна версия обучающих данных.

**Действие инженера.** Для каждой записи сохраняйте идентификатор, время события, источник метки и правила исключения дубликатов.

**Альтернативы.** Сравните три подхода: (A) простое правило без ML, (B) модульное ML-решение с отдельной проверкой входных данных и (C) интегрированный сервис со сложной моделью. Для каждого укажите потребность в данных, ожидаемые инженерные риски, возможность тестирования и минимальную инфраструктуру. Таблица ниже задаёт форму отчёта, а не готовый рейтинг.

| Вариант | Данные и зависимости | Проверка | Основной риск |
|---|---|---|---|
| Правила | Фиксированные условия | Набор тест-кейсов | Ограниченная гибкость |
| ML-модуль | Версионированные признаки | Offline и API tests | Drift и skew |
| Сложный сервис | Несколько компонентов | End-to-end и нагрузка | Каскадный отказ |

**Проверка решения.** Подготовьте две позитивные и две негативные пробы; задайте метрику успеха, базовый сценарий и критерий остановки внедрения. Обсудите, какие допущения останутся непроверенными даже после удачного теста.

## 5.3 Ситуация 3: Проверяемость предположений

Заказчик использует службу поддержки университета. Новое требование заставляет рассмотреть принцип «Проверяемость предположений». Основная неопределённость состоит в том, как одновременно обеспечить долю обращений, вовремя получивших ответ, не допустить ошибки приоритетов и сохранить возможность изменения модели.

**Постановка.** Каждое обещание системы должно иметь способ испытания: метрика, тестовый набор, протокол измерения, владелец и допустимый порог.

**Действие инженера.** Фразу «система работает быстро» замените требованием «p95 инференса < 200 мс при 50 запросах/с».

**Альтернативы.** Сравните три подхода: (A) простое правило без ML, (B) модульное ML-решение с отдельной проверкой входных данных и (C) интегрированный сервис со сложной моделью. Для каждого укажите потребность в данных, ожидаемые инженерные риски, возможность тестирования и минимальную инфраструктуру. Таблица ниже задаёт форму отчёта, а не готовый рейтинг.

| Вариант | Данные и зависимости | Проверка | Основной риск |
|---|---|---|---|
| Правила | Фиксированные условия | Набор тест-кейсов | Ограниченная гибкость |
| ML-модуль | Версионированные признаки | Offline и API tests | Drift и skew |
| Сложный сервис | Несколько компонентов | End-to-end и нагрузка | Каскадный отказ |

**Проверка решения.** Подготовьте две позитивные и две негативные пробы; задайте метрику успеха, базовый сценарий и критерий остановки внедрения. Обсудите, какие допущения останутся непроверенными даже после удачного теста.

# 6. Самостоятельная работа: индивидуальные проектные задания

Для каждого задания студент создаёт отдельный небольшой артефакт: схему, набор тестов, код либо таблицу сравнения, прикладывает наблюдаемый результат и объясняет ограничения.

1. **AI-система как инженерный продукт.** Для сортировки обращений на схеме должны присутствовать входной API, предобработка текста, модель, журнал решений и оператор, который может изменить приоритет. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

2. **Исследовательский прототип и production-сервис.** Сравните notebook с ручной загрузкой CSV и сервис с автоматическим источником событий, мониторингом и версионированием артефактов. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

3. **Замкнутый жизненный цикл.** Постройте схему возврата от обнаруженного drift к экспертизе данных и переобучению с обязательным утверждением релиза. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

4. **Роли в команде.** Создайте матрицу RACI для этапов «разметка», «утверждение метрики», «выкатка», «откат». Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

5. **Понятие технического долга.** Найдите, что сломается, если входной признак «тип обращения» переименуют в поставляющей системе. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

6. **Поток обучающих данных.** Для каждой записи сохраняйте идентификатор, время события, источник метки и правила исключения дубликатов. Сформулируйте проверяемое требование, продемонстрируйте

альтернативу и приложите протокол хотя бы одного эксперимента.

7. **Поток предсказания.** Разделите задержку 120 мс на сеть, преобразование и собственно вызов модели; предложите бюджет для каждой части. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

8. **Человекоцентричный контроль.** Определите правило эскалации: низкая уверенность, недопустимое содержимое или ошибка входных данных. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

9. **Критерии завершённости.** Подготовьте акт приемки, в котором отдельно представлены качество модели и доступность API. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

10. **Компромисс качество–стоимость.** Сравните две модели с F1=0.88 и F1=0.90 при p95 40 и 350 мс соответственно. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

11. **Проверяемость предположений.** Фразу «система работает быстро» замените требованием «p95 инференса < 200 мс при 50 запросах/с». Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

12. **Документация и трассируемость.** Отследите ошибочный ответ от request_id через версию модели до commit и снимка данных. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

## 7. Вопросы для текущего и итогового контроля

1. В чём назначение и ограничения принципа «AI-система как инженерный продукт»?

2. Как проверить на практике утверждение: «Для сортировки обращений

на схеме должны присутствовать входной API, предобработка текста, модель, журнал решений и оператор, который может изменить приоритет.»?

