DOI https://doi.org/10.46845/2541-8254-2026-2(54)-70-70
УДК 004.891.2

РАЗРАБОТКА ГИБРИДНОГО ПОДХОДА К АНАЛИЗУ ДАННЫХ ОТ ОБРАТНОЙ СВЯЗИ ДЛЯ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ ПРОЦЕССА СОПРОВОЖДЕНИЯ ПРИКЛАДНОГО ПО

М. П. Рождественский, студент 1-го курса магистратуры
E-mail: maksim.rozhdestvenskiy@yandex.ru
ФГБОУ ВО «Калининградский государственный технический университет»

М. А. Романов, старший преподаватель
E-mail: mikhail.romanov@klgtu.ru
ФГБОУ ВО «Калининградский государственный технический университет»

В статье предложен гибридный подход к семантическому анализу неструктурированной обратной связи, позволяющий автоматизировать первичный триаж дефектов при сопровождении прикладного программного обеспечения. Разработана архитектура системы поддержки принятия решений, объединяющая символьный контур на основе графовой онтологии дефектов (Neo4j) и нейросетевой контур на базе языковых моделей с инференсом через ONNX и TensorFlow.js. Предложен механизм адаптивного согласования результатов, динамически балансирующий доверие к каждому контуру в зависимости от длины входного текста, а также формализована метрика уверенности, определяющая режим работы системы (автоматический, СППР, ручной). Описаны модульный ETL-конвейер с автоматизированными контрольными точками качества, стратегия эмпирически обоснованной синтетической генерации данных и план baseline-моделирования. Подход обеспечивает интерпретируемость решений, устойчивость к лексическому разнообразию русскоязычных обращений и соответствует принципам ответственного искусственного интеллекта.

Ключевые слова: *гибридный подход, семантический анализ, неструктурированная обратная связь, триаж дефектов, сопровождение программного обеспечения, графовая онтология, символьный контур, нейросетевая классификация, трансформерные модели, адаптивное согласование, система поддержки принятия решений (СППР).*

ВВЕДЕНИЕ

В условиях экспоненциального роста объема неструктурированной обратной связи процессы сопровождения современного прикладного программного обеспечения сталкиваются с критическими вызовами. Мобильные приложения генерируют десятки тысяч пользовательских отзывов ежемесячно, корпоративные системы накапливают тысячи тикетов технической поддержки, а логи ошибок содержат непрерывный поток текстовых сообщений. При этом в условиях информационной перегрузки критичные дефекты нередко остаются незамеченными или получают заниженный приоритет, что напрямую влияет на стабильность продукта, сроки устранения неисправностей и пользовательский опыт. Существующие подходы к автоматизации анализа текстовой обратной связи обладают существенными ограничениями. Статистические методы (например, векторизация на основе TF-IDF) не учитывают семантическую близость синонимичных формулировок ошибок и демонстрируют низкую устойчивость к лексическому разнообразию пользовательского языка. Системы, основанные на правилах, требуют постоянного ручного обновления шаблонов и неадаптивны к новым, ранее не встречавшимся типам дефектов. Современные трансформерные модели демонстри-

руют высокую точность классификации, но функционируют как «черный ящик», лишая аналитика возможности интерпретировать рекомендации, что недопустимо в процессах сопровождения критичного ПО и нарушает принципы ответственного ИИ.

Целью настоящей работы является разработка гибридного подхода к семантическому анализу неструктурированных данных обратной связи для автоматизации первичного триажа дефектов при сопровождении прикладного программного обеспечения. Для ее достижения решаются следующие задачи:

- Проектирование графовой онтологии дефектов ПО и интеграция ее в символьный контур анализа.
- Разработка модульного ETL-конвейера с автоматизированными контрольными точками качества (quality gates) и аудитом данных.
- Обоснование стратегии эмпирически сгенерированных обучающих данных и формирование воспроизводимого датасета с контролем утечек таргета.
- Проектирование архитектуры системы поддержки принятия решений (ИСППР) с механизмом адаптивного согласования результатов символьного и нейросетевого компонентов.
- Планирование и формализация baseline-моделирования для доказательства предсказуемости сигнала в данных перед внедрением гибридного конвейера.

ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ И ВЫЯВЛЕНИЕ ПРОБЕЛОВ

Проблема автоматизации первичного триажа дефектов и анализа неструктурированной обратной связи активно исследуется в области программной инженерии и обработки естественного языка. Существующие подходы можно классифицировать на три группы: академические исследовательские системы, коммерческие платформы автоматизации и открытые NLP-инструменты. Ниже проводится систематизация ключевых аналогов, их сравнительный анализ и формулировка выявленных пробелов, закрываемых предлагаемым гибридным подходом (таблица 1).

Таблица 1 – Сравнительная характеристика существующих решений и предлагаемой системы

Решение	Тип данных	Язык	Интерпретируемость	Онтология	Режим работы
DeepTriage	Баг-репорты (EN)	Английский	Низкая (черный ящик)	Отсутствует	Пакетный
Jira Automation	Тикеты Jira	Многоязычный	Низкая (проприетарный)	Отсутствует	Потоковый/Пакетный
Bug-Classification-NLP	GitHub Issues (EN)	Английский	Средняя (feature importance)	Отсутствует	API-сервис
Предлагаемое решение	Отзывы, тикеты, логи (RU)	Русский	Высокая (пояснения + уверенность)	Neo4j (графовая)	Интерактивная СППР

Анализ аналогов позволяет сформулировать следующие методологические пробелы в области автоматизации триажа дефектов:

- Отсутствие русскоязычных систем, объединяющих семантический анализ текстовой обратной связи с формализованной моделью предметной области (онтологией дефектов ПО).
- Слабая интерпретируемость нейросетевых классификаторов, препятствующая их внедрению в процессы сопровождения, где требуется объяснимость каждого решения.
- Отсутствие воспроизводимых стратегий генерации и валидации обучающих выборок в условиях дефицита размеченных данных для редких типов дефектов.
- Неадаптированность механизмов согласования результатов для гибридных систем: большинство подходов либо полностью автоматизированы, либо требуют ручной разметки без промежуточного слоя СППР.

Предлагаемый подход закрывает выявленные пробелы за счет интеграции символьных методов (интерпретируемость, онтология) и субсимвольных моделей (точность, адаптивность), формируя методологическую базу для создания прозрачных и воспроизводимых систем анализа обратной связи в русскоязычном сегменте.

МЕТОДОЛОГИЯ ГИБРИДНОГО ПОДХОДА

Предлагаемый методологический аппарат построен на адаптации стандарта CRISP-DM (Cross-Industry Standard Process for Data Mining) [1] к задачам семантического анализа неструктурированной обратной связи в процессах сопровождения прикладного ПО. Ключевым отличием от классического подхода является интеграция контура поддержки принятия решений (human-in-the-loop) на этапе Evaluation и внедрение механизма адаптивного согласования на этапе Modeling. Ниже детализируются компоненты гибридного конвейера, формализация метрики уверенности и архитектура согласования результатов.

В рамках исследования этапы CRISP-DM [1] трансформированы следующим образом:

- Business Understanding → Формализация бизнес-метрик: сокращение времени триажа, полнота выявления критичных дефектов, точность классификации.
- Data Understanding & Preparation → Эмпирически обоснованная синтетическая генерация данных, проектирование ETL-конвейера с quality gates и стратифицированным разбиением (70/15/15).
- Modeling → Параллельное выполнение символьного (онтология + правила) и нейросетевого контуров с последующим адаптивным согласованием.
- Evaluation → Двухуровневая валидация: автоматический расчет метрик качества (Macro-F1, PR-AUC) + экспертная оценка интерпретируемости.
- Deployment → Развертывание в режиме СППР с механизмом логирования экспертных коррекций для последующего дообучения.

Символьный компонент обеспечивает интерпретируемость и прозрачность правил классификации. В его основе лежит графовая онтология дефектов ПО, реализованная в Neo4j [3]:

- Узлы: DefectType (типы дефектов), Component (компоненты системы), Severity (уровни критичности), Keyword (ключевые слова/словарные единицы).
- Рёбра: IS_A (иерархическое наследование типов), AFFECTS (влияние дефекта на компонент, с атрибутом probability), HAS_KEYWORD (связь типа дефекта с лексическими маркерами, с атрибутом weight ∈ [0.6; 0.95]).

