МРНТИ 28.23.15

10.51889/2959-5894.2026.93.1.023

Д. Узак [1*], М.М. Мералиев [2], А. Г. Серек [3], А.К. Хачатрян [4], Д.У. Сексенова [5]

[1]Казақстан-Британ техникалық университеті, Алматы қ., Қазақстан
[2]Сулейман Демирель атындағы университет (SDU), Алматы қ., Қазақстан
[3]Astana IT University, Астана қ., Қазақстан
[4]Ш. Есенов атындағы Каспий мемлекеттік технологиялар және инжиниринг университеті, Ақтау қ., Қазақстан,
[5]Абай атындағы Қазақ ұлттық педагогикалық университеті, Алматы қ., Қазақстан
[*]*e-mail: d_uzak@kbtu.kz*

БАҒДАРЛАМАЛЫ ҚҰРАЛДЫҢ САПАСЫН АРТТЫРУ ҮШІН ЖАСАНДЫ ИНТЕЛЛЕКТ НЕГІЗІНДЕГІ АВТОМАТТАНДЫРЫЛҒАН СЫНАҚ ЖҮЙЕЛЕРІН ӘЗІРЛЕУ ЖӘНЕ ЕНГІЗУ

Аңдатпа

Зерттеу бағдарламалық жасақтаманың сапасын арттыру үшін жасанды интеллект негізіндегі гибридті автоматтандырылған сынақ әдісін ұсынады. Ол BiLSTM жасанды нейрондық желілері негізіндегі ақаулы болжауды, Q-оқыту арқылы сынақтың басымдығын есептеуді және оракулсыз орталарда тексеру үшін метаморфтық сынақты қолданады. Әдіс жоғары дәлдік пен шығындарды үнемдеу тұрғысынан тиімдірек сынақты қамтамасыз етеді. Жүйе компоненттері CI/CD ортасында Docker Swarm арқылы іске қосылуымен бірге Python тілінде PyTorch және OpenAI Gym көмегімен іске асырылды. Эксперименттер қолмен жүргізілетін сынақпен салыстырғанда сынақ уақытын 50% қысқартуды, ақауларды анықтау дәлдігін 35-40% арттыруды және адамның қатысуын 60-70% азайтуды көрсетті. Ұсынылған тәсіл agile-орталарда және CI/CD конвейерлерінде практикалық қолдану мүмкіндігін көрсетеді.

Түйін сөздер: автоматтандырылған сынақ, жасанды интеллект, машиналық оқыту, BiLSTM, Q-оқыту, метаморфтық сынақ, бағдарламалық жасақтама сапасы, DevOps.

Д. Узак[1], М.М. Мералиев[2], А. Г. Серек[3], А.К. Хачатрян[4], Д.У. Сексенова[5]
[1]Казахстанско-Британский технический университет, г. Алматы, Казахстан
[2]Университет имени Сулеймана Демиреля (SDU), г. Алматы, Казахстан
[3]Astana IT University, г. Астана, Казахстан
[4]Каспийский государственный университет технологий и инжиниринга имени Ш. Есенова, г.Актау, Казахстан
[5]Казахский национальный педагогический университет имени Абая, г. Алматы, Казахстан,

РАЗРАБОТКА И ВНЕДРЕНИЕ СИСТЕМ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЕ НА ОСНОВЕ ИИ ДЛЯ ПОВЫШЕНИЯ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Аннотация

В исследовании предлагается гибридный метод автоматизированного тестирования на основе искусственного интеллекта для повышения качества программного обеспечения. Он включает предсказание дефектов на основе нейронных сетей BiLSTM, расчёт приоритета тестов с помощью Q-обучения и метаморфное тестирование для проверки в средах без оракула. Метод обеспечивает более эффективное тестирование с повышенной точностью и экономией затрат. Компоненты системы были реализованы на Python с использованием PyTorch и OpenAI Gym, а развертывание осуществлялось с помощью Docker Swarm в среде CI/CD. Эксперименты показали сокращение времени тестирования на 50%, повышение точности обнаружения дефектов на 35-40% и снижение человеческого участия на 60-70% по сравнению с ручным тестированием. Предложенный подход демонстрирует практическую применимость в agile-средах и конвейерах CI/CD.