3. В чём назначение и ограничения принципа «Исследовательский прототип и production-сервис»?
4. Как проверить на практике утверждение: «Сравните notebook с ручной загрузкой CSV и сервис с автоматическим источником событий, мониторингом и версионированием артефактов.»?
5. В чём назначение и ограничения принципа «Замкнутый жизненный цикл»?
6. Как проверить на практике утверждение: «Постройте схему возврата от обнаруженного drift к экспертизе данных и переобучению с обязательным утверждением релиза.»?
7. В чём назначение и ограничения принципа «Роли в команде»?
8. Как проверить на практике утверждение: «Создайте матрицу RACI для этапов «разметка», «утверждение метрики», «выкатка», «откат».»?
9. В чём назначение и ограничения принципа «Понятие технического долга»?
10. Как проверить на практике утверждение: «Найдите, что сломается, если входной признак «тип обращения» переименуют в поставляющей системе.»?
11. В чём назначение и ограничения принципа «Поток обучающих данных»?
12. Как проверить на практике утверждение: «Для каждой записи сохраняйте идентификатор, время события, источник метки и правила исключения дубликатов.»?
13. В чём назначение и ограничения принципа «Поток предсказания»?
14. Как проверить на практике утверждение: «Разделите задержку 120 мс на сеть, преобразование и собственно вызов модели; предложите бюджет для каждой части.»?

15. В чём назначение и ограничения принципа «Человекоцентричный контроль»?
16. Как проверить на практике утверждение: «Определите правило эскалации: низкая уверенность, недопустимое содержимое или ошибка входных данных.»?
17. В чём назначение и ограничения принципа «Критерии завершённости»?
18. Как проверить на практике утверждение: «Подготовьте акт приемки, в котором отдельно представлены качество модели и доступность API.»?
19. В чём назначение и ограничения принципа «Компромисс качество–стоимость»?
20. Как проверить на практике утверждение: «Сравните две модели с F1=0.88 и F1=0.90 при p95 40 и 350 мс соответственно.»?
21. В чём назначение и ограничения принципа «Проверяемость предположений»?
22. Как проверить на практике утверждение: «Фразу «система работает быстро» замените требованием «p95 инференса < 200 мс при 50 запросах/с».»?
23. В чём назначение и ограничения принципа «Документация и трассируемость»?
24. Как проверить на практике утверждение: «Отследите ошибочный ответ от request_id через версию модели до commit и снимка данных.»?
25. Почему высокое offline-качество не гарантирует надежность промышленной AI-системы?
26. Какие артефакты нужны для воспроизводимости и анализа инцидента?

# 8. Глоссарий ключевых понятий

**AI-система как инженерный продукт**
В отличие от функции предсказания, продукт включает ввод данных, обработку ошибок, интерфейс, регистрацию решений и процедуру обновления.

**Исследовательский прототип и production-сервис**
Прототип оценивает принципиальную применимость алгоритма при ограниченном наборе данных.

**Замкнутый жизненный цикл**
Сбор требований, исследование данных, обучение, оценка, внедрение и эксплуатация образуют итеративную петлю.

**Роли в команде**
Product owner формулирует ценность; data engineer строит потоки; data scientist проверяет гипотезы; ML engineer обеспечивает инференс; SRE и безопасность поддерживают эксплуатацию.

**Понятие технического долга**
Зависимость от нестабильных признаков, копирование логики предобработки и неявные потребители предсказаний образуют скрытый технический долг ML-систем.

**Поток обучающих данных**
Обучающий pipeline преобразует сырые записи в проверенный набор с метаданными происхождения, разбиением и фиксированной схемой.

**Поток предсказания**
Онлайн-инференс содержит десериализацию, валидацию, преобразование, вызов модели и сериализацию ответа.

**Человекоцентричный контроль**
Высокорисковые или неуверенные предсказания передаются человеку.

**Критерии завершённости**
Definition of Done для AI-системы фиксирует не только accuracy, но и документированные ограничения, тесты, ресурсы, владельца и план инцидентов.

**Компромисс качество–стоимость**

Точность может расти одновременно с требованиями к GPU, энергопотреблению и времени ответа.

**Проверяемость предположений**

Каждое обещание системы должно иметь способ испытания: метрика, тестовый набор, протокол измерения, владелец и допустимый порог.

**Документация и трассируемость**

Связь бизнес-цели, датасета, эксперимента, версии модели и релиза позволяет объяснить происхождение любого результата.

## 9. Итоги и связь со следующей темой

Представленные методы имеют смысл как части целостного управляемого жизненного цикла. Ключевой результат инженерного проектирования состоит не в максимальном числе компонентов или сложности модели, а в убедительной связи требований, проектных решений, измерений и механизмов сопровождения. Дальнейшее изучение курса развивает эти связи на следующем этапе жизненного цикла.

# Список литературы

[1] Huyen C. Designing Machine Learning Systems. O'Reilly Media, 2022. https://www.oreilly.com/library/view/designing-machine-learning/9781098107956/.

[2] Sculley D. et al. Hidden Technical Debt in Machine Learning Systems. NeurIPS, 2015. https://research.google/pubs/hidden-technical-debt-in-machine-learning-systems/.

[3] Google Developers. Rules of Machine Learning: Best Practices for ML Engineering. https://developers.google.com/machine-learning/guides/rules-of-ml.

[4] Breck E. et al. The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction. 2017. https://research.google/pubs/the-ml-test-score-a-rubric-for-ml-production-readiness-and-technical-debt-reduction/.

[5] Kleppmann M. Designing Data-Intensive Applications. O'Reilly Media, 2017. https://dataintensive.net/.

[6] NIST. AI Risk Management Framework (AI RMF 1.0). NIST AI 100-1, 2023. https://doi.org/10.6028/NIST.AI.100-1.

[7] Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 3rd ed., O'Reilly Media, 2022. https://www.oreilly.com/library/view/hands-on-machine-learning/9781098125967/.