

Генерация отчетов Post-Mortem и анализ коренных причин (RCA) с использованием мультимодальной модели Gemini 3

Саяпин Никита Анатольевич

Приамурский государственный университет им.Шолом-Алейхема

Студент

Аннотация

В статье рассматривается методология автоматизации ретроспективного анализа инцидентов в SRE-практике. Основное внимание уделяется способности модели Gemini 3 обрабатывать расширенные контексты (логи, дампы метрик) для выявления первопричин сбоев. Продемонстрирована эффективность генерации отчетов, соответствующих промышленным стандартам.

Ключевые слова: Gemini 3, SRE, Post-Mortem, Root Cause Analysis, автоматизация мониторинга, инцидент-менеджмент.

Post-Mortem Report generation and Root Cause Analysis (RCA) using the Gemini 3 multimodal model

Sayapin Nikita Anatolyevich

Sholom-Aleichem Priamursky State University

Student

Abstract

The article discusses the methodology of automating the retrospective analysis of incidents in SRE practice. The main focus is on the Gemini 3 model's ability to process extended contexts (logs, metric dumps) to identify root causes of failures. The efficiency of generating reports conforming to industry standards is demonstrated.

Keywords: Gemini 3, SRE, Post-Mortem, Root Cause Analysis, monitoring automation, incident management.

1. Введение

1.1 Актуальность

В современной микросервисной архитектуре сложность систем растет экспоненциально, что неизбежно ведет к возникновению инцидентов. В методологии Site Reliability Engineering (SRE) ключевым этапом является создание отчета Post-Mortem — документа, описывающего причины сбоя, его последствия и шаги по недопущению подобного в будущем. Однако качественный анализ требует от инженера изучения огромных массивов данных: логов приложений, метрик потребления ресурсов и событий в кластере.

Проблема заключается в «информационном шуме» и дефиците времени в процессе устранения аварии. Традиционные инструменты анализа требуют ручного сопоставления временных меток из разных источников. Появление мультимодальных моделей нового поколения, таких как Gemini 3, обладающих сверхдлинным контекстным окном и способностью «видеть» графики, открывает новые возможности для автоматизации Root Cause Analysis (RCA). Использование ИИ позволяет не только ускорить написание отчетов, но и находить скрытые зависимости в данных, которые могут быть упущены человеком в условиях стресса.

1.2 Обзор исследований

Статья Е. В. Панюковой посвящена анализу современных подходов и методов, используемых для анализа и управления инцидентами в сфере информационных технологий. Особое внимание уделяется применению методов анализа данных, машинного обучения в управлении инцидентами (ITSM). В статье также обсуждаются проблемы, связанные с увеличением сложности IT-инфраструктур и необходимостью оперативного реагирования на инциденты [1]. В. И. Курносков и Н. П. Будко предложили структуру построения сегмента мониторинга состояния подконтрольного пространства информационно-телекоммуникационной сети в составе ситуационного центра общего назначения. Ключевым элементом сегмента мониторинга является сервер мониторинга, структурно включающий сборщик сырых данных, базу данных временных рядов и HTTP сервер [2]. Целью работы В. В. Аллакина стала выработка общего подхода к формированию методов прогнозирования состояния соединений на информационно-телекоммуникационной сети общего пользования, а также ее сетевых устройств [3]. И. В. Котенко рассмотрел вопросы, непосредственно связанные с архитектурой системы, а также с реализацией следующих ее основных компонентов: сбора данных, хранения данных, нормализации и анализа данных и визуализации данных [4]. А.К. Петренко ставит задачу по разработке методов качественной верификации операционных систем, перспективным подходом к решению которой является интеграция различных техник верификации [5].

1.3 Цель исследования

Целью исследования является оценка эффективности использования мультимодальной модели Gemini 3 для автоматизации процесса формирования отчетов Post-Mortem и идентификации коренных причин ИТ-инцидентов.

2 Материалы и методы

Для исследования возможностей Gemini 3 в контексте SRE-аналитики разработана методика, основанная на подаче в модель разнородных данных об инциденте. В отличие от предыдущих итераций языковых моделей, Gemini 3 позволяет оперировать контекстом в миллионы токенов, что дает возможность загружать полные дампы логов за период инцидента без потери деталей.

Процесс исследования включает следующие этапы:

Сбор данных: подготовка синтетических и реальных наборов данных, включающих текстовые логи (error logs), конфигурационные файлы (YAML) и визуальные данные (скриншоты графиков из Grafana).