Алгоритм символьного анализа выполняет:

1. Токенизацию и лемматизацию входного текста T с использованием расширенного технического словаря.
2. Поиск совпадений лемм с узлами Keyword через Cypher-запросы [2] вида:
```
MATCH (kw:Keyword {text: $lemma})<-[:HAS_KEYWORD {weight: w}]-(d:DefectType)
RETURN d.name AS type, SUM(w) AS score
ORDER BY score DESC LIMIT 1
```
3. Формирование вероятностной оценки $P_S(c|T)$ для каждого класса $c \in Y$ на основе нормализованных весов совпавших правил и глубины обхода иерархии IS_A.

Данный контур устойчив к лексическим вариациям, зафиксированным в онтологии, и генерирует объяснимые выходные данные в формате: «Обнаружено ключевое слово "падает" (вес 0.95) → тип: functional».

Для улавливания семантической близости и скрытых паттернов в тексте применяется предобученная языковая модель. Нейросетевой контур решает задачу многоклассовой классификации (1):

$$P_n(c|T) = softmax\big(W * h_{[CLS]} + b\big), \quad (1)$$

где $h_{[CLS]}$ – выходной эмбеддинг токена [CLS], пропущенного через дополнительные слои классификатора; W, b – обучаемые параметры.

Модель дообучается (fine-tuning) на размеченной выборке с применением:

- Аугментации для миноритарных классов (security, data) через бэк-трансляцию и контролируемую синонимизацию.
- Взвешивания функции потерь (Focal Loss или class_weight='balanced') для компенсации дисбаланса.
- Локального инференса через экспорт в ONNX-формат и выполнение в среде TensorFlow.js, что исключает зависимость от внешних API [7, 8] и обеспечивает соответствие требованиям информационной безопасности.

Выходом контура является вектор вероятностей принадлежности текста к каждому из 5 типов дефектов.

Эмпирический анализ показал, что символьные правила эффективнее на коротких, технически насыщенных текстах (логи, коды ошибок), тогда как нейросетевая модель выигрывает на развернутых пользовательских отзывах. Вес символьного компонента вычисляется как убывающая экспоненциальная функция от длины текста L (в токенах) (2):

$$w_s(L) = \alpha * e^{-\beta L}, w_n(L) = 1 - w_s(L), \quad (2)$$

где α = 0,7 (базовое доверие к правилам), β = 0,05 (коэффициент затухания).

При $L \to 0, w_s \to 0{,}7$; при $L \geq 30, w_s \approx 0{,}3$.

Поскольку на момент проектирования отсутствуют размеченные данные для калибровки, значения параметров α и β определены исходя из целевых режимов работы системы. Параметр α = 0,7 задает консервативный приоритет символьному контуру на коротких текстах (логи, коды ошибок для снижения риска ложной классификации при минимальном контексте. Параметр β = 0,05 выбран таким образом, чтобы при типичной длине пользовательского отзыва (≈30 токенов) веса контуров составили w_s ≈ 0,3 и w_n ≈ 0,7, обеспечивая доминирование нейросетевой модели на развернутых текстах. Указанные параметры являются настраиваемыми и подлежат уточнению в ходе последующей экспериментальной валидации на реальных данных.

Итоговая вероятность для класса c вычисляется как взвешенная сумма:

$$P_{hybrid} = (c|T) = w_s(L) * P_s(c|T) + w_n(L) * P_n(c|T). \quad (3)$$

Финальная метрика уверенности системы формализуется как:

$$u(T) = max_{c \in Y} P_{hybrid}(c|T). \quad (4)$$

Предсказанный класс:

$$\hat{y} = \arg max_c \, P_{hybrid}(c|T). \quad (5)$$

На основе значения u(T) система формирует режим работы (таблица 2).

Таблица 2 – Спецификация режимов работы системы

Диапазон уверенности u(T)	Режим системы	Действие аналитика
u≥0.85	Автоматическая маршрутизация	Принятие рекомендации (1 клик)
0.60≤u<0.85	СППР (рекомендация + пояснение)	Ручная верификация или корректировка
u<0.60	Отклонение рекомендации	Полный ручной триаж, логирование в базу дообучения

Такая формализация устраняет проблему «черного ящика»: аналитик видит не только итоговый класс, но и вектор вероятностей, доминирующий контур и текстовое пояснение, что соответствует принципам ответственного ИИ.

