ISSN 2713-1734

Научная статья
УДК 004.89:378.146
EDN IBXHXG
DOI 10.17150/2713-1734.2026.8(3).342-355

А.В. Родионов
Байкальский государственной университет,
г. Иркутск, Российская Федерация

И.А. Дорощенко
Байкальский государственной университет,
г. Иркутск, Российская Федерация

Модель и архитектура системы автоматизации оценки заданий с использованием больших языковых моделей

Ааннотация. В работе представлена формальная модель и архитектура информационной системы для автоматизации оценки студенческих работ, развёртываемой в локальной инфраструктуре образовательного учреждения. В качестве демонстрационной предметной области выбраны курсы программирования, для которых характерно сочетание формализуемых и неформализуемых критериев оценивания. Система сочетает детерминированные проверки (модульное тестирование, статический анализ, запуск в изолированной среде) с генерацией развёрнутой обратной связи на основе большой языковой модели. Ключевым элементом является принцип фактологически обоснованной обратной связи: каждое оценочное суждение модели привязано к проверяемому основанию — результатам тестов, диагностикам анализатора, фрагментам кода. Конвейер оценивания формализован как управляемая система с обратной связью, в контур управления которой включён преподаватель. Предложен алгоритм маршрутизации решений с адаптивным порогом уверенности, обеспечивающий автоматическое направление неоднозначных случаев на ручную проверку. Архитектура построена по микросервисной модели с плагинной системой расширения на другие предметные области.

Ключевые слова. Автоматизированное оценивание, большие языковые модели, системный анализ, конвейерная архитектура, многокритериальное оценивание, маршрутизация решений.

Информация о статье. Дата поступления: 19 марта 2026 г.; дата принятия к публикации: 13 октября 2026 г.; дата онлайн-размещения: 11 ноября 2026 г.

Для цитирования. Родионов А.В. Модель и архитектура системы автоматизации оценки заданий с использованием больших языковых моделей / А.В. Родионов, И.А. Дорощенко. — DOI 10.17150/2713-1734.2026.8(3).342-355. — EDN IBXHXG // System Analysis & Mathematical Modeling. — 2026. — Т. 8, № 3. — С. 342–355.

© Родионов А.В., Дорощенко И.А., 2026

Original article

A.V. Rodionov
Baikal State University,
Irkutsk, Russian Federation

I.A. Doroshchenko
Baikal State University,
Irkutsk, Russian Federation

Model and Architecture of an Automated Assessment System Using Large Language Models

Abstract. This paper presents a formal model and architecture of an information system designed to automate the assessment of students' works, deployed within the local infrastructure of an educational institution. The proposed approach is domain-independent; programming courses are chosen as a demonstrative application domain due to their characteristic combination of formalizable and non-formalizable assessment criteria. The system combines deterministic evaluation methods (unit testing, static analysis, and execution in an isolated environment) with the generation of detailed feedback based on a large language model. A key element is the principle of fact-based feedback: each evaluative statement produced by the model is linked to verifiable evidence, including test results, analyzer diagnostics, and code fragments. The assessment pipeline is formalized as a feedback-controlled system incorporating an instructor into the control loop. A decision-routing algorithm with an adaptive confidence threshold is proposed, ensuring the automatic routing of ambiguous cases for manual review. The architecture is based on a microservices model featuring a plugin system for extension into other subject domains.

Keywords. Automated assessment, large language models, systems analysis, pipeline architecture, multi-criteria assessment, solution routing.

Article info. Received 19 March, 2026; Accepted 10 October, 2026; Available online 11 November, 2026.

For Citation. Rodionov A.V., Doroshchenko I.A. Model and Architecture of an Automated Assessment System Using Large Language Models. *System Analysis & Mathematical Modeling*, 2026, vol. 8, no. 3, pp. 342–355. (In Russian). EDN: IBXHXG. DOI: 10.17150/2713-1734.2026.8(3).342-355.

Введение

Автоматизация оценивания студенческих работ является одной из ключевых задач цифровой трансформации высшего образования. Рост числа обучающихся, в том числе в дистанционном формате, приводит к увеличению нагрузки на преподавателей, связанной с проверкой и предоставлением обратной связи. Проблема затрагивает широкий спектр учебных дисциплин — от гуманитарных (оценка эссе, письменных работ) до технических (проверка программного кода, расчётных заданий). В настоящей работе в качестве предметной области выбраны курсы программирования, которые представляют особый интерес благодаря сочетанию объективно проверяемых критериев (корректность выполнения, прохождение тестов) и критериев, требующих экспертной интерпретации (качество кода, читаемость, оригинальность решения).