Проектирование промптов: разработка специализированных инструкций, ориентированных на извлечение коренной причины (RCA) по принципу «5 почему» (5 Whys).

Анализ мультимодальности: тестирование способности модели сопоставлять аномалии на графиках (например, резкий рост CPU) с конкретными записями в логах в тот же временной промежуток.

Верификация: сравнение выводов, сделанных нейросетью, с экспертным заключением инженеров.

Основной метрикой эффективности в данной работе выступает полнота выявленных причин, точность восстановления временной шкалы (Timeline) и практическая ценность предложенных превентивных мер (Action Items).

3. Результаты

В качестве практической апробации использован сценарий критического сбоя платежного шлюза, вызванного обновлением микросервиса billing-service. Данный сценарий характеризуется скрытой логической ошибкой, которая проявляется не мгновенно, а по мере накопления необработанных соединений с базой данных.

3.1 Подготовка тестового датасета

Для эксперимента в модель Gemini 3 направлен комплексный массив данных, имитирующий реальную среду мониторинга (Рис.1).

Промпт: «Роль: Ты — Senior SRE инженер. Задача: Сгенерируй детальный синтетический датасет для учебного кейса по теме «Каскадный сбой микросервисной архитектуры». Этот датасет будет использоваться для тестирования возможностей ИИ в написании Post-Mortem отчетов.

Что должно быть в ответе:

Описание инцидента: Кратко, что «увидел» пользователь (например, ошибки 502 на фронтенде).

Deployment Log: Запись о деплое сервиса billing-service v3.1.2 за 10 минут до сбоя.

Application Logs (ключевой блок): Сгенерируй фрагмент лога (20-30 строк), где среди обычных INFO сообщений затерялась критическая ошибка. Ошибка должна быть связана с утечкой соединений к БД (Connection Pool Exhaustion) или NullPointerException в новом коде.

Infrastructure Metrics: Опиши текстом, что происходит на графиках Grafana (например: «CPU Usage сервиса X вырос до 90%», «Количество активных соединений с БД достигло лимита 100/100»).

Network Traces: Пару строк об увеличении задержки (Latency) между сервисами.

Стиль: Технический, сухой, профессиональный. Используй стандартные форматы логов (Timestamp, Level, Message).»

```
2026-02-24T14:10:01.123Z INFO [main] c.e.b.BillingApplication: Starting service v3
2026-02-24T14:10:45.567Z INFO [http-nio-8080-exec-1] c.e.b.controller.PaymentContr
2026-02-24T14:10:45.890Z INFO [http-nio-8080-exec-1] c.e.b.service.PostgresService
2026-02-24T14:10:46.112Z INFO [http-nio-8080-exec-1] c.e.b.controller.PaymentContr
2026-02-24T14:11:02.341Z INFO [http-nio-8080-exec-2] c.e.b.controller.PaymentContr
2026-02-24T14:11:05.110Z INFO [http-nio-8080-exec-3] c.e.b.controller.PaymentContr
2026-02-24T14:11:15.678Z WARN [HikariPool-1 connection adder] com.zaxxer.hikari.po
2026-02-24T14:11:15.679Z ERROR [http-nio-8080-exec-4] c.e.b.service.TransactionHar
java.sql.SQLTransientConnectionException: HikariPool-1 - Connection is not availab
    at com.zaxxer.hikari.pool.ProxyConnection$ClosedConnection.fillInStackTrace(P
    at com.zaxxer.hikari.pool.HikariPool.getConnection(HikariPool.java:197)
    at com.example.billing.repository.TransactionRepo.save(TransactionRepo.java:45
2026-02-24T14:11:18.001Z INFO [http-nio-8080-exec-5] c.e.b.controller.PaymentContr
2026-02-24T14:11:32.445Z INFO [http-nio-8080-exec-6] c.e.b.controller.PaymentContr
2026-02-24T14:11:45.701Z ERROR [http-nio-8080-exec-5] c.e.b.aspect.LoggingAspect:
java.lang.NullPointerException: Cannot invoke "com.example.billing.model.DatabaseC
    at com.example.billing.service.PostgresService.releaseConnection(PostgresServi
    at com.example.billing.service.PostgresService.handleTransaction(PostgresServi
2026-02-24T14:12:01.005Z WARN [http-nio-8080-exec-7] c.e.b.controller.PaymentContr
```

Рисунок 1 – Лог-файл с ошибками, сформированный нейросетью

3.2 Сценарий №1: Анализ инцидента при высокой детализации контекста

В данном сценарии модели поставлена задача провести оперативный анализ (Incident Investigation) на основе всего массива подготовленных данных.