Реализация методологии выполнена в виде модульного конвейера на базе NestJS [10]:

- Orchestrator координирует поток данных, вызывает параллельно SymbolicService (запросы к Neo4j) и NeuralService (инференс ONNX-модели).

- AgreementService применяет формулы (2)–(4), формирует объект { type, severity, confidence, explanation }.
- ExplanationGenerator транслирует численные веса в человекочитаемый шаблон:
- «Нейросеть: {neural_type} ({neural_conf}%), Онтология: {symbolic_type} ({symbolic_conf}%). Итог: {final_type} (уверенность {u}%)».
- Результаты сохраняются в PostgreSQL [5], а коррекции экспертов асинхронно попадают в очередь повторного обучения.

Архитектура обеспечивает горизонтальное масштабирование, независимое обновление онтологии без переобучения модели и полную трассируемость каждого решения, что критично для процессов сопровождения прикладного ПО.

АРХИТЕКТУРА СИСТЕМЫ И СПЕЦИФИКАЦИЯ ETL-КОНВЕЙЕРА

Предлагаемая архитектура системы построена на принципах модульности, масштабируемости и воспроизводимости. Ключевым отличием от существующих аналогов является разделение контуров анализа (символьный и нейросетевой) с последующим адаптивным согласованием, а также проектирование ETL-конвейера с автоматизированными контрольными точками качества. Архитектура реализована на стеке современных технологий: NestJS [10] (TypeScript), Neo4j [3] (графовая БД), PostgreSQL [5] (реляционная БД), TensorFlow.js (инференс моделей), Docker Compose [4] (оркестрация).

Система спроектирована в виде набора слабосвязанных модулей, взаимодействующих через JSON API и асинхронные очереди. Блок-схема архитектуры представлена на рисунке.