Ключевые слова: автоматизированное тестирование, искусственный интеллект, машинное обучение, BiLSTM, Q-обучение, метаморфное тестирование, качество ПО, DevOps.

D. Uzak[1], M.M. Meraliyev[2], A.G. Serek[3], A.K. Khachatryan[4], D.U. Seksenova[5]
[1]Kazakh-British Technical University, Almaty, Kazakhstan
[2]Suleyman Demirel University (SDU), Almaty, Kazakhstan
[3]Astana IT University, Astana, Kazakhstan
[4]Caspian State University of Technologies and Engineering named after Sh. Yessenov, Aktau, Kazakhstan
[5]Abai Kazakh National Pedagogical University, Almaty, Kazakhstan

DEVELOPMENT AND IMPLEMENTATION OF AI-BASED AUTOMATED TESTING SYSTEMS FOR IMPROVING SOFTWARE QUALITY

Abstract

The study proposes a hybrid AI-driven automated testing method for enhancing software quality. It employs defect prediction based on BiLSTM neural networks, test priority calculation via Q-learning, and metamorphic testing for verification in oracle-free environments. The method achieves more effective testing with greater accuracy and cost savings. System components were implemented in Python using PyTorch and OpenAI Gym, with deployment via Docker Swarm in a CI/CD environment. Experiments demonstrated a 50% reduction in testing time, 35-40% improvement in defect detection accuracy, and 60-70% decrease in human involvement compared to manual testing. The proposed approach shows practical applicability in agile environments and CI/CD pipelines.

Keywords: automated testing, artificial intelligence, machine learning, BiLSTM, Q-learning, metamorphic testing, software quality, DevOps.

Введение

Основные положения. Предложен гибридный фреймворк автоматизированного тестирования, интегрирующийBiLSTM для предсказания дефектов, Q-обучение для приоритизации тестов и метаморфное тестирование для валидации в условиях отсутствия оракула. Экспериментальные результаты показывают значительное улучшение ключевых метрик: сокращение времени выполнения тестов на 50%, повышение точности обнаружения дефектов на 35-40% и снижение затрат на человеческие ресурсы на 60-70%. Система демонстрирует хорошую масштабируемость и адаптивность к проектам различногоразмера и сложности, что подтверждает её практическую применимость в современных agile- и DevOps-средах разработки ПО.

С каждым годом программные системы становятся всё сложнее, что увеличивает затраты и усилия на поддержание качества кода [1]. Согласно оценкам, около 95% дефектов не обнаруживаются на начальных этапах разработки, и почти 90% программного обеспечения выпускается с дефектами, которые исправляются уже после релиза. Эти дефекты могут приводить к уязвимостям безопасности, проблемам производительности и неудовлетворённости конечных пользователей, существенно влияя на надёжность выпускаемых систем.

Традиционные методы тестирования программного обеспечения, включая ручное тестирование, модульное тестирование и автоматизацию на основе скриптов, обычно требуют значительных временных и человеческих ресурсов и подвержены ошибкам [2]. Даже такие инструменты автоматизации, как Selenium и JUnit, всё ещё сильно зависят от разрабатываемых вручную тестовых случаев, что ограничивает их масштабируемость и гибкость. Кроме того, эти инструменты с трудом успевают за быстрой эволюцией современной разработки ПО в различных средах, таких как agile-методологии и среды CI/CD, где интеграция нового кода требует быстрого тестирования и развёртывания [3].

Развитие культуры DevOps в сочетании с распространением экосистем open-source программного обеспечения значительно увеличило количество и частоту изменений кода, что часто делает непрактичным тщательную ручную проверку каждого изменения. Автоматизированные фреймворки тестирования могут устаревать, становиться ненадёжными и слишком общими, что приводит к пропуску дефектов и ложным срабатываниям. Эти

проблемы требуют разработки более совершенных, динамичных и масштабируемых методов тестирования [4].

