Таблица 2

Рекомендации по выбору версии для типовых сценариев

Сценарий	Рекомендуемая версия	Основание выбора
Edge-устройства (Raspberry Pi, Jetson Nano)	YOLO11n	Малая модель, ускоренный CPU-инференс, низкое потребление памяти.
Видеонаблюдение 24/7	YOLO11n/s или YOLOv10s	YOLO11 – экономия ресурсов на CPU; YOLOv10 – минимальная задержка при наличии GPU и отсутствии NMS.
Автономное вождение	YOLOv10m или YOLO11m	YOLOv10 – более предсказуемая задержка без NMS; YOLO11m – выше точность при наличии вычислительного бюджета.
Промышленный контроль качества	YOLO11m (и варианты с сегментацией) / YOLOv10m	Нужны высокая точность по мелким объектам и высокая частота кадров; выбор зависит от доступного ускорителя.
Облачный сервис	YOLO11x или YOLOv10x	Приоритет точности; задержка на GPU приемлема, выбирается по нагрузке и стоимости вычислений.

Заключение

Отдельный интерес представляет сравнение не только средних значений, но и распределений латентности, поскольку именно предсказуемость задержки определяет пригодность детектора для реального времени.

Сопоставление YOLOv8–YOLO11 показывает, что эволюция после 2023 года в значительной мере направлена на снижение задержки и стоимости вывода при близких уровнях точности: YOLOv9 усиливает обучение через PGI, YOLOv10 устраняет необходимость NMS и тем самым уменьшает задержку, а YOLO11 сокращает число параметров и ускоряет CPU-инференс за счёт замены базовых блоков и расширения внимания, на практике это позволяет обоснованно выбирать версию под ограничения конкретного стенда (CPU/GPU, память, требуемый FPS) и сокращает риск неоправданных затрат при развёртывании, при этом дальнейшая проверка должна включать доменную переносимость, устойчивость к артефактам видеопотока и эффект квантования на целевом оборудовании.

Список литературы

1. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). P. 779-788. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf (date accessed: 18.12.2025).

2. YOLOv7: Trainable state-of-the-art object detector. arXiv preprint arXiv:2207.02696. URL: https://arxiv.org/abs/2207.02696 (date accessed: 18.12.2025).

3. YOLOv8: A state-of-the-art real-time object detector. Ultralytics GitHub Repository. URL: https://github.com/ultralytics/ultralytics (date accessed: 15.12.2025).

4. YOLOv9: Learning what you want to learn using programmable gradient information. arXiv preprint arXiv:2402.13616. URL: https://arxiv.org/abs/2402.13616 (date accessed: 20.12.2025).

5. YOLOv10: Real-time end-to-end object detection. arXiv preprint arXiv:2405.14458. URL: https://arxiv.org/abs/2405.14458 (date accessed: 18.12.2025).

6. YOLO11: Advanced Real-time Object Detection. Ultralytics Documentation. URL: https://docs.ultralytics.com/ (date accessed: 16.12.2025).

ОБЗОР РАЗВИТИЯ И ТЕХНОЛОГИЧЕСКИХ ВЫЗОВОВ ПЛАТФОРМ ДЛЯ ПРОВЕДЕНИЯ СОРЕВНОВАНИЙ ПО СПОРТИВНОМУ ПРОГРАММИРОВАНИЮ

Креминь А. А., Мозговенко А. А.

ФГБОУ ВО «Мелитопольский государственный университет», Мелитополь, e-mail: alex.kremi@mail.ru, ya@amozgovenko.ru

Постановка проблемы. Рост массовых онлайн-соревнований по спортивному программированию, характеризующихся одновременной подачей тысяч решений, многоязычной поддержкой и разнообразием форматов задач, обнажил архитектурные ограничения существующих платформ автоматизированной проверки. Практика их эксплуатации показывает, что различия в способах изоляции исполнения кода, организации очередей компиляции и запуска, а также в механизмах масштабирования вычислительных ресурсов приводят к нестабильному времени проверки, трудновоспроизводимым результатам и рискам нарушения безопасности. Отсутствие систематизированного анализа архитектурных решений, подтверждённого сопоставлением реальных платформ и их инженерных компромиссов, затрудняет выработку обоснованных рекомендаций по проектированию устойчивых и детерминированных соревновательных систем, что формирует центральную проблему настоящего исследования.