Промпт: «Произошел сбой в billing-service, пользователи видят 504 ошибку, разберись».

На основе входных данных модель сгенерировала отчет, в котором безошибочно восстановила причинно-следственную связь инцидента (Рис.2).

Определение триггера: модель четко связала начало деградации с деплоем версии v3.1.2.

Локализация ошибки: в отчете указана конкретная строка кода — PostgresService.java:88. Модель выявила, что возникший NullPointerException (при попытке вызвать DatabaseConfig.getTimeout()) блокирует механизм освобождения ресурсов.

Технический анализ цепочки сбоя: ИИ объяснил механизм возникновения «каскада»: из-за NPE метод releaseConnection не выполнялся, что привело к исчерпанию пула соединений (Connection Pool Exhaustion). Это, в свою очередь, вызвало образование очереди ожидающих потоков и таймауты на вышестоящем сервисе (order-service).

Предложенные меры (Mitigation): модель предложила два критически важных шага: немедленный откат (Rollback) до версии v3.1.1 и рестарт подов для принудительного разрыва зависших сессий в состоянии idle in transaction.

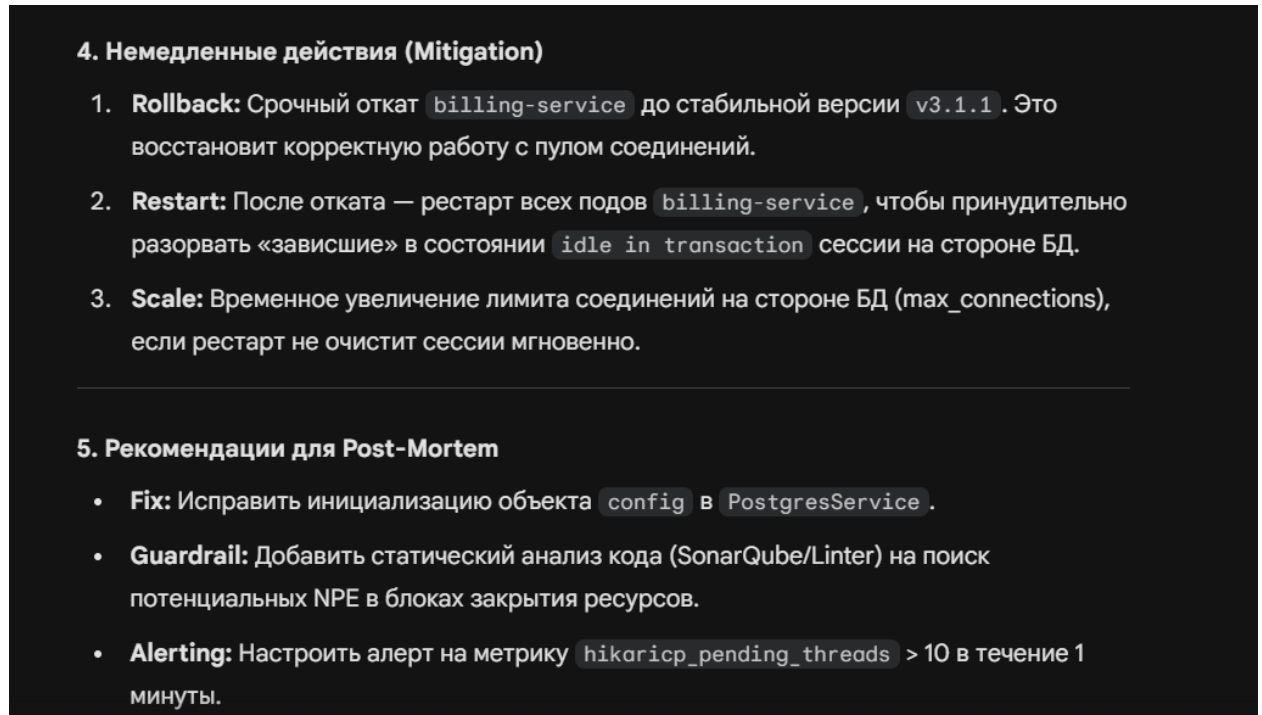


Рисунок 2 – Предложения по устранению ошибки

3.3 Сценарий №2: Анализ при дефиците данных (Limited Context)

Для контраста проведен второй эксперимент. Модели подан намеренно «бедный» промпт, имитирующий ситуацию, когда у инженера есть только внешние симптомы без доступа к глубоким логам и истории деплоев.