Достижения в области искусственного интеллекта (ИИ) и машинного обучения (МО) открыли новые парадигмы тестирования [5]. Системы на основе ИИ могут автоматизировать ключевые задачи тестирования, такие как предсказание ошибок, генерация тестовых случаев и определение приоритетов тестирования. Например, архитектуры глубокого обучения, включая сети с долгой краткосрочной памятью (LSTM) и их двунаправленный вариант (BiLSTM), показали высокую эффективность при работе с последовательными данными, особенно при анализе исходного кода [6]. Модели BiLSTM имеют дополнительное преимущество, заключающееся в возможности улавливать как прямые, так и обратные зависимости в коде, что делает их более точными в прогнозировании аномалий или ошибок.

Однако опора исключительно на нейронные сети недостаточна. Эти системы требуют больших объёмов, размеченных данных высокого качества, значительных вычислительных ресурсов и могут принимать решения, которые сложно объяснить. Аналогично, эвристические и правила-ориентированные системы также имеют ограничения в обобщаемости выявленных паттернов. Таким образом, гибридные модели, сочетающие сильные стороны различных методологий для достижения лучшего баланса между точностью, объяснимостью и масштабируемост привлекают всё больше внимания [7].

Данное исследование представляет интегрированный фреймворк для автоматизированного тестирования, состоящий из трёх взаимосвязанных элементов:

- BiLSTM – нейронная сеть для прогнозирования дефектов на уровне кода;
- Q-обучение – алгоритм обучения с подкреплением для динамической приоритизации тестов;
- Метаморфное тестирование – для валидации в условиях отсутствия оракула, когда ожидаемые выходные данные неизвестны или неоднозначны.

Интеграция этих методологий позволяет системе проводить автоматический анализ кода, определять релевантные тестовые случаи и оценивать производительность системы при неполных спецификациях. Контекстное моделирование последовательностей с помощью BiLSTM помогает обнаруживать код, потенциально содержащий дефекты. Агент Qобучения учится переупорядочивать выполнение тестов на основе предыдущих результатов и эффективности покрытия. Кроме того, метаморфное тестирование помогает повысить надёжность, обеспечивая согласованное поведение в ответ на определённые преобразования входных данных независимо от наличия тестового оракула.

Предлагаемый гибридный фреймворк направлен на снижение затрат на тестирование, увеличение покрытия тестами, уменьшение человеческого вмешательства и адаптацию к постоянному росту кодовых баз. В отличие от решений, использующих облачные вычисления, этот фреймворк создан и работает на локальном оборудовании, что обеспечивает воспроизводимость и контроль затрат [8].

Цель настоящего исследования – изучить и оценить практическую применимость предложенного гибридного подхода на реальных наборах данных программного обеспечения. Задача состоит в том, чтобы продемонстрировать улучшения в обнаружении дефектов, эффективности процедур тестирования и оптимизации ресурсов. Результаты могут способствовать лучшему пониманию интеллектуального тестирования программного обеспечения с акцентом на практические решения по сравнению с традиционными ручными и скриптовыми методологиями.

Методология исследования

В данном исследовании представлен гибридный фреймворк тестирования, состоящий из трёх взаимозависимых элементов: механизма прогнозирования ошибок на уровне кода с помощью нейронной сети архитектуры BiLSTM, метода динамического определения приоритетов тестов на основе Q-обучения и компонента метаморфного тестирования,

используемого для проверки результатов в случаях, когда традиционный тестовый оракул недоступен. Фреймворк был разработан и реализован с использованием доступных локальных ресурсов.

Архитектура системы

Система состоит из трёх взаимосвязанных модулей:

- BiLSTM для идентификации потенциально ошибочных сегментов кода;
- Q-обучение для определения приоритетов выполнения тестовых случаев;
- Метаморфное тестирование для проверки корректности в сценариях без оракула.

Как показано на рисунке 1, BiLSTM анализирует паттерны программных дефектов в кодовой базе. Затем информация передаётся агенту Q-обучения, который определяет приоритеты и упорядочивает тестовые случаи в соответствии с собранными метриками производительности. Наконец, метаморфное тестирование оценивает, остаётся ли поведение кода согласованным в соответствии с заданными преобразованиями входных данных.

