АО «АЛМАТИНСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет инженерных и информационных технологий

Кафедра компьютерной инженерии

ПРОЕКТИРОВАНИЕ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

ЛЕКЦИЯ № 14

Мониторинг, надежность, масштабирование и безопасность AI-систем

Теоретико-практический конспект

Составитель: Алтыбай А.

Кафедра компьютерной инженерии

Алматы, 2026

Аннотация и назначение лекции

Лекция 14 раскрывает тему «Мониторинг, надежность, масштабирование и безопасность AI-систем» 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. Дрейф данных
2. Дрейф концепции
3. Статистики мониторинга
4. Мониторинг качества
5. Service SLO
6. План инцидента
7. Нагрузочное масштабирование
8. Robustness и OOD
9. Практические сценарии, контроль качества и проектные задания.

Ожидаемые результаты обучения

По завершении студент способен:

1. объяснить и аргументированно применить принцип «Дрейф данных» к инженерному решению;
2. объяснить и аргументированно применить принцип «Дрейф концепции» к инженерному решению;

3. объяснить и аргументированно применить принцип «Статистики мониторинга» к инженерному решению;
4. объяснить и аргументированно применить принцип «Мониторинг качества» к инженерному решению;
5. объяснить и аргументированно применить принцип «Service SLO» к инженерному решению;
6. объяснить и аргументированно применить принцип «План инцидента» 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 эксплуатируемую ML-платформу. Существенные условия: изменение распределения входов, деградацию модели. Основная проверяемая характеристика: долю инцидентов и время восстановления. 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. изучить наблюдаемость AI-систем, data/model drift, деградацию качества, latency/cost monitoring, стратегии масштабирования, rollback и базовые угрозы безопасности.

Ключевые понятия. observability, data drift, concept drift, model drift, latency, throughput, cost, robustness, canary, rollback, adversarial input, prompt injection.

1.1 Что мониторить

Мониторинг AI-сервиса включает как минимум четыре группы сигналов: инфраструктура (CPU/GPU, память, ошибки), сервис (latency, throughput, availability), данные (схема, пропуски, распределения) и модель (scores, calibration, delayed labels, business KPI).

Если реальные labels приходят с задержкой, нельзя ждать их единственного сигнала: нужно использовать proxy-метрики и drift detection.

1.2 Data drift, concept drift и model degradation

Data drift означает изменение распределения входов P(X). Concept drift означает изменение зависимости P(Y|X), когда прежняя связь между признаками и целью больше не работает. Деградация может происходить и без статистически сильного drift — например, из-за изменения downstream-процесса.

Drift detector не должен автоматически означать retrain. Сначала необходимо понять причину и проверить влияние на целевую метрику.

1.3 Latency, throughput и стоимость

Для production измеряют не только среднее, но и percentiles p50/p95/p99. В LLM-системах дополнительно полезны tokens/sec, time-to-first-token и стоимость по модели/запросу. Высокий tail latency может ухудшить пользовательский опыт даже при нормальном среднем.

Оптимизация включает batching, caching, autoscaling, quantization, выбор меньшей модели и маршрутизацию запросов.

1.4 Безопасность и robustness

AI-компонент добавляет новые поверхности атаки: adversarial inputs, data poisoning, model extraction, prompt injection, unsafe tool use и утечку кон-

текста. Защита включает validation, least privilege, изоляцию инструментов, rate limits, content policies, secrets management и аудит.

Robustness означает устойчивость к допустимым изменениям входа и отказам зависимостей. Для высокорисковых сценариев нужен safe fallback и human-in-the-loop.

1.5 Canary, rollback и incident response

Canary deployment направляет небольшой процент трафика новой версии и сравнивает метрики. При нарушении guardrail система должна быстро откатиться на стабильную версию. Rollback следует проектировать заранее: если артефакт или schema несовместимы, откат может быть сложнее простого переключения версии.

После инцидента проводится postmortem с причиной, влиянием, действиями и профилактикой.

1.6 Выводы

▷ AI observability объединяет инфраструктурные, data, model и business сигналы.

▷ Drift — индикатор изменения среды, а не автоматическая команда на retraining.

▷ Tail latency и стоимость являются полноценными метриками AI-системы.

▷ Безопасность должна учитывать специфические угрозы моделей и LLM.

▷ Rollback и incident response проектируются до инцидента.

1.7 Контрольные вопросы

▷ Чем data drift отличается от concept drift?

