АО «АЛМАТИНСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет инженерных и информационных технологий

Кафедра компьютерной инженерии

ПРОЕКТИРОВАНИЕ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

ЛЕКЦИЯ № 15

Комплексное проектирование промышленной AI-системы

Теоретико-практический конспект

Составитель: Алтыбай А.

Кафедра компьютерной инженерии

Алматы, 2026

Аннотация и назначение лекции

Лекция 15 раскрывает тему «Комплексное проектирование промышленной AI-системы» 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, 3]

План занятия

1. Итоговая постановка
2. Контекстная архитектура
3. Распределение требований
4. Данные и версионирование
5. Экспериментальная методика
6. Структура репозитория
7. Процесс релиза
8. Наблюдаемость
9. Практические сценарии, контроль качества и проектные задания.

Ожидаемые результаты обучения

По завершении студент способен:

1. объяснить и аргументированно применить принцип «Итоговая постановка» к инженерному решению;
2. объяснить и аргументированно применить принцип «Контекстная архитектура» к инженерному решению;

3. объяснить и аргументированно применить принцип «Распределение требований» к инженерному решению;
4. объяснить и аргументированно применить принцип «Данные и версионирование» к инженерному решению;
5. объяснить и аргументированно применить принцип «Экспериментальная методика» к инженерному решению;
6. объяснить и аргументированно применить принцип «Структура репозитория» 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 университетский AI-сервис. Существенные условия: несогласованность компонентов, необоснованную стоимость. Основная проверяемая характеристика: выполнение требований приемки. 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, 3] Цель лекции. объединить знания курса в методику проектирования полной промышленной AI-системы и показать порядок архитектурного решения, документирования, оценки рисков и развития системы.

Ключевые понятия. system design, trade-off, architecture decision record, risk register, cost model, capacity planning, rollout plan, ownership, decommissioning.

1.1 Шаг 1. Контекст и критерии успеха

Сначала фиксируются пользователи, решение, бизнес-процесс, границы ответственности и измеримые критерии успеха. Архитектура без четких требований неизбежно оптимизирует случайные свойства. Следует явно указать ограничения: сроки, бюджет, инфраструктура, регуляторные условия и доступность данных.

Хорошее design document начинается с problem statement и non-goals — того, что система сознательно не будет решать.

1.2 Шаг 2. Данные и модель

Определяются источники, ownership, качество, схема, labeling, split strategy, baseline и метод оценки. Затем сравниваются модели с учетом не только качества, но и latency, памяти, стоимости, интерпретируемости и лицензионных ограничений.

Для LLM-проекта отдельно проектируются retrieval, prompt/versioning, guardrails и evaluation set.

1.3 Шаг 3. Serving и инфраструктура

Выбирается batch/online/streaming режим, API contract, runtime, контейнеризация, стратегия масштабирования и кэширования. Capacity planning оценивает ожидаемый трафик и ресурсы. Нужно определить SLO и behavior при перегрузке или отказе внешних зависимостей.

Ключевой принцип: failure mode должен быть предсказуемым. Система не должна превращать временную недоступность модели в отказ всего продукта, если возможен безопасный fallback.

1.4 Шаг 4. MLOps, мониторинг и управление рисками

До запуска задаются правила experiment tracking, registry, approval, deployment, rollback и retraining. Monitoring plan связывает каждый риск с

наблюдаемым сигналом и действием. Risk register описывает вероятность, влияние, владельца и mitigation.

Для production должно быть ясно, кто получает alert, кто принимает решение об остановке модели и где находится предыдущая стабильная версия.

1.5 Шаг 5. Документирование и эволюция

Architecture Decision Record фиксирует важное решение и альтернативы. Model card описывает назначение и ограничения модели. Runbook объясняет оператору действия при инциденте. Change log позволяет понимать эволюцию поведения.

AI-система должна иметь план окончания жизненного цикла: когда модель признается устаревшей, как удаляются данные, кто отключает сервис и как проверяется отсутствие зависимых клиентов.

1.6 Итоговый чек-лист

Перед запуском полезно провести design review и production readiness review.

- ▷ Есть измеримые business/ML/system критерии успеха.
- ▷ Данные имеют owner, version, validation и lineage.
- ▷ Training и inference используют совместимый preprocessing.
- ▷ Модель сравнивалась с baseline и прошла segment/error analysis.
- ▷ API и deployment имеют SLO, timeout, fallback и rollback.
- ▷ Эксперименты, модели и конфигурации трассируются.
- ▷ Есть monitoring, alerting и incident response plan.
- ▷ Проверены privacy, fairness, security и human oversight.
- ▷ Документация достаточна для передачи системы другой команде.

1.7 Выводы

▷ Системное проектирование начинается с требований и ограничений, а не с выбора модели.