В контексте курсов программирования проверка и оценка студенческих работ представляет собой трудоёмкую задачу, нагрузка которой растёт линейно с числом обучающихся. В больших группах объём ручной проверки приводит к задержкам с предоставлением обратной связи [1; 2], тогда как своевременная и подробная обратная связь является одним из ключевых факторов эффективного обучения. По имеющимся оценкам, использование средств на основе искусственного интеллекта для подготовки отзывов позволяет существенно сократить время на проверку [3].

Автоматизированные средства проверки программных работ (автогрейдеры) развиваются с 1960-х гг. и получили широкое распространение в массовых онлайн-курсах [4–7]. Они обеспечивают мгновенную проверку функциональной корректности путём запуска кода в изолированной среде (песочнице) и сопоставления результатов с эталонными. Однако такие системы ограничиваются объективными критериями и не способны сформировать развёрнутые комментарии о качестве кода, стиле программирования или допущенных ошибках [2].

Появление больших языковых моделей (англ. LLM — Large Language Model) [8; 9] открыло возможности для генерации содержательных комментариев к студенческим работам. Исследования показывают, что при использовании детализированных рубрик LLM способны оценивать решения не менее строго и последовательно, чем эксперты [10]. Вместе с тем прямое применение LLM сопряжено с рисками: нестабильность оценок, галлюцинации модели, отсутствие привязки к объективным данным [11].

Цель работы — разработать архитектуру и формальную модель информационной системы, объединяющей преимущества детерминированного тестирования и генеративных возможностей LLM для автоматизации оценивания студенческих работ. Научная новизна работы заключается в следующем:

– формализован конвейер оценивания как управляемая система с обратной связью, в контур управления которой включён преподаватель;

– предложен принцип фактологически обоснованной обратной связи, ограничивающий генерацию LLM проверяемыми основаниями;

– разработан алгоритм маршрутизации решений с адаптивным порогом уверенности.

Обзор литературы

Классические системы автоматического оценивания основаны на запуске студенческого кода в песочнице и сравнении результата с ожидаемым выходом или проверке прохождения набора тестов [4]. Fox и соавторы описали систему MAGIC (Massive

Automated Grading in the Cloud), которая за три года автоматически проверила сотни тысяч программных решений, обеспечивая мгновенную обратную связь и возможность повторных попыток [4]. Архитектура подобных систем эволюционировала в сторону модульности и расширяемости.

Однако тестовое покрытие не всегда является полным: студенты могут эксплуатировать неполноту тестов, подгоняя решения под известные проверяемые случаи [2]. Альтернативные методы на основе формальной семантики программ позволяют находить семантические отклонения и генерировать контрпримеры, но требуют сложного инструментария и не получили широкого распространения. Для оценки нефункциональных аспектов применяются статические анализаторы (линтеры) и системы обнаружения заимствований, такие как MOSS [1].

Gobrecht и соавторы [12] исследовали возможность полной замены ручного оценивания системой на основе ИИ, продемонстрировав снижение субъективности. Pathak и соавторы предложили подход «Rubric is All You Need», заключающийся в предоставлении модели подробной, адаптированной под конкретный вопрос рубрики оценивания [10]. Исследование показало, что использование специфичных рубрик позволило LLM значительно точнее оценивать логическую корректность и качество программ. Hong и соавторы в работе RULERS [11] выделили три проблемы «LLM-судей»: нестабильность оценок при незначительных изменениях промпта, отсутствие проверяемых оснований для суждений, несоответствие шкалы оценки человеческим стандартам. Авторы предлагают фиксировать рубрики и требовать от модели явного указания подтверждающих оснований (evidence).

Коммерческая платформа Codio в 2023 году внедрила инструмент LLM Rubric Autograder[1], позволяющий модели генерировать рубрику и оценивать решения. Ряд университетов (Осло[2], ряд немецких вузов [13]) разрабатывают собственные защищённые решения для работы с LLM, мотивируя это требованиями соблюдения законодательства о защите персональных данных (GDPR) и недопустимостью передачи студенческих работ внешним облачным сервисам.

Анализ существующих решений выявляет нишу на пересечении автоматического тестирования и генеративной обратной связи с обеспечением формальных гарантий обоснованности и

[1] The LLM Rubric Auto-grader: Instructor Documentation // Codio. 2023. URL: https://docs.codio.com/instructors/authoring/assessments/llm-based-auto-rubric.html (дата обращения: 02.02.2026).

[2] University of Oslo builds GPT-based chat for higher education in 9 weeks with Azure // Microsoft Customer Stories. 2024. URL: https://www.microsoft.com/en/customers/story/22441-university-of-oslo-azure (дата обращения: 09.02.2026).

контролируемости. Предлагаемая система отличается от аналогов формализацией конвейера оценивания как управляемой системы, строгим принципом фактологической обоснованности, ограничивающим выходы LLM, алгоритмом маршрутизации с адаптивным порогом и локальным развёртыванием.