Материалы и методы исследования

Анализ публикаций 2020–2025 годов демонстрирует, что исследовательский и инженерный фокус в области платформ для спортивного программирования сместился от описания базовой функциональности онлайн-проверки к решению прикладных задач масштабируемости, безопасной изоляции исполнения и воспроизводимости

результатов в условиях массовых контестов; это проявляется в появлении работ, систематизирующих типы автоматизированных систем оценки и выделяющих архитектурные узкие места при высоких нагрузках. Параллельно усилилось внимание к практическим инженерным решениям: исследования последних лет подробно рассматривают микросервисную декомпозицию, контейнеризацию и оркестрацию как основные приёмы обеспечения горизонтального масштабирования и быстрой реагируемости подсистем проверки, а также анализируют особенности балансировки очередей заданий и управления ресурсами в распределённых средах. Новое направление, заметно усилившееся в 2023–2025 годах, связано с оценкой семантической корректности кода и попытками использовать модели и инструменты на базе искусственного интеллекта для улучшения качества проверки – как дополнение к традиционным тест-наборным подходам, так и для создания гибридных фреймворков оценки, что ставит свои требования к архитектуре платформ и к процедурам валидации оценок. Практические отчёты и блоги команд крупнейших платформ подтверждают ускоренную эволюцию интерфейсов и инфраструктуры: разработчики отмечают рост объёмов трафика, расширение набора поддерживаемых языков и постоянные улучшения подсистем компиляции/изолированного исполнения в ответ на изменяющиеся потребности сообщества. Наконец, в ряде научных публикаций и сборников появляются систематизированные описания требований к online-judge-системам и их архитектурам, предлагающие формальные модели компонентов и критерии оценки проектных компромиссов, что создаёт основу для сопоставительного анализа и формализации рекомендаций по проектированию устойчивых соревновательных платформ.

Целью настоящего исследования является сравнительно-аналитическое осмысление архитектурных решений современных платформ для проведения соревнований по спортивному программированию с выявлением ключевых тенденций их развития и системных технологических вызовов, возникающих при масштабировании и усложнении соревновательной инфраструктуры.

Результаты исследования и их обсуждение

Анализ предметной области.

В качестве объектов анализа рассматривались типовые архитектурные компоненты таких систем, включая подсистемы приёма и хранения пользовательских решений, механизмы компиляции и исполнения программного кода, средства автоматизированной проверки результатов, а также сервисы управления соревнованиями и пользовательскими сессиями.

При отборе материалов учитывались платформы, обладающие устойчивой практикой эксплуатации, поддерживающие массовое участие и различные языки программирования, что позволило сосредоточиться на архитектурных решениях, доказавших свою работоспособность в условиях реальных соревновательных нагрузок.

В рамках исследования применялся структурно-функциональный анализ, позволивший сопоставить архитектурные элементы платформ с их функциональным назначением в соревновательном процессе. Такой подход дал возможность проследить, каким образом технические решения отражаются на ключевых характеристиках системы, включая пропускную способность, время отклика и детерминированность результатов проверки решений.

Анализ проводился с учётом типовых сценариев эксплуатации, характерных для соревнований различного уровня, от локальных учебных контестов до международных турниров с одновременным участием большого числа команд.

Для выявления эволюционных тенденций в архитектуре платформ использовался историко-технологический метод, основанный на сопоставлении ранних и современных архитектурных подходов. Рассматривались переходы от монолитных систем к модульным и распределённым архитектурам, а также изменение роли вычислительной инфраструктуры в процессе проверки решений. Это позволило проследить, каким образом рост вычислительных требований и усложнение форматов соревнований влияли на выбор архитектурных решений и стимулировали внедрение новых технологических подходов, таких как изоляция исполнения пользовательского кода и динамическое масштабирование ресурсов.

Дополнительно в исследовании применялся метод качественного анализа инженерных ограничений, возникающих при проектировании и эксплуатации платформ для спортивного программирования. Рассматривались типовые технологические вызовы, связанные с безопасностью исполнения недоверенного кода, обеспечением воспроизводимости результатов и поддержкой гетерогенных вычислительных сред. Анализ данных аспектов проводился без привязки к конкретным реализациям, что позволило обобщить результаты и сформулировать выводы, применимые к широкому классу соревновательных платформ.

В результате сравнительного анализа архитектурных решений современных платформ для проведения соревнований по спортивному программированию были рассмотрены ключевые представители данного класса цифровых систем, обладающие разной историей развития, функциональной направленностью и масштабом эксплуатации.