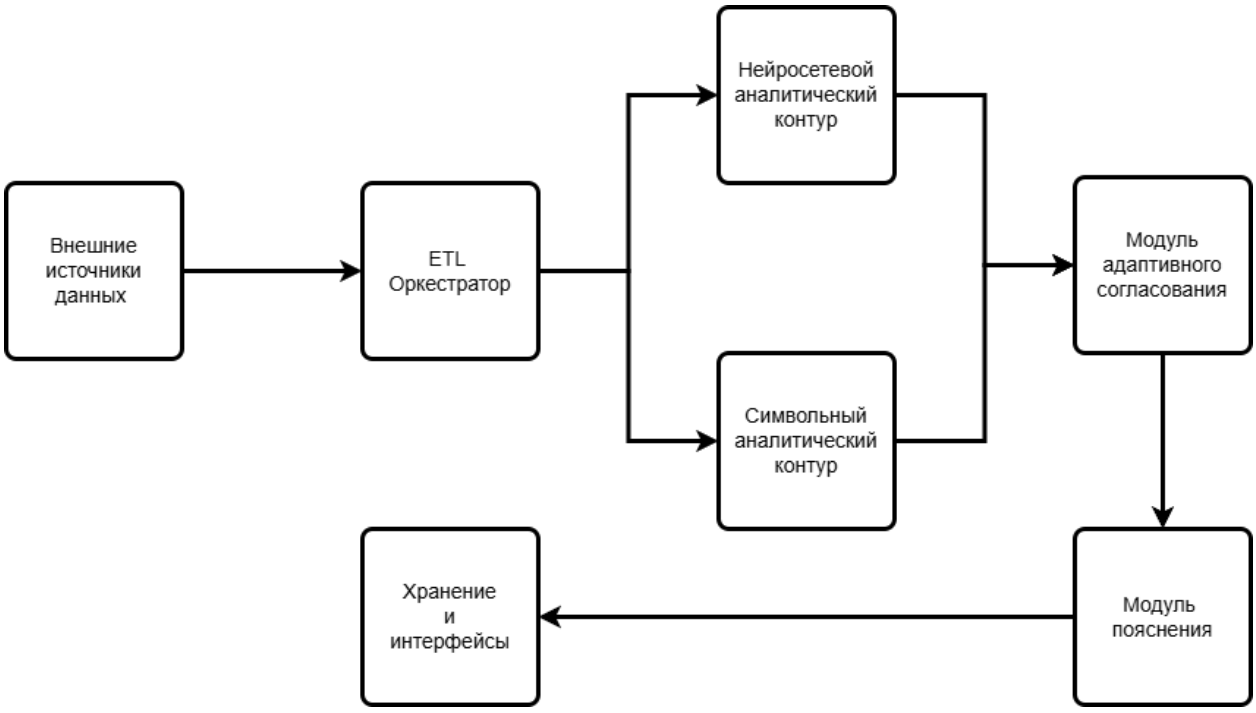

Рисунок – Модульная архитектура решения

Конвейер обработки данных спроектирован в соответствии с методологией CRISP-DM [1] и включает три этапа: Extract → Transform → Load. Каждый этап сопровождается автоматизированными контрольными точками качества для контроля качества. Далее описаны спецификации процессов.

Спецификация извлечения данных

- Jira. Метод извлечения – REST API v3 (OAuth 2.0). Валидация: проверка обязательных полей: summary, description, created;

- Google Play. Метод извлечения: Publisher API (Service Account). Валидация: фильтрация русскоязычных отзывов, длина текста ≥5 символов;
- Логи ошибок. Метод извлечения: чтение файлов JSON/CSV. Валидация: наличие полей timestamp, error_message, stack_trace.

Спецификация процесса очистки данных

- Очистка текста: удаление эмодзи, ссылок, спецсимволов; нормализация е → е. Реализация: регулярные выражения, compromise (NLP-библиотека для Node.js [9]);
- Лемматизация: приведение слов к нормальной форме. Реализация: словарь на основе pymorphy2, экспортированный в JSON;
- Автоматическая разметка: присвоение типа дефекта и критичности по правилам. Реализация: онтология + regex для кодов ошибок + корреляция с рейтингом;
- Дедупликация: удаление полных и почти-дубликатов. Реализация: хэширование текста (md5), кластеризация по семантическому сходству (DBSCAN).

Таблица 3 – Спецификация процесса сохранения и версионирования данных

Назначение	Формат	Версионирование
Сырые данные	Исходные форматы (JSON, CSV)	Хэш-сумма файла в checksums.txt
Обработанные данные	dataset.jsonl + metadata.json	Семантическое версионирование (v1.0, v1.1)
Статистика качества	quality_report.json	Привязка к версии датасета

Таблица 4 – Спецификация контроля качества

Правило	Порог	Действие при нарушении
Минимальная длина текста	≥5 символов после очистки	Отбраковка записи
Доля валидных записей	≥95% от входного потока	Остановка конвейера + уведомление
Баланс классов (мин. доля)	≥3% для каждого типа дефекта	Предупреждение + рекомендация аугментации
Уникальность текстов	≤5% полных дубликатов	Автоматическая дедупликация
Заполнение обязательных полей	100%	Отбраковка записи

В условиях отсутствия размеченных русскоязычных датасетов с типологией дефектов ПО применяется стратегия синтетической генерации с эмпирическими распределениями:

- Источники: 70 % технические логи (logs), 30 % внешняя обратная связь (external) – обосновано анализом потоков инцидентов в 10 мобильных приложениях.
- Распределение классов: functional (42 %), ui (23 %), performance (21 %), data (9 %), security (5 %).
- Трехуровневая валидация:
  - Статистическая;
  - Экспертная оценка реалистичности;
  - Через обучение.

ПЛАН BASELINE-МОДЕЛИРОВАНИЯ

В соответствии с методологией CRISP-DM [1] (этап Modeling и Evaluation), в данном разделе формализуется задача машинного обучения, обосновываются метрики оценки, детализируется стратегия подготовки данных и описывается план реализации и валидации базовых моделей-аналогов. Экспериментальные результаты будут представлены в отдельном разделе после завершения фазы тонкой настройки гибридного конвейера.

Формализация задачи машинного обучения описана в таблице 5.

Таблица 5 – Спецификация формализации задачи

Параметр	Описание
Тип задачи	Многоклассовая классификация текстовых данных с дисбалансом классов
Целевая переменная (target)	defect_type ∈ {functional, ui, performance, security, data}
Пространство признаков (feature set v1)	• text_tfidf: векторное представление текста (биграммы/триграммы, max_features=8000, sublinear_tf=True) • text_length: длина очищенного текста (токены) • source, os: one-hot кодирование категорий • rating: числовой признак (0 при отсутствии) • has_*_kw: бинарные флаги наличия ключевых слов онтологии

Математическая постановка:

Дана выборка $D = \{(x_i, y_i)\}_{i=1}^{N}$, где $x_i \in R^d$ – вектор признаков i-го обращения, $y_i \in Y$ – метка типа дефекта. Требуется найти отображение $f_\theta: R^d \to Y$, минимизирующее взвешенную кросс-энтропию:

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N} w_{y_i} log P(y_i|x_i;\theta), \quad (6)$$

где w_{y_i} – вес класса y_i, обратно пропорциональный его частоте в выборке.