- Какие метрики сервиса следует мониторить кроме accuracy?
- Что такое p95 latency?
- Какие угрозы характерны для LLM-приложений?
- Почему canary deployment снижает риск обновления модели?

2. Систематическое изложение темы и инженерные решения

2.1 Дрейф данных

Covariate drift описывает изменение распределения входных признаков; его наличие не доказывает падение качества. Изменение доли меток и связи X–Y требует отдельного анализа. [2]

Проверяемый пример. Сравните распределения длин обращений за два месяца.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сравните распределения длин обращений за два месяца.

2.2 Дрейф концепции

Concept drift меняет условную зависимость между признаками и целью, поэтому модель может деградировать без заметного изменения признаков. [3]

Проверяемый пример. Покажите, как изменение правил приоритета меняет метки при прежних текстах.

Вопрос для обсуждения. Как проверить выполнение такого требования: Покажите, как изменение правил приоритета меняет метки при прежних текстах.

2.3 Статистики мониторинга

PSI, KS и Jensen–Shannon divergence полезны как сигналы, но зависят от размера выборки и бинов. Порог требует калибровки под продукт. [4]

Проверяемый пример. Посчитайте PSI по распределениям двух категориальных признаков.

Вопрос для обсуждения. Как проверить выполнение такого требования: Посчитайте PSI по распределениям двух категориальных признаков.

2.4 Мониторинг качества

При delayed labels online качество невозможно знать немедленно. Вместо него используют прокси, ручную разметку и последующую метрику на созревших метках. [5]

Проверяемый пример. Опишите dashboard с recall по готовым меткам последнего месяца.

Вопрос для обсуждения. Как проверить выполнение такого требования: Опишите dashboard с recall по готовым меткам последнего месяца.

2.5 Service SLO

Availability, p95/p99, доля 5xx и saturation отражают качество сервиса. Error budget задает допустимое нарушение SLO за период. [6]

Проверяемый пример. Оцените, сколько минут простоя допустимо при доступности 99.9% за 30 дней.

Вопрос для обсуждения. Как проверить выполнение такого требования: Оцените, сколько минут простоя допустимо при доступности 99.9% за 30 дней.

2.6 План инцидента

Runbook указывает, как определить проблему, ограничить ущерб, откатить версию, уведомить владельцев и зафиксировать выводы. [1]

Проверяемый пример. Составьте инструкцию при удвоении числа ошибочных предсказаний после релиза.

Вопрос для обсуждения. Как проверить выполнение такого требования: Составьте инструкцию при удвоении числа ошибочных предсказаний после релиза.

2.7 Нагрузочное масштабирование

Autoscaling полезен при вариативном потоке запросов, но холодный старт и дефицит GPU создают задержку. Профиль нагрузки важнее пиковой цифры. [2]

Проверяемый пример. Спроектируйте queue + worker pool для скачкообразных поступлений изображений.

Вопрос для обсуждения. Как проверить выполнение такого требования: Спроектируйте queue + worker pool для скачкообразных поступлений изображений.

2.8 Robustness и OOD

Out-of-distribution входы, шум и adversarial примеры могут нарушать предпосылки модели. Детектор неопределенности не гарантирует безопасность, его проверяют отдельно. [3]

Проверяемый пример. Проверьте, как классификатор реагирует на текст на незнакомом языке.

Вопрос для обсуждения. Как проверить выполнение такого требования: Проверьте, как классификатор реагирует на текст на незнакомом языке.

2.9 Безопасность модели

Опасны model extraction, data poisoning, prompt injection и небезопасная десериализация моделей. Не доверяйте загружаемым артефактам. [4]

Проверяемый пример. Используйте безопасный формат хранения и проверку подписи поступившего model artifact.

Вопрос для обсуждения. Как проверить выполнение такого требования: Используйте безопасный формат хранения и проверку подписи поступившего model artifact.

2.10 Приватность логов

Inference logs могут содержать персональные данные; retention, redaction и доступ должны быть ограничены. [5]

Проверяемый пример. Сделайте журнал с request_id, latency, model_version без исходного личного текста.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сделайте журнал с request_id, latency, model_version без исходного личного текста.

2.11 Экономика эксплуатации

Стоимость складывается из постоянных ресурсов и затрат на запрос; capacity planning учитывает пики, пакетирование и запас надежности. [6]

Проверяемый пример. Рассчитайте месячную стоимость на 100k запросов при заданной цене сервера.