При выборе конкретных примеров в качестве предметов анализа ориентировались на платформы, которые активно используются сообществом программистов для проведения регулярных соревнований, обладают устойчивой инфраструктурой и представлены значительным числом участников. К числу таких платформ относятся Codeforces, CodeChef и HackerRank, каждая из которых иллюстрирует характерные архитектурные подходы и технологические вызовы, присущие современным соревновательным системам.

Codeforces – российская платформа для проведения онлайн-соревнований по спортивному программированию, созданной и поддерживаемой группой программистов во главе с Михаилом Мирзаяновым (ИТМО), широкая аудиторию в более чем 600,000 зарегистрированных пользователей. Она функционирует как самостоятельная цифровая экосистема, где регулярно проводятся контесты, начисляются рейтинги участников и обеспечивается автоматическая проверка решений. Платформа является одной из наиболее известных и крупных в мире соревновательного программирования, с миллионами зарегистрированных пользователей и активным сообществом программистов.

Анализ платформы позволяет выделить её как одну из наиболее массовых и технически зрелых систем в области соревновательного программирования. Платформа возникла как независимый проект и развивалась в направлении поддержки регулярных алгоритмических контестов с глобальной рейтинговой системой, что потребовало проектирования архитектуры, способной обрабатывать десятки тысяч одновременных запросов от участников в период проведения раундов. Codeforces реализует сложный механизм автоматической проверки решений, включающий приём исходного кода, компиляцию, исполнение в изолированной среде и сравнение результатов с тестовыми сценариями, что требует балансировки между скоростью выполнения и безопасностью, особенно в условиях мощных нагрузок во время пиковых периодов соревнований. Кроме того, наличие функционала для публикации задач, редакционных материалов и блогов участников создаёт дополнительные требования к интеграции компонентов управления контентом и соревновательной логики, что усложняет архитектуру системы и делает её мультисервисной по своей природе.

В противоположность Codeforces архитектура CodeChef отражает другой модельный подход, ориентированный на разнообразие форматов соревнований и длительность взаимодействия с пользователем. Платформа проводит как краткие контесты, так и более длительные соревнования, что формирует нагрузку на инфраструктуру в более разрозненные временные окна, но с большим объёмом задач и участников, стремящихся к разным типам участия.

Такое распределение нагрузок диктует необходимость гибкого механизма масштабирования ресурсов, в котором подсистемы проверки решений, очередей исполнения и базы данных должны динамически адаптироваться к изменяющемуся числу активных участников. Сама архитектура CodeChef ориентирована на обеспечение устойчивой производительности при вариативных условиях эксплуатации, что требует разработки продвинутых механизмов кеширования, оптимизации ввода-вывода и управления вычислительными ресурсами для минимизации времени отклика и предотвращения деградации качества обслуживания.

Платформа HackerRank представляет собой ещё один пример технологически ориентированной системы, но с более широким контекстом применения: помимо классических соревнований по алгоритмическому программированию, она активно используется для оценки профессиональных навыков разработчиков в контексте рекрутинга и технических собеседований.

Архитектурные решения HackerRank включают многоуровневую систему тестирования, охватывающую не только алгоритмы, но и практические задания на разнообразных языках программирования, что требует от инфраструктуры поддержки обширного набора компиляторов и сред выполнения. Такая мультиаспектная проверка решений создаёт дополнительные технологические вызовы, связанные с безопасным исполнением кода, детерминированностью результатов и консистентностью оценок при различной конфигурации вычислительных сред.

Подобные требования накладывают на архитектуру платформы обязательность строгой изоляции процессов, обеспечения репродуцируемости тестов и контроля за средой исполнения, что в совокупности увеличивает сложность реализации и эксплуатацию данных систем.

Выводы

Благодаря анализу архитектурных подходов к созданию платформ для проведения соревнований по спортивному программированию можно сделать ряд обобщённых выводов о текущем состоянии и ключевых технологических вызовах этой области.

Эволюция систем от монолитных реализаций к модульным и распределённым архитектурам продемонстрировала, что центральными свойствами успешной платформы являются способность обеспечивать надёжную изоляцию выполнения пользовательского кода, детерминированность результатов проверки и гибкость в управлении вычислительными ресурсами; именно эти требования определяют выбор инструментов контейнеризации, песочниц, очередей заданий и механизмов масштабирования. Проведенный анализ архитектурных особенностей указанных платформ выявил несколько

общих технологических вызовов, характерных для подобных цифровых экосистем.

Во-первых, это обеспечение изоляции выполнения кода: платформы должны гарантировать, что решения участников не вмешиваются в работу основной системы и не нарушают её целостность, что требует применения контейнеризации, песочниц и строгих политик безопасности.