Постановка задачи

Пусть задано множество студенческих работ $S = \{s_1, s_2, ..., s_n\}$, рубрика (критерии) оценивания $R = \{r_1, r_2, ..., r_k\}$, где каждый критерий r_j определяет шкалу оценок от 0 до w_j (максимальный балл), и набор детерминированных проверок $D = \{d_1, d_2, ..., d_m\}$ (тесты, линтеры, метрики).

Требуется построить отображение F, которая для каждой работы s_i формирует вектор оценок по критериям $a_i = (a_{i1}, ..., a_{ik})$, где $0 <= a_{ij} <= w_j$, структурированный комментарий c_i, каждое утверждение в котором обосновано ссылкой на результат d_l из D или на конкретный фрагмент кода и метрику уверенности $u(s_i)$ из [0, 1], определяющую необходимость ручной проверки.

Критерии рубрики разделяются на две категории. Объективные критерии R_{obj} представляют собой формализуемые условия, однозначно проверяемые программно (прохождение тестов, отсутствие предупреждений компилятора, соответствие стилевым соглашениям). Семантические критерии R_{sem} соответствуют условиям, требующим интерпретации (качество реализации алгоритма, читаемость кода, оригинальность решения).

Для объективных критериев оценка определяется детерминированными проверками и задаётся соотношением

$$a_{ij} = d_{rj}(s_i),\ r_j \in R_{obj},$$

где s_i — оцениваемая работа, r_j — объективный критерий, а d_{rj} — детерминированная функция проверки, ассоциированная с данным критерием.

Для семантических критериев оценка формируется модулем большой языковой модели на основе фактологической базы и определяется выражением

$$a_{ij} = G(s_i, r_j, E(s_i)),\ r_j \in R_{sem},$$

где G — функция генерации оценки и комментария, $E(s_i) = \{d_1(s_i), d_2(s_i), ..., d_m(s_i)\}$ — множество результатов всех детерминированных проверок, формирующих фактологическую базу для работы s_i.

Дополнительно вводятся следующие ограничения:

1. Ограничение обоснованности: для любого утверждения t, содержащегося в комментарии c_i, существует элемент $e \in E(s_i) \cup$

$L(s_i)$ такой, что утверждение t логически следует из e, где $L(s_i)$ — множество фрагментов исходного кода работы s_i.

Ограничение контролируемости: финальный результат оценивания утверждается преподавателем, который принимает решение о подтверждении либо отклонении автоматически сгенерированной оценки.

Ограничение приватности: все данные, используемые в процессе оценивания, обрабатываются и хранятся в пределах контролируемого вычислительного периметра.

Формальная модель конвейера оценивания

Процесс оценивания моделируется как последовательный конвейер с обратной связью и описывается кортежем

$P = \langle S, D, G, C, M \rangle$, где S — пространство входных работ, D — множество детерминированных проверок, G — функция генерации обратной связи на основе языковой модели, C — функция проверки согласованности результатов, M — функция маршрутизации работ между автоматическим и ручным этапами оценивания.

Принципиальным является то, что функция G принимает в качестве аргументов не только работу студента и рубрику, но и результаты детерминированных проверок, формирующие фактологическую базу:

$$G(s_i, R, E(s_i)) \rightarrow (a_i, c_i).$$

Контур обратной связи замыкается через преподавателя — его решения о подтверждении или отклонении оценки используются для адаптации параметров функции маршрутизации M, что обеспечивает управляемость и устойчивость системы оценивания (рис. 1).

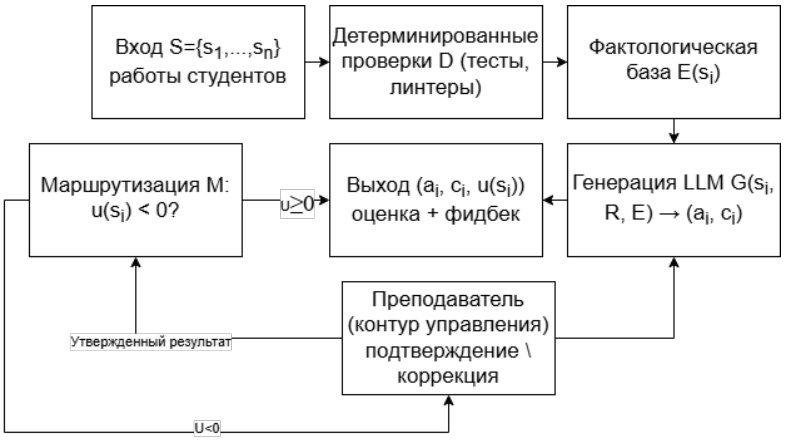


Рис. 1. Формальная модель конвейера оценивания как управляемой системы с обратной связью

Основной принцип предложенной модели заключается в том, что каждое утверждение в генерируемой обратной связи должно быть подкреплено ссылкой на проверяемое основание. Определим политику обоснованности как ограничение:

$$\forall t \in c_i \; \exists e \in E(s_i) \cup L(s_i) : t \vdash e,$$

где t — отдельное утверждение в комментарии c_i, $E(s_i)$ — множество результатов детерминированных проверок, $L(s_i)$ —множество фрагментов исходного кода работы s_i, а символ $\vdash$ обозначает логическое следование утверждения t из элемента e.