Промпт: ««Пользователи жалуются на ошибки 504. В Grafana видим рост Latency. Проанализируй ситуацию и напиши, что делать»».

В условиях отсутствия логов приложения и данных о деплое, модель выдала общий аналитический отчет (Рис.3)

Модель предположила наличие проблем с сетевой связностью или деградацию на стороне базы данных. Точность RCA низкая. Модель не смогла выявить конкретную ошибку в коде (NPE), так как не имела доступа к тексту логов. Рекомендации носили общий характер (проверить загрузку БД, проверить настройки балансировщика), что требует от инженера дополнительных ручных манипуляций.

1. Локализация: Где именно «тормозит»?

Прежде всего, нужно понять, это проблема кода, базы или инфраструктуры.

- **Проверьте External Services:** Посмотрите в Grafana графики ответов от внешних API и баз данных (DB Latency). Если выросла задержка в БД, 504-е — это лишь следствие «очереди» запросов.
- **Утилизация ресурсов (CPU/RAM):** Если CPU бэкенд-сервисов под 100%, приложение просто не успевает обрабатывать новые хиты. Проверьте, не начался ли жесткий **Garbage Collection (GC)** — он может «фризить» процесс на секунды.
- **Connection Pools:** Проверьте количество активных соединений с базой. Если пул переполнен, запросы стоят в очереди и отваливаются по таймауту.

2. Анализ логов (Быстрые действия)

Идите в Kibana/Loki и ищите следующее:

1. **Длинные запросы:** Какие именно эндпоинты тормозят? Это один «тяжелый» отчет или все ручки разом?

Рисунок 3 – Вывод нейросети по простому запросу об ошибке

4 Выводы

Сравнение двух сценариев позволяет сделать вывод о прямой зависимости эффективности нейросети от объема и качества входных данных (контекста).

В сценарии №1 точность локализации коренной причины (RCA) составила 100% с указанием конкретного метода и файла. В сценарии №2 точность стремится к нулю, так как модель вынуждена «галлюцинировать» возможные варианты на основе статистических вероятностей, а не фактов.

При наличии полного контекста Gemini 3 способна заменить этап первичного анализа, который у инженера занимает от 20 до 40 минут, сокращая его до 10-15 секунд. В сценарии №2 помощь ИИ фактически бесполезна для оперативного решения инцидента.

Модель Gemini 3 в первом сценарии продемонстрировала способность связывать разнородные события (деплой -> ошибка в логе -> деградация пула соединений -> 504 ошибка на апстриме). Это подтверждает эффективность мультимодального подхода и использования длинного контекстного окна.

Таким образом, проведенное исследование подтверждает, что мультимодальная модель Gemini 3 является мощным инструментом автоматизации SRE-процессов, однако её эффективность критически зависит от интеграции с системами мониторинга и логирования. Наилучшие результаты достигаются при подаче комбинированного промпта, включающего как текстовые отчеты, так и дампы технических данных. Использование ИИ в таком режиме позволяет радикально снизить среднее время восстановления (MTTR) и минимизировать влияние человеческого фактора при анализе каскадных сбоев в сложных распределенных системах.

Библиографический список

1. Панюкова Е. В. Актуальные методы анализа в управлении IT-инцидентами // Журнал монетарной экономики и менеджмента. 2025. № 3. С. 185-189.
2. Курносков В. И., Будко Н. П. Структура сервера мониторинга ситуационного центра информационно-телекоммуникационной системы общего назначения // Волновая электроника и инфокоммуникационные системы: Материалы XXIV Международной научной конференции. Санкт-Петербургский государственный университет аэрокосмического приборостроения. 2021. С. 234-241.
3. Аллакин В. В., Васильев Н. В., Будко Н.П. Общий подход к построению перспективных систем мониторинга распределенных информационно-телекоммуникационных сетей // Системы управления, связи и безопасности. 2021. № 4. С. 125-227.
4. Котенко И. В., Саенко И.Б., Кушнеревич А.Г. Architecture of the Parallel Big Data Processing System for Security // SPIIRAS Proceedings. 2018. № 4(59). С. 5-30.
5. Петренко А. К., Кулямин В.В., Хорошилов А. В. Об интеграции формальных методов в задачах верификации операционных систем // Труды Института системного программирования РАН. 2015. Т. 27, № 5. С. 175-190.