Во-вторых, необходима детерминированность условий проверки: одинаковый алгоритм должен получить аналогичный результат при любых условиях запуска, что требует стандартизованных сред выполнения и чётко определённых ограничений по времени и памяти.

В-третьих, важнейшим вызовом является масштабируемость архитектуры: системы обязаны обеспечивать надёжную работу при пиковых нагрузках, что предполагает автоматическое распределение нагрузки между вычислительными узлами, оптимизацию очередей заданий и грамотное управление ресурсами [1].

Наконец, интеграция аналитических и социальных функций, таких как рейтинги, форумы и образовательные материалы, требует от архитектуры гибкости и расширяемости, поскольку подобные компоненты должны не только работать автономно, но и поддерживать консистентное взаимодействие с основными модулями системы проверки.

Практика показала, что неправильная декомпозиция системы на сервисы или пренебрежение стандартами среды выполнения приводит к трудноуловимым ошибкам воспроизводимости и к рискам безопасности, тогда как продуманная модульность упрощает эволюцию функционала и снижает стоимость сопровождения.

Масштабируемость остаётся критическим фактором – как в аспекте обработки пиковых нагрузок во время массовых контестов, так и в обеспечении низкой задержки отклика для интерактивных задач – и требует внедрения адаптивных стратегий балансировки, продвинутых политик очередей и мониторинга загрузки. Кроме того, мультиформатность современных соревнований и потребность в поддержке множества языков программирования ставят дополнительные требования к унификации интерфейсов компиляции и исполнения, к стандартизации ограничений по времени и памяти, а также к обеспечению репродуцируемости тестовых сред.

Наконец, интеграция социальных, рейтинговых и образовательных компонентов в соревновательную инфраструктуру предъявляет высокие требования к согласованности данных и гибкости взаимодействия подсистем: платформа должна не только точно оценивать корректность решений, но и поддерживать прозрачную обратную связь, аналитические отчёты и инструменты для обучения участников.

Исходя из этих наблюдений, целесообразно направлять дальнейшие инженерные усилия на развитие модульных архитектур с явной границей ответственности сервисов, стандартизацию сред выполнения и тестовых спецификаций, внедрение механизмов автоматического масштабирования и продвинутых средств наблюдения за поведением системы; параллельно необходимы исследования по формализации критериев детерминированности и воспроизводимости в контексте соревновательной оценки, а также по разработке методик экспериментального сравнения архитектурных решений в реальных нагрузочных сценариях.

Такое сочетание практических улучшений и методологического осмысления будет способствовать созданию более надёжных, безопасных и педагогически ценных платформ, способных удовлетворять растущие требования сообществ программистов и образовательных учреждений.

Список литературы

1. Евстропов Г. О. Системы оценивания в задачах с автоматической проверкой решений // Информационный журнал по олимпиадам и программированию. Информатика и образование. 2016. № 3. С. 65-67.

2. Иртегов Д. В., Нестеренко Т. В., Чурина Т. Г. Системы автоматизированной оценки заданий по программированию: разработка, использование и перспективы // Вестник Новосибирского государственного университета. Серия: Информационные технологии. 2019. Т. 17, № 2. С. 61-73.

3. Корнеев Г. А., Елизаров Р. А. Автоматическое тестирование решений на соревнованиях по программированию // Телекоммуникации и информатизация образования. 2003. № 1. С. 61-73.

4. Материалы VIII Всероссийской с международным участием научно-практической конференции «День спортивной информатики», (Москва, 2 декабря 2024 года) / Ассоциация компьютерных наук в спорте. Москва, 2024. 132 с.

ПУЗЫРЬКОВАЯ СОРТИРОВКА: СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМА НА СОВРЕМЕННЫХ ЯЗЫКАХ ПРОГРАММИРОВАНИЯ В ЧИТАЕМОСТИ КОДА

Кузнецова В. А.

ФГБОУ ВО «Мелитопольский государственный университет», Мелитополь, e-mail: skald.beregovoi@yandex.ru

Научный руководитель: Береговой А. В.

Введение

Пузырьковая сортировка (Bubble Sort) – один из старейших алгоритмов сортировки, известный еще с середины XX века. Его название, впервые введенное в работе Айверсона (Iverson) в 1962 году, точно отражает суть процесса: элементы с наибольшими (или наименьшими) значениями постепенно «всплывают» к своему законному месту в конце (или начале) массива, подобно пузырькам воздуха в воде. [1]

Цель исследования – на основе различных современных языков программирования (C++,