На практике это реализуется через шаблон промпта, требующий от LLM цитировать фрагменты кода или сообщения об ошибках, на которых основаны её выводы, а также через последующую верификацию наличия ссылок в выходных данных модели (рис. 2).

Данный принцип близок к методу evidence-anchored scoring, рекомендованному Hong и соавт. [11], и существенно ограничивает пространство возможных выходов LLM. Если модель не может обосновать утверждение, оно исключается из итогового отчёта.

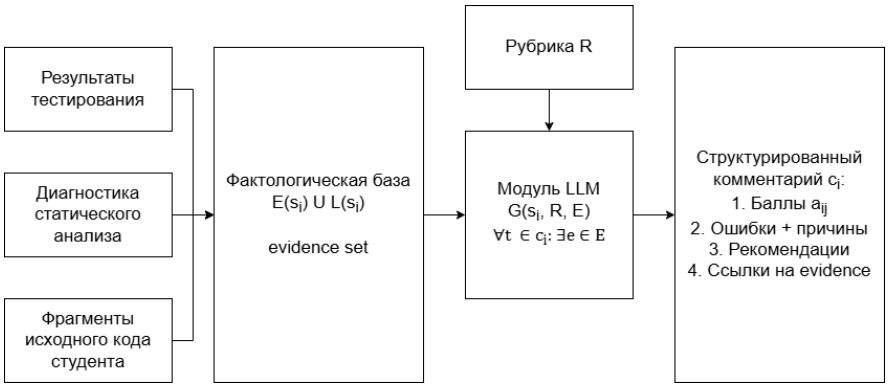


Рис. 2. Принцип фактологически обоснованной обратной связи

Для определения необходимости ручной проверки предлагается алгоритм, основанный на композитной оценке уверенности. Для работы s_i модуль LLM запускается дважды с одними и теми же входными данными, формируя оценки $a^{(1)}$ и $a^{(2)}$. Метрика согласованности определяется как

$$v(s_i) = 1 - \frac{\| a^{(1)} - a^{(2)} \|}{\| w \|},$$

где $w = (w_1, ..., w_k)$ — вектор максимальных баллов. Полнота фактологической базы оценивается как $p(s_i)$ — отношение числа обоснованных утверждений к общему числу утверждений в коммен-

тарии. Доля однозначных детерминированных проверок обозначается $d(s_i)$.

Метрика уверенности вычисляется как взвешенная комбинация:

$$u(s_i) = \alpha v(s_i) + \beta p(s_i) + \gamma d(s_i),$$

где $\alpha + \beta + \gamma = 1$ — настраиваемые весовые коэффициенты. Маршрутизация осуществляется по правилу: если $u(s_i) < \theta$, то работа направляется на ручную проверку; в противном случае результат считается достаточно надёжным для быстрого подтверждения. Порог θ адаптируется по ретроспективным данным:

$$\theta_{t+1} = \theta_t + \eta\,(\text{rate}_{\text{confirm}} - \text{rate}_{\text{reject}}),$$

где η— скорость обучения, $\text{rate}_{\text{confirm}}$ и $\text{rate}_{\text{reject}}$ — доли подтверждённых и отклонённых преподавателем автоматических оценок (рис. 3).