Вопрос для обсуждения. Как проверить выполнение такого требования: Рассчитайте месячную стоимость на 100k запросов при заданной цене сервера.

2.12 Postmortem и улучшение

Разбор инцидента отделяет непосредственную ошибку от системной причины и фиксирует действия, которые проверяются тестами. [1]

Проверяемый пример. После отказа индекса добавьте проверку готовности и сценарий восстановления.

Вопрос для обсуждения. Как проверить выполнение такого требования: После отказа индекса добавьте проверку готовности и сценарий восстановления.

3. Математические инструменты и количественные критерии

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

$$\mathrm{PSI} = \sum_{b=1}^{B} (p_b - q_b) \ln \frac{p_b}{q_b} \tag{1}$$

$$\text{Availability} = 1 - \frac{T_{\text{downtime}}}{T_{\text{total}}} \tag{2}$$

При расчёте PSI применяют фиксированные бины и аккуратную обработку нулевых долей; сам по себе PSI не измеряет падение качества.

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.

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

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
import numpy as np

def psi(ref, cur, eps=1e-9):
    ref = np.asarray(ref, dtype=float)
    cur = np.asarray(cur, dtype=float)
    a = np.maximum(ref / ref.sum(), eps)
    b = np.maximum(cur / cur.sum(), eps)
    return float(np.sum((b - a) * np.log(b / a)))

baseline = [30, 50, 20]
current = [15, 55, 30]
print("PSI:", round(psi(baseline, current), 3))
# Equal bin definitions and comparable sampling are mandatory.
```

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
total_minutes = 30 * 24 * 60
allowed = total_minutes * (1 - 0.999)
print("downtime budget (minutes):", round(allowed, 2))
```

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: Дрейф данных

Заказчик использует эксплуатируемую ML-платформу. Новое требование заставляет рассмотреть принцип «Дрейф данных». Основная неопределённость состоит в том, как одновременно обеспечить долю инцидентов и время восстановления, не допустить деградацию модели и сохранить возможность изменения модели.

Постановка. Covariate drift описывает изменение распределения входных признаков; его наличие не доказывает падение качества. Изменение доли меток и связи X–Y требует отдельного анализа.

Действие инженера. Сравните распределения длин обращений за два месяца.

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: План инцидента

Заказчик использует эксплуатируемую ML-платформу. Новое требование заставляет рассмотреть принцип «План инцидента». Основная неопределённость состоит в том, как одновременно обеспечить долю инцидентов и время восстановления, не допустить деградацию модели и сохранить возможность изменения модели.

Постановка. Runbook указывает, как определить проблему, ограничить ущерб, откатить версию, уведомить владельцев и зафиксировать выводы.

Действие инженера. Составьте инструкцию при удвоении числа ошибочных предсказаний после релиза.

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: Экономика эксплуатации

Заказчик использует эксплуатируемую ML-платформу. Новое требование заставляет рассмотреть принцип «Экономика эксплуатации». Основная неопределённость состоит в том, как одновременно обеспечить долю инцидентов и время восстановления, не допустить деградацию модели и сохранить возможность изменения модели.

Постановка. Стоимость складывается из постоянных ресурсов и затрат на запрос; capacity planning учитывает пики, пакетирование и запас надежности.

Действие инженера. Рассчитайте месячную стоимость на 100k запросов при заданной цене сервера.

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. **Дрейф данных.** Сравните распределения длин обращений за два месяца. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.
2. **Дрейф концепции.** Покажите, как изменение правил приоритета меняет метки при прежних текстах. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.
3. **Статистики мониторинга.** Посчитайте PSI по распределениям двух категориальных признаков. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.
4. **Мониторинг качества.** Опишите dashboard с recall по готовым меткам последнего месяца. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.
5. **Service SLO.** Оцените, сколько минут простоя допустимо при доступности 99.9% за 30 дней. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.
6. **План инцидента.** Составьте инструкцию при удвоении числа ошибочных предсказаний после релиза. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.
7. **Нагрузочное масштабирование.** Спроектируйте queue + worker pool для скачкообразных поступлений изображений. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите

протокол хотя бы одного эксперимента.