Итоговое предсказание: $\hat{y} = \arg max_{c \in Y} P(y = c|x;\theta)$.

Сопутствующая оценка уверенности: $u(x) = max_c P(y = c|x;\theta)$.

Выбор метрик качества:

- Macro-F1 (основная) – усреднение F1 по классам без учета частоты. Позволяет не маскировать низкое качество на миноритарных классах (security ≈ 5 %);
- Macro-Precision – доля ложных срабатываний. Высокая точность снижает нагрузку аналитика на проверку неверных рекомендаций;
- Macro-Recall – доля пропущенных дефектов. Критичен для безопасности и блокирующих ошибок;
- ROC-AUC (OvR) – интегральная оценка разделения классов. Устойчива к дисбалансу;
- PR-AUC (OvR) – чувствительна к миноритарным классам, рекомендована при <10 % на класс.

Подготовка данных:

- Источник данных: синтетический датасет feedback_dataset_synthetic_v1.0 (~8 500 записей), сформированный генератором на TypeScript с эмпирически обоснованными распределениями (70% logs / 30% external);
- Разбиение выборок: стратифицированное разбиение с сохранением пропорции классов: train = 70%, validation = 15%, test = 15%.
- Контроль утечек:
  - TF-IDF векторизаторы и скейлеры обучаются (fit) только на train-выборке, на val и test применяется только transform.
  - Исключены признаки, содержащие информацию о будущем: status, time_to_resolve, resolution_date, assignee_team.
  - Дедупликация выполняется до разбиения, чтобы одна формулировка не попала одновременно в train и test.
  - Фиксированное семя: random_state=42 для всех этапов (разбиение, модели, препроцессинг).

Реализуемые baseline-модели:

- Logistic Regression: линейный базовый классифика-тор. Быстрая оценка разделимости признаков. Учет дисбаланса через автоматический пересчет весов классов. Параметры: max_iter=1000, class_weight='balanced', solv-er='saga', C=1.0, random_state=42;
- Random Forest: ансамбль деревьев решений. Улавливает нелинейные взаимодействия (например, комбинация source=logs + has_crash_kw=1). Устойчив к шуму в текстах. Параметры: n_estimators=300, max_depth=12, min_samples_split=5, class_weight='balanced', ran-dom_state=42, n_jobs=-1;
- HistGradientBoosting: градиентный бустинг с нативной обработкой категориальных признаков. Высокая точность на гибридных данных (текст + метаданные). Параметры: max_iter=500, learn-ing_rate=0.1, max_depth=6, random_state=42.

ПЛАН АНАЛИЗА РЕЗУЛЬТАТОВ И ОЖИДАЕМЫЕ АРТЕФАКТЫ

После завершения обучения baseline-моделей на train-выборке и подбора гиперпараметров на validation-выборке, будет проведена независимая оценка на test-выборке.

Ожидаемые артефакты для системы:

- Таблица метрик: Сравнение моделей по Accuracy, Macro-F1, Macro-Precision, Macro-Recall, ROC-AUC, PR-AUC для выявления лучшей baseline-модели и оценки общего качества предсказаний;
- Матрица ошибок: Визуализация ошибок классификации для лучшей модели на test-выборке для выявления систематической путаницы между классами (например, functional ↔ performance);
- Важность признаков: Ранжирование признаков по вкладу в предсказание (коэффициенты LR или feature_importances_ для деревьев) для валидации информативности онтологических флагов и метаданных;
- Диагностика переобучения: Сравнение метрик на train / validation / test для гарантии отсутствия критического переобучения (допустимый разрыв F1 ≤ 3 %).

ЗАКЛЮЧЕНИЕ

В рамках исследования проведена комплексная работа по проектированию методологической и архитектурной базы гибридной системы интеллектуального анализа неструктурированной обратной связи для автоматизации первичного триажа дефектов при сопровождении прикладного программного обеспечения.