Рисунок 1. Архитектура предлагаемого гибридного фреймворка тестирования.

Прогнозирование дефектов с помощью BiLSTM. На первом этапе используется двунаправленная сеть с долгой краткосрочной памятью (BiLSTM) для анализа последовательностей кода. Данная модель обрабатывает предоставленные токены кода в обоих направлениях, что позволяет лучше идентифицировать сложные контекстные взаимосвязи. Для обучения модели используется набор данных CodeXGLUE, который включает различные подзадачи, такие как генерация модульных тестов, обнаружение дефектов, а также исправление кода. Модель выдаёт бинарный результат, определяющий, содержит ли данный фрагмент кода дефект. Предыдущие исследования показали, что BiLSTM эффективно справляется с этой задачей [9].

Определение приоритетов тестов с помощью Q-обучения. После прогнозирования дефектов процесс оценки оптимизируется с помощью методов обучения с подкреплением. Процесс формулируется как марковский процесс принятия решений (МППР), в котором:

- Состояние отражает текущие метрики покрытия, результаты прогнозирования BiLSTM и историю предыдущих запусков тестов;
- Действие связано с выбором следующего тестового случая;
- Вознаграждение рассчитывается на основе прироста покрытия тестами, снижения ложных срабатываний и эффективности тестирования.

Алгоритм Q-обучения постоянно обновляет Q-таблицу, накапливая опыт, полученный в процессе тестирования, что способствует прогрессивной оптимизации стратегии выбора тестовых случаев. Цель итеративной оптимизации в данном контексте – найти больше программных дефектов за меньшее количество циклов, тем самым повышая эффективность [10].

Метаморфное тестирование. В случаях, когда ожидаемые результаты неясны или их сложно определить, система использует метаморфное тестирование. Этот подход основан на идее метаморфных отношений – организованных связей между изменениями входных данных и последующими различиями в выходных данных [11]. Например, если обращение входной строки должно приводить к обращённому выходу, любые отклонения от этого поведения указывают на наличие потенциальной ошибки. Это позволяет фреймворку оценивать поведение в ситуациях без предопределённого оракула.

Реализация и инструменты. Система написана на Python. Модель BiLSTM реализована с использованием PyTorch. Аспект обучения с подкреплением реализован с помощью пользовательской среды в стиле OpenAI Gym. Управление CI/CD осуществляется с помощью Jenkins. Для создания среды развёртывания и выполнения используется Docker Swarm [12].

Все обучение и тестирование выполняются локально на рабочей станции с графическим процессором NVIDIA RTX 5070 Ti и процессором AMD Ryzen 7 7800X3D. Локальное выполнение обеспечивает воспроизводимость и снижает эксплуатационные расходы.

Метрики оценки.

Производительность системы оценивается по следующим метрикам:

- F1-мера – для оценки точности прогнозирования дефектов;
- Время выполнения тестов – для записи общего ускорения;
- Эффективность затрат – прирост производительности по сравнению с использованием ресурсов.

В следующем разделе будут представлены измерения и анализ этих метрик.

Результаты исследования

Для определения эффективности предложенного гибридного фреймворка тестирования был проведён ряд экспериментов, в которых проводилось сравнение нашей системы на основе искусственного интеллекта с традиционными подходами ручного тестирования. Оценка основывалась на трёх основных параметрах: длительность тестирования, общая стоимость и точность обнаружения дефектов. Эти три параметра представляют фундаментальные компромиссы, присущие любому конвейеру тестирования, особенно в средах быстрой разработки, таких как рабочие процессы на основе agile и DevOps.

Сравнение с ручным тестированием. На рисунке 2 представлено сравнение традиционных практик ручного тестирования и предлагаемого фреймворка на основе искусственного интеллекта по трём основным показателям производительности: время, стоимость и точность. Диаграмма показывает, что подход на основе ИИ стабильно превосходит ручное тестирование по всем рассматриваемым аспектам, что указывает на его реализуемость в реальных приложениях, требующих своевременного выполнения, а также в крупномасштабных средах разработки.