▷ Данные, модель, serving, MLOps и governance должны проектироваться совместно.

▷ Production readiness — это проверяемый набор условий, а не субъективное впечатление.

▷ Жизненный цикл включает эксплуатацию, обновление и завершение системы.

1.8 Контрольные вопросы

▷ Какие разделы должны входить в design document AI-системы?

▷ Что такое capacity planning?

▷ Какие условия должны быть выполнены перед production rollout?

▷ Зачем нужен risk register?

▷ Почему decommissioning следует учитывать при проектировании?

2. Систематическое изложение темы и инженерные решения

2.1 Итоговая постановка

Комплексный проект соединяет бизнес-решение, данные, модель, сервис и процесс эксплуатации. Его спецификация должна раскрывать, почему AI необходим. [2]

Проверяемый пример. Опишите минимальную версию сервиса автоматической классификации обращений с операторской проверкой.

Вопрос для обсуждения. Как проверить выполнение такого требования: Опишите минимальную версию сервиса автоматической классификации обращений с операторской проверкой.

2.2 Контекстная архитектура

На верхнем уровне фиксируют взаимодействие пользователя, внутренней платформы, внешних поставщиков и служб безопасности. Потоки данных и границы доверия отмечают отдельно. [3]

Проверяемый пример. Создайте контекстную и контейнерную схемы на основе C4.

Вопрос для обсуждения. Как проверить выполнение такого требования: Создайте контекстную и контейнерную схемы на основе C4.

2.3 Распределение требований

Каждое требование должно иметь архитектурное место и тест: бизнес-критерии проверяются на уровне процесса, latency на уровне API, privacy в потоке данных. [4]

Проверяемый пример. Постройте traceability matrix из пяти требований и пяти проверок.

Вопрос для обсуждения. Как проверить выполнение такого требования: Постройте traceability matrix из пяти требований и пяти проверок.

2.4 Данные и версионирование

Сбор, валидаторы, разметка, lineage и train–serve consistency образуют цепочку. Она должна выдерживать обновление схемы источника. [5]

Проверяемый пример. Спланируйте переход поля priority из строк в перечисление без поломки старой модели.

Вопрос для обсуждения. Как проверить выполнение такого требования: Спланируйте переход поля priority из строк в перечисление без поломки старой модели.

2.5 Экспериментальная методика

Baseline и кандидат оцениваются на фиксированном test, валидация подбирает гиперпараметры, а отчет включает ошибки и неопределенность. [6]

Проверяемый пример. Сформируйте таблицу F1, p95 latency, memory footprint и cost по двум кандидатам.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сформируйте таблицу F1, p95 latency, memory footprint и cost по двум кандидатам.

2.6 Структура репозитория

Код отделяют от конфигураций, тестов, данных и артефактов. README должен позволять воспроизвести ключевой эксперимент. [7]

Проверяемый пример. Разработайте дерево src/, tests/, configs/, pipelines/, docs/.

Вопрос для обсуждения. Как проверить выполнение такого требования: Разработайте дерево src/, tests/, configs/, pipelines/, docs/.

2.7 Процесс релиза

После unit и data tests модель проходит evaluation gate и staging. Production rollout контролируется guardrails и сопровождается планом отката. [8]

Проверяемый пример. Опишите канареечный rollout 10% пользователей с условиями отмены.

Вопрос для обсуждения. Как проверить выполнение такого требования: Опишите канареечный rollout 10% пользователей с условиями отмены.

2.8 Наблюдаемость

Эксплуатационный обзор содержит сервисные и модельные метрики, сигналы дрейфа и жалобы пользователей. Владельцы реагируют на определенные события. [1]

Проверяемый пример. Создайте проект dashboard и примеры двух alerts.

Вопрос для обсуждения. Как проверить выполнение такого требования: Создайте проект dashboard и примеры двух alerts.

2.9 Управление риском

Составьте risk register для ошибок модели, утечек, атак, изменений данных и отказов инфраструктуры. У каждого риска назначают owner и mitigation. [2]

Проверяемый пример. Оцените критичность ошибок в системе обращения за социальной помощью.

Вопрос для обсуждения. Как проверить выполнение такого требования: Оцените критичность ошибок в системе обращения за социальной помощью.

2.10 Технико-экономическое обоснование

Качество и стоимость оцениваются совместно; предпочтительна архитектура, удовлетворяющая требованиям при допустимом бюджете и сложно-

сти. [3]

Проверяемый пример. Сравните облачное inference и локальное CPU-развертывание в пилоте.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сравните облачное inference и локальное CPU-развертывание в пилоте.

2.11 Документирование решений

Architecture Decision Records фиксируют контекст, альтернативы, последствия и дату. Model card и data card дополняют техническое описание. [4]

Проверяемый пример. Подготовьте ADR для выбора TF-IDF+LinearSVC вместо тяжёлой нейронной модели.