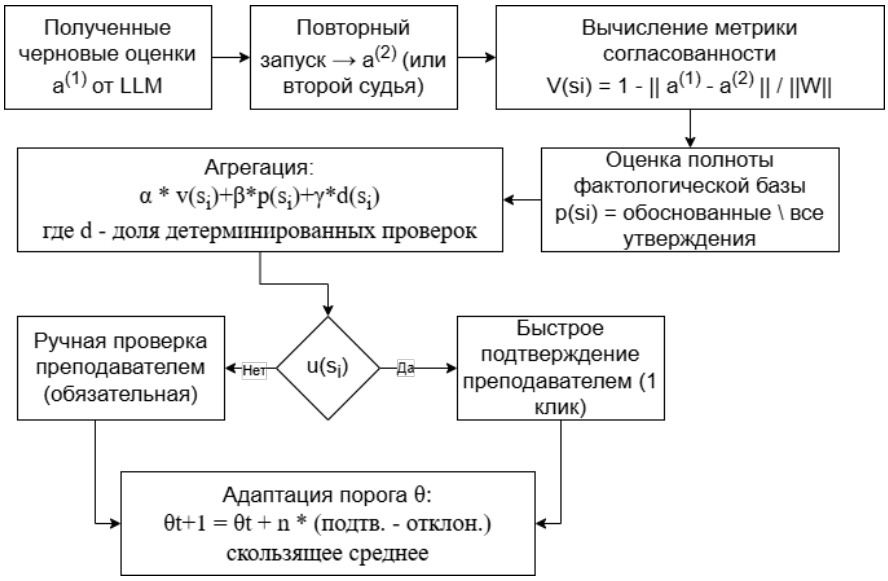


Рис. 3. Алгоритм маршрутизации решений с адаптивным порогом уверенности

Архитектура информационной системы

Система охватывает процесс от отметки посещаемости до сдачи и проверки заданий, предоставления отзывов и аналитики успеваемости. Акторами являются: студент (сдаёт работу, просматривает оценку, отмечает посещение), преподаватель (создаёт задания и рубрики, запускает проверку, подтверждает результаты, просматривает аналитику), администратор (управляет инфраструктурой и пользователями). Диаграмма вариантов использования представлена на рис. 4.

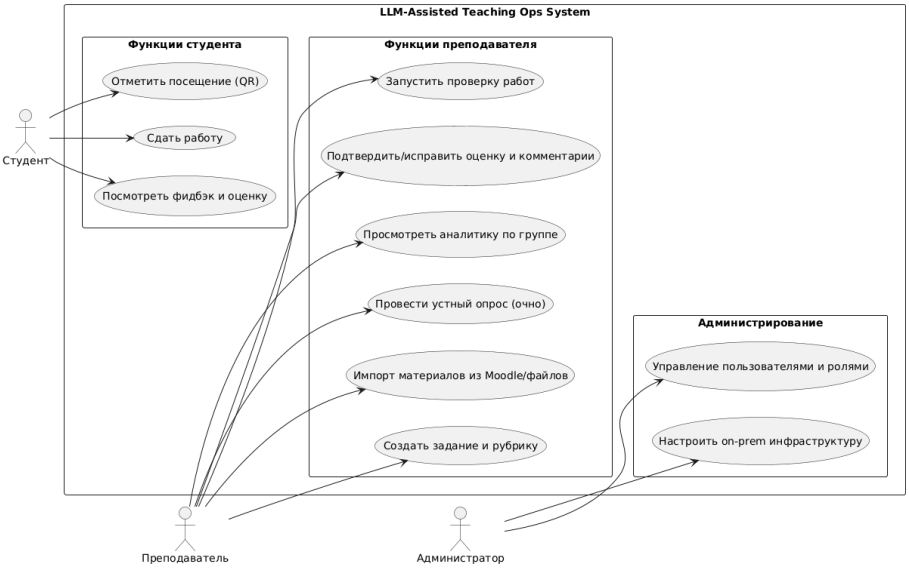


Рис. 4. Диаграмма вариантов использования системы

Система построена по микросервисной модели с единым оркестратором оценивания (рис. 5). Ядро включает: шлюз приложения с модулем аутентификации (API Gateway), сервис курсов и заданий (Course Service), сервис приёма работ (Submission Service), оркестратор оценивания (Assessment Orchestrator), сервис рубрик с поддержкой версионирования, модуль аудита и журналирования, сервис учёта посещаемости (Attendance Service), сервис аналитики (Analytics Service).

Подсистема оценивания программных работ представлена на диаграмме как связка Assessment Orchestrator и Evaluation Pipeline. Конвейер Evaluation Pipeline инкапсулирует все этапы автоматической проверки: изолированный запуск кода, выполнение тестов, статический анализ и генерацию содержательной обратной связи с использованием LLM. Эти механизмы скрыты за единым интерфейсом конвейера и воспринимаются остальной системой как сервис оценки.

Оркестратор выполняет управляющую роль — он инициирует обработку новой работы, последовательно активирует этапы конвейера и агрегирует результаты проверки. При поступлении решения оркестратор:

1. Инициирует выполнение конвейера оценки в изолированной среде.

2. Получает результаты тестирования и анализа кода.

3. Запрашивает генерацию обратной связи на основе рубрики и контекста задания.

4. Проверяет согласованность и полноту полученных данных.
5. Сохраняет итоговые результаты и передаёт их в подсистему хранения и аудита.