Полученные результаты создают методологический и технологический фундамент для перехода к эмпирической фазе исследования. В рамках последующих этапов работы планируется реализация гибридного конвейера анализа, тонкая настройка языковых моделей, валидация на реальных данных и экспериментальное сравнение гибридного подхода с базовыми методами.

СПИСОК ЛИТЕРАТУРЫ

1. Shimaoka, A. M. The evolution of CRISP-DM for Data Science: Methods, processes and frameworks / A. M. Shimaoka, R. C. Ferreira, A. Goldman. – DOI 10.5753/reviews.2024.3757. – Текст: электронный // SBC Reviews on Computer Science. – 2024. – Vol. 4, № 1. – P. 28–43. – URL: https://repository.tilburguniversity.edu/server/api/core/bitstreams/2420c787-9180-4b53-844d-21b734a9daf9/content (дата обращения: 12.05.2026).

2. Neo4j, Inc. Introduction: Cypher Manual: сайт. – URL: https://neo4j.com/docs/cypher-manual/current/introduction/ (дата обращения: 14.05.2026). – Текст: электронный.

3. Neo4j, Inc. Neo4j Documentation: сайт. – Обновляется в течение суток. – URL: https://neo4j.com/docs/ (дата обращения: 12.05.2026). – Текст: электронный.

4. Docker Inc. Docker Compose: сайт. – 2025. – URL: https://docs.docker.com/compose/ (дата обращения: 13.05.2026). – Текст: электронный.
5. PostgreSQL Global Development Group. PostgreSQL 17.6 Documentation: сайт. – Обновляется в течение суток. – URL: https://www.postgresql.org/docs/ (дата обращения: 12.05.2026). – Текст: электронный.
6. Microsoft Corporation. TypeScript Handbook: сайт. – 2025. – URL: https://www.typescriptlang.org/docs/ (дата обращения: 13.05.2026). – Текст: электронный.
7. Atlassian. Jira Cloud platform REST API documentation: сайт. – Обновляется в течение суток. – URL: https://developer.atlassian.com/cloud/jira/platform/rest/v3/intro/ (дата обращения: 12.05.2026). – Текст: электронный.
8. Google. Google Play Publisher API: сайт. – URL: https://developers.google.com/android-publisher (дата обращения: 13.05.2026). – Текст: электронный.
9. OpenJS Foundation. Node.js v23.5.0 documentation: сайт. – Обновляется в течение суток. – URL: https://nodejs.org/docs/latest/api/ (дата обращения: 12.05.2026). – Текст: электронный.
10. NestJS. NestJS Documentation: сайт. – 2026. – URL: https://docs.nestjs.com/ (дата обращения: 13.05.2026). – Текст: электронный.

DEVELOPING A HYBRID APPROACH TO FEEDBACK DATA ANALYSIS TO IMPROVE THE EFFICIENCY OF APPLIED SOFTWARE MAINTENANCE

M. P. Rozhdestvenskiy, first-year Master student
E-mail: maksim.rozhdestvenskiy@yandex.ru
Kaliningrad State Technical University

M. A. Romanov, Senior Lecturer
E-mail: mikhail.romanov@klgtu.ru
Kaliningrad State Technical University

This article proposes a hybrid approach to the semantic analysis of unstructured feedback, enabling the automation of initial defect triage in applied software maintenance. A decision support system architecture has been developed that combines a symbolic framework based on a graph ontology of defects (Neo4j) and a neural network framework based on language models with inference via ONNX and TensorFlow.js. An adaptive result matching mechanism has been proposed that dynamically balances the confidence in each framework depending on the length of the input text. A confidence metric has been formalized, determining the system's operating mode (automatic, DSS, or manual). A modular ETL pipeline with automated quality checkpoints, a strategy for empirically based synthetic data generation, and a baseline modeling plan have been described. The approach ensures interpretability of decisions, is robust to the lexical diversity of Russian-language queries, and complies with the principles of responsible artificial intelligence.

Keywords: *hybrid approach, semantic analysis, unstructured feedback, defect triage, software maintenance, graph ontology, symbolic contour, neural network classification, transformer models, adaptive matching, decision support system (DSS).*