Вопрос для обсуждения. Как проверить выполнение такого требования: Подготовьте ADR для выбора TF-IDF+LinearSVC вместо тяжёлой нейронной модели.

2.12 Защита итогового проекта

Демонстрация должна включать рабочий сценарий, ошибочный запрос, анализ метрик и краткий rollback plan. Жюри проверяет доказуемость утверждений, а не интерфейсную красоту. [5]

Проверяемый пример. Составьте план 8-минутной защиты с демонстрацией и ответами на инженерные вопросы.

Вопрос для обсуждения. Как проверить выполнение такого требования: Составьте план 8-минутной защиты с демонстрацией и ответами на инженерные вопросы.

3. Математические инструменты и количественные критерии

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

$$\min_{S}\Big(C_{\text{dev}} + C_{\text{serving}} + C_{\text{risk}}\Big) \tag{1}$$

$$\text{при} \quad F_1 \geq f_0, \quad p95 \leq \tau_0, \quad \text{Availability} \geq a_0 \tag{2}$$

Комплексный выбор архитектуры является условной оптимизацией: ограничения системы могут исключить модель с наибольшим offline-качеством.

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.

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

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, 3]

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
@dataclass
class ReleaseGate:
    min_macro_f1: float = 0.83
    max_p95_ms: float = 150.
    max_critical_errors: int = 0


def release_ok(metrics, gate=ReleaseGate()):
    return (metrics["macro_f1"] >= gate.min_macro_f1 and
            metrics["p95_ms"] <= gate.max_p95_ms and
            metrics["critical_errors"] <= gate.max_critical_errors)

candidate = {"macro_f1": 0.86, "p95_ms": 138., "critical_errors": 0}
print("release:", release_ok(candidate))
```

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 traceability(requirements):
```

```
    missing = [r["id"] for r in requirements if not r.get("test")]
    return {"covered": len(requirements) - len(missing), "missing": missing}
print(traceability([{"id":"R1","test":"test_api"}, {"id":"R2"}]))
```

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: Итоговая постановка

Заказчик использует университетский AI-сервис. Новое требование заставляет рассмотреть принцип «Итоговая постановка». Основная неопределённость состоит в том, как одновременно обеспечить выполнение требований приемки, не допустить необоснованную стоимость и сохранить возможность изменения модели.

Постановка. Комплексный проект соединяет бизнес-решение, данные, модель, сервис и процесс эксплуатации. Его спецификация должна раскрывать, почему AI необходим.

Действие инженера. Опишите минимальную версию сервиса автоматической классификации обращений с операторской проверкой.

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.

5.2 Ситуация 2: Структура репозитория

Заказчик использует университетский AI-сервис. Новое требование заставляет рассмотреть принцип «Структура репозитория». Основная неопределённость состоит в том, как одновременно обеспечить выполнение требований приемки, не допустить необоснованную стоимость и сохранить возможность изменения модели.

Постановка. Код отделяют от конфигураций, тестов, данных и артефактов. README должен позволять воспроизвести ключевой эксперимент.

Действие инженера. Разработайте дерево src/, tests/, configs/, pipelines/, docs/.

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: Документирование решений

Заказчик использует университетский AI-сервис. Новое требование заставляет рассмотреть принцип «Документирование решений». Основная неопределённость состоит в том, как одновременно обеспечить выполнение требований приемки, не допустить необоснованную стоимость и сохранить возможность изменения модели.

Постановка. Architecture Decision Records фиксируют контекст, альтернативы, последствия и дату. Model card и data card дополняют техническое описание.

Действие инженера. Подготовьте ADR для выбора TF-IDF+LinearSVC вместо тяжёлой нейронной модели.

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. **Итоговая постановка.** Опишите минимальную версию сервиса автоматической классификации обращений с операторской проверкой. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

2. **Контекстная архитектура.** Создайте контекстную и контейнерную схемы на основе C4. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

3. **Распределение требований.** Постройте traceability matrix из пяти требований и пяти проверок. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

4. **Данные и версионирование.** Спланируйте переход поля priority из строк в перечисление без поломки старой модели. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

5. **Экспериментальная методика.** Сформируйте таблицу F1, p95 latency, memory footprint и cost по двум кандидатам. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

6. **Структура репозитория.** Разработайте дерево src/, tests/, configs/, pipelines/, docs/. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

7. **Процесс релиза.** Опишите канареечный rollout 10% пользователей с условиями отмены. Сформулируйте проверяемое требование, проде-

монстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

8. **Наблюдаемость.** Создайте проект dashboard и примеры двух alerts. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

9. **Управление риском.** Оцените критичность ошибок в системе обращения за социальной помощью. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

10. **Технико-экономическое обоснование.** Сравните облачное inference и локальное CPU-развертывание в пилоте. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

11. **Документирование решений.** Подготовьте ADR для выбора TF-IDF+LinearSVC вместо тяжёлой нейронной модели. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

12. **Защита итогового проекта.** Составьте план 8-минутной защиты с демонстрацией и ответами на инженерные вопросы. 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 «Итоговая постановка»?