8. **Robustness и OOD.** Проверьте, как классификатор реагирует на текст на незнакомом языке. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

9. **Безопасность модели.** Используйте безопасный формат хранения и проверку подписи поступившего model artifact. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

10. **Приватность логов.** Сделайте журнал с request_id, latency, model_version без исходного личного текста. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

11. **Экономика эксплуатации.** Рассчитайте месячную стоимость на 100k запросов при заданной цене сервера. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

12. **Postmortem и улучшение.** После отказа индекса добавьте проверку готовности и сценарий восстановления. 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 «Дрейф данных»?
2. Как проверить на практике утверждение: «Сравните распределения длин обращений за два месяца.»?
3. В чём назначение и ограничения принципа «Дрейф концепции»?
4. Как проверить на практике утверждение: «Покажите, как изменение правил приоритета меняет метки при прежних текстах.»?

5. В чём назначение и ограничения принципа «Статистики мониторинга»?
6. Как проверить на практике утверждение: «Посчитайте PSI по распределениям двух категориальных признаков.»?
7. В чём назначение и ограничения принципа «Мониторинг качества»?
8. Как проверить на практике утверждение: «Опишите dashboard с recall по готовым меткам последнего месяца.»?
9. В чём назначение и ограничения принципа «Service SLO»?
10. Как проверить на практике утверждение: «Оцените, сколько минут простоя допустимо при доступности 99.9% за 30 дней.»?
11. В чём назначение и ограничения принципа «План инцидента»?
12. Как проверить на практике утверждение: «Составьте инструкцию при удвоении числа ошибочных предсказаний после релиза.»?
13. В чём назначение и ограничения принципа «Нагрузочное масштабирование»?
14. Как проверить на практике утверждение: «Спроектируйте queue + worker pool для скачкообразных поступлений изображений.»?
15. В чём назначение и ограничения принципа «Robustness и OOD»?
16. Как проверить на практике утверждение: «Проверьте, как классификатор реагирует на текст на незнакомом языке.»?
17. В чём назначение и ограничения принципа «Безопасность модели»?
18. Как проверить на практике утверждение: «Используйте безопасный формат хранения и проверку подписи поступившего model artifact.»?
19. В чём назначение и ограничения принципа «Приватность логов»?
20. Как проверить на практике утверждение: «Сделайте журнал с request_id, latency, model_version без исходного личного текста.»?
21. В чём назначение и ограничения принципа «Экономика эксплуатации»?

22. Как проверить на практике утверждение: «Рассчитайте месячную стоимость на 100k запросов при заданной цене сервера.»?
23. В чём назначение и ограничения принципа «Postmortem и улучшение»?
24. Как проверить на практике утверждение: «После отказа индекса добавьте проверку готовности и сценарий восстановления.»?
25. Почему высокое offline-качество не гарантирует надежность промышленной AI-системы?
26. Какие артефакты нужны для воспроизводимости и анализа инцидента?

8. Глоссарий ключевых понятий

Дрейф данных
Covariate drift описывает изменение распределения входных признаков; его наличие не доказывает падение качества.

Дрейф концепции
Concept drift меняет условную зависимость между признаками и целью, поэтому модель может деградировать без заметного изменения признаков.

Статистики мониторинга
PSI, KS и Jensen–Shannon divergence полезны как сигналы, но зависят от размера выборки и бинов.

Мониторинг качества
При delayed labels online качество невозможно знать немедленно.

Service SLO
Availability, p95/p99, доля 5xx и saturation отражают качество сервиса.

План инцидента
Runbook указывает, как определить проблему, ограничить ущерб, откатить версию, уведомить владельцев и зафиксировать выводы.

Нагрузочное масштабирование
Autoscaling полезен при вариативном потоке запросов, но холодный старт и дефицит GPU создают задержку.

Robustness и OOD
Out-of-distribution входы, шум и adversarial примеры могут нарушать предпосылки модели.

Безопасность модели
Опасны model extraction, data poisoning, prompt injection и небезопасная десериализация моделей.

Приватность логов
Inference logs могут содержать персональные данные; retention, redaction и доступ должны быть ограничены.

Экономика эксплуатации

Стоимость складывается из постоянных ресурсов и затрат на запрос; capacity planning учитывает пики, пакетирование и запас надежности.

Postmortem и улучшение

Разбор инцидента отделяет непосредственную ошибку от системной причины и фиксирует действия, которые проверяются тестами.

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] 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/.

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

[5] OWASP. Top 10 for Large Language Model Applications. https://genai.owasp.org/llm-top-10/.

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