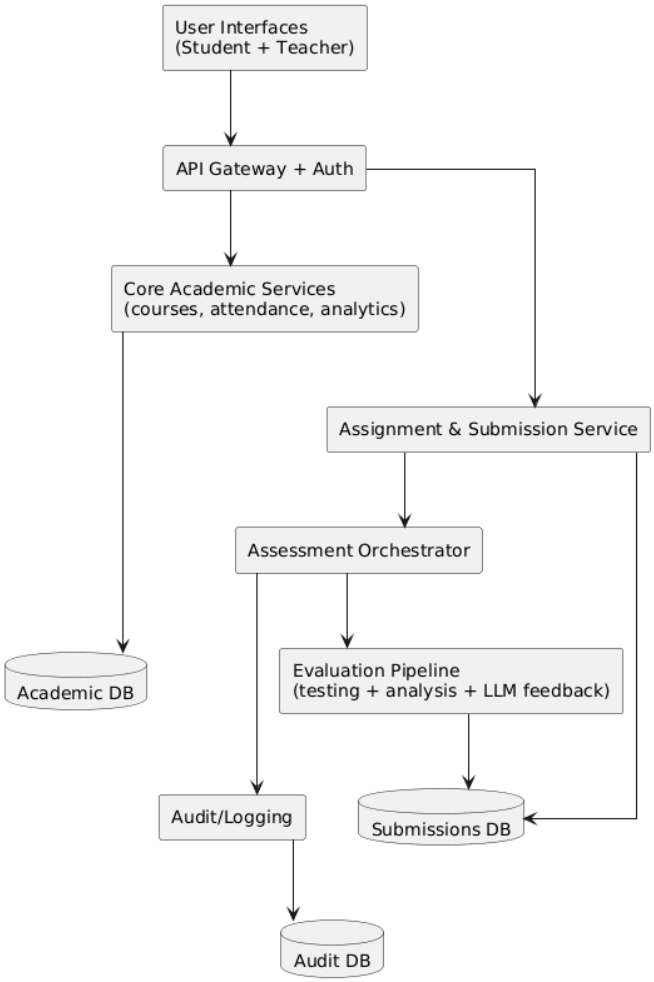


Рис. 5. Компонентная архитектура системы

Последовательность действий при проверке работы представлена на диаграмме последовательности (рис. 6). Студент загружает решение через портал, сервис приёма работ регистрирует попытку и инициирует процесс оценки. Оркестратор распределяет задачу на доступный узел проверки, где выполняются компиляция, тестирование и статический анализ. Затем формируется запрос к модулю LLM, который генерирует черновой набор оценок по критериям и комментариев с отсылкой к конкретным частям кода. Модуль проверки согласованности анализирует стабильность и обоснованность результатов. При низкой

уверенности работа направляется на обязательную ручную проверку; при высокой — преподаватель может утвердить результат одним действием.

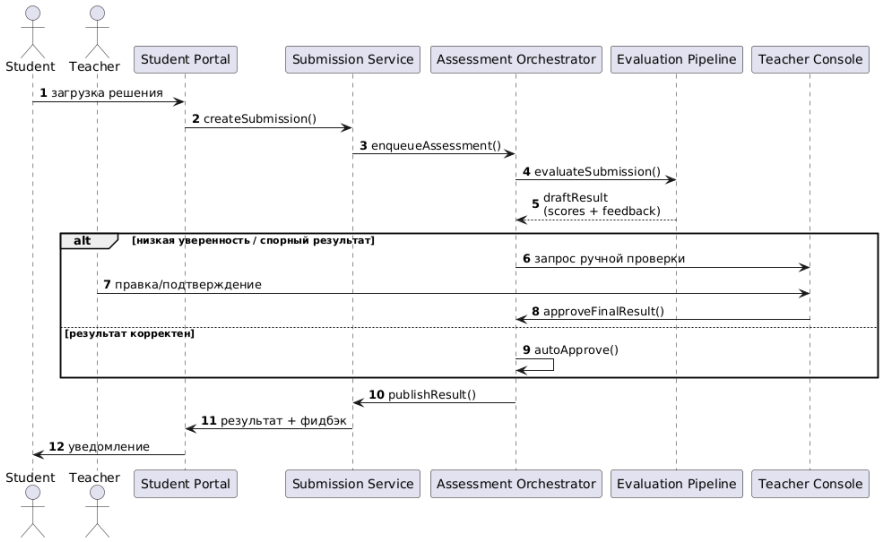


Рис. 6. Диаграмма последовательности проверки программной работы

Система проектируется для развёртывания в локальной инфраструктуре образовательного учреждения, что обеспечивает сохранность персональных данных. Схема развёртывания (рис. 7) включает кластер серверов приложений (API Gateway, основные сервисы, оркестратор), узлы проверки (Evaluation Nodes) с контейнерной изоляцией, узел LLM с GPU, кластер базы данных (PostgreSQL) и объектное хранилище. Обратный прокси-сервер обеспечивает доступ из локальной сети или через VPN.

Расширяемость обеспечивается плагинной архитектурой. Каждый плагин реализует унифицированный интерфейс: получает на вход идентификатор работы и возвращает структурированный результат проверки. Оркестратор не привязан к конкретному типу заданий — он управляет потоком работ согласно конфигурации, определяющей последовательность шагов для каждого типа. Для работ по программированию конфигурация включает следующие этапы: «компиляция -> тестирование -> статический анализ -> генерация обратной связи LLM». Для эссе: «проверка уникальности -> оценка LLM». Для добавления нового типа заданий достаточно реализовать соответствующий плагин, реализующий унифицированный интерфейс оценивания, и зарегистрировать последовательность шагов в конфигурации оркестратора.

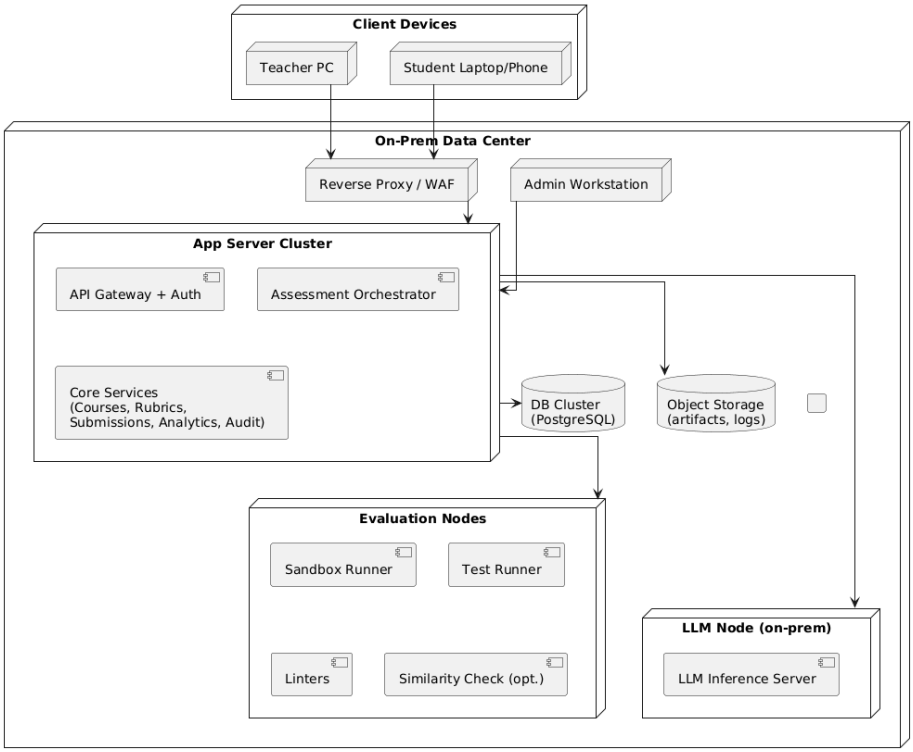


Рис. 7. Схема развёртывания системы в локальной инфраструктуре

В систему интегрирован модуль фиксации посещаемости на основе одноразовых токенов. Преподаватель инициирует формирование QR-кода, содержащего уникальный токен, привязанный к конкретному занятию и временному окну. Студент считывает код мобильным устройством, после чего система фиксирует присутствие. Токен одноразовый и ограничен по времени, что минимизирует возможность передачи третьим лицам. Данные о посещаемости интегрируются в аналитическую панель.

Принцип «человек в контуре управления» реализуется через преподавательскую консоль — веб-интерфейс, в котором преподаватель работает с очередью проверенных работ. Для результатов с высокой уверенностью доступно быстрое подтверждение: преподаватель просматривает сгенерированный отзыв и утверждает его одним действием. Работы с низкой уверенностью помечаются как требующие обязательного внимания и сопровождаются указанием причины (низкая согласованность, неполная фактологическая база, противоречие между результатами тестов и выводами модели).

В режиме ручной верификации преподаватель просматривает все промежуточные данные (отчёты тестирования, диагностики

анализаторов, предложения модели с обоснованиями) и при необходимости корректирует баллы и формулировки. Все действия журналируются, что обеспечивает аудируемость процесса.

Заключение

В работе предложена архитектура и формальная модель информационной системы автоматизированного оценивания программных работ, основанная на разделении детерминированного и вероятностного компонентов. Конвейер оценивания формализован как управляемая система с обратной связью, в контур управления которой включён преподаватель. Введён и формализован принцип фактологически обоснованной обратной связи, ограничивающий генерацию языковой модели проверяемыми основаниями. Предложен алгоритм маршрутизации решений с адаптивным порогом уверенности, обеспечивающий выявление неоднозначных случаев. Совокупность предложенных решений формирует целостный подход к построению систем автоматизированного оценивания, в которых генеративные возможности LLM используются в управляемом режиме с сохранением ведущей роли преподавателя. Дальнейшие исследование предполагают формальную верификацию алгоритма маршрутизации, сравнительный анализ различных языковых моделей, разработка методики калибровки порога уверенности, а также расширение функциональности в части реализации модуля ассистирования при устном опросе.

References

1. Aiken A. MOSS: A System for Detecting Software Similarity. *Stanford University*. Available at: https://theory.stanford.edu/~aiken/moss/.

2. Liu X., Wang S., Wang P., Wu D. Automatic Grading of Programming Assignments: An Approach Based on Formal Semantics. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET). IEEE*, 2019, pp. 126–137. DOI: 10.1109/ICSE-SEET.2019.00022.

3. Winkel D. AI Grading in Higher Ed: Faster, Fairer, and More Helpful. *TeachingEntrepreneurship.org*. 2024. November. Available at: https://teachingentrepreneurship.org/ai-grading-benefits-2/.

4. Fox A., Patterson D.A., Joseph S., McCulloch P. MAGIC: Massive Automated Grading in the Cloud. *Changee/Wapla/HybridEd@EC-TEL 2015: CEUR Workshop Proceedings*, 2015, vol. 1599, pp. 39–48.

5. Ihantola P., Ahoniemi T., Karavirta V., Seppälä O. Review of Recent Systems for Automatic Assessment of Programming Assignments. *Proceedings of the 10th Koli Calling International Conference on Computing Education Research. ACM*, 2010, pp. 86–93. DOI: 10.1145/1930464.1930480.

6. Paiva J.C., Leal J.P., Figueira Á. Automated Assessment in Computer Science Education: A State-of-the-Art Review. *ACM Transactions on Computing Education*, 2022, vol. 22, no. 3, pp. 1–40. DOI: 10.1145/3513140.

7. Ala-Mutka K.M. A Survey of Automated Assessment Approaches for Programming Assignments. *Computer Science Education*, 2005, vol. 15, no. 2, pp. 83–102. DOI: 10.1080/08993400500150747.

8. Brown T.B., Mann B., Ryder N. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, vol. 33, pp. 1877–1901. DOI: 10.48550/arXiv.2005.14165.

9. Touvron H., Lavril T., Izacard G., Martinet X., LachauxM.-A. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint*, 2023. arXiv: 2302.13971. DOI: 10.48550/arXiv.2302.13971.

10. Pathak A., Gandhi R., Uttam V., Ramamoorthy A., Ghosh P. Rubric Is All You Need: Enhancing LLM-based Code Evaluation With Question-Specific Rubrics. *arXiv preprint*, 2025. arXiv: 2503.23989. DOI: 10.48550/arXiv.2503.23989.

11. Hong Y. Rulers: Locked Rubrics and Evidence-Anchored Scoring for Robust LLM Evaluation. *arXiv preprint*, 2026. arXiv:2601.08654. DOI: 10.48550/arXiv.2601.08654.

12. Gobrecht A., Tuma F., Möller M., Zöller T., Zakhvatkin M., A. Wuttig, Sommerfeldt H., Schütt S. Beyond Human Subjectivity and Error: a Novel AI Grading System. *arXiv preprint*. 2024. arXiv:2405.04323. DOI: 10.48550/arXiv.2405.04323.

13. Khakhar J. Privacy and copyright concerns with the utilization of OpenAI's services / J. Khakhar [и др.] // Proceedings of EasyChair. 2024.

Информация об авторах

Родионов Алексей Владимирович — кандидат технических наук, доцент, кафедра математических методов и цифровых технологий, Байкальский государственный университет, 664003, Российская Федерация, г. Иркутск, ул. Ленина, 11, e-mail: RodionovAV@bgu.ru.

Дорощенко Игорь Александрович — аспирант, кафедра математических методов и цифровых технологий, Байкальский государственный университет, 664003, Российская Федерация, г. Иркутск, ул. Ленина, 11, e-mail: 010781@bgu.ru.

Information about the Authors

Alexey V. Rodionov — PhD in Technical Sciences, Associate Professor, Department of Mathematical Methods and Digital Technologies, Baikal State University, 11 Lenin Str., Irkutsk, 664003, Russian Federation, e-mail: RodionovAV@bgu.ru.

Igor A. Doroshchenko — PhD Student, Department of Mathematical Methods and Digital Technologies, Baikal State University, 11 Lenin Str., Irkutsk, 664003, Russian Federation, e-mail: 010781@bgu.ru.

Вклад авторов

Все авторы сделали эквивалентный вклад в подготовку публикации. Авторы заявляют об отсутствии конфликта интересов.

Contribution of the Authors

The authors contributed equally to this article. The authors declare no conflicts of interests.