2. Как проверить на практике утверждение: «Опишите минимальную версию сервиса автоматической классификации обращений с операторской проверкой.»?

3. В чём назначение и ограничения принципа «Контекстная архитектура»?

4. Как проверить на практике утверждение: «Создайте контекстную и контейнерную схемы на основе C4.»?
5. В чём назначение и ограничения принципа «Распределение требований»?
6. Как проверить на практике утверждение: «Постройте traceability matrix из пяти требований и пяти проверок.»?
7. В чём назначение и ограничения принципа «Данные и версионирование»?
8. Как проверить на практике утверждение: «Спланируйте переход поля priority из строк в перечисление без поломки старой модели.»?
9. В чём назначение и ограничения принципа «Экспериментальная методика»?
10. Как проверить на практике утверждение: «Сформируйте таблицу F1, p95 latency, memory footprint и cost по двум кандидатам.»?
11. В чём назначение и ограничения принципа «Структура репозитория»?
12. Как проверить на практике утверждение: «Разработайте дерево src/, tests/, configs/, pipelines/, docs/.»?
13. В чём назначение и ограничения принципа «Процесс релиза»?
14. Как проверить на практике утверждение: «Опишите канареечный rollout 10% пользователей с условиями отмены.»?
15. В чём назначение и ограничения принципа «Наблюдаемость»?
16. Как проверить на практике утверждение: «Создайте проект dashboard и примеры двух alerts.»?
17. В чём назначение и ограничения принципа «Управление риском»?
18. Как проверить на практике утверждение: «Оцените критичность ошибок в системе обращения за социальной помощью.»?
19. В чём назначение и ограничения принципа «Технико-экономическое обоснование»?

20. Как проверить на практике утверждение: «Сравните облачное inference и локальное CPU-развертывание в пилоте.»?
21. В чём назначение и ограничения принципа «Документирование решений»?
22. Как проверить на практике утверждение: «Подготовьте ADR для выбора TF-IDF+LinearSVC вместо тяжёлой нейронной модели.»?
23. В чём назначение и ограничения принципа «Защита итогового проекта»?
24. Как проверить на практике утверждение: «Составьте план 8-минутной защиты с демонстрацией и ответами на инженерные вопросы.»?
25. Почему высокое offline-качество не гарантирует надежность промышленной AI-системы?
26. Какие артефакты нужны для воспроизводимости и анализа инцидента?

8. Глоссарий ключевых понятий

Итоговая постановка
Комплексный проект соединяет бизнес-решение, данные, модель, сервис и процесс эксплуатации.

Контекстная архитектура
На верхнем уровне фиксируют взаимодействие пользователя, внутренней платформы, внешних поставщиков и служб безопасности.

Распределение требований
Каждое требование должно иметь архитектурное место и тест: бизнес-критерии проверяются на уровне процесса, latency на уровне API, privacy в потоке данных.

Данные и версионирование
Сбор, валидаторы, разметка, lineage и train–serve consistency образуют цепочку.

Экспериментальная методика
Baseline и кандидат оцениваются на фиксированном test, валидация подбирает гиперпараметры, а отчет включает ошибки и неопределенность.

Структура репозитория
Код отделяют от конфигураций, тестов, данных и артефактов.

Процесс релиза
После unit и data tests модель проходит evaluation gate и staging.

Наблюдаемость
Эксплуатационный обзор содержит сервисные и модельные метрики, сигналы дрейфа и жалобы пользователей.

Управление риском
Составьте risk register для ошибок модели, утечек, атак, изменений данных и отказов инфраструктуры.

Технико-экономическое обоснование
Качество и стоимость оцениваются совместно; предпочтительна архитектура, удовлетворяющая требованиям при допустимом бюджете и сложности.

Документирование решений

Architecture Decision Records фиксируют контекст, альтернативы, последствия и дату.

Защита итогового проекта

Демонстрация должна включать рабочий сценарий, ошибочный запрос, анализ метрик и краткий rollback plan.

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

[4] MLflow. Official documentation: Tracking, Models, Registry. https://mlflow.org/docs/latest/index.html.

[5] FastAPI. Official documentation. https://fastapi.tiangolo.com/.

[6] Docker Inc. Docker documentation. https://docs.docker.com/.

[7] 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/.

[8] Mitchell M. et al. Model Cards for Model Reporting. FAT*, 2019. https://arxiv.org/abs/1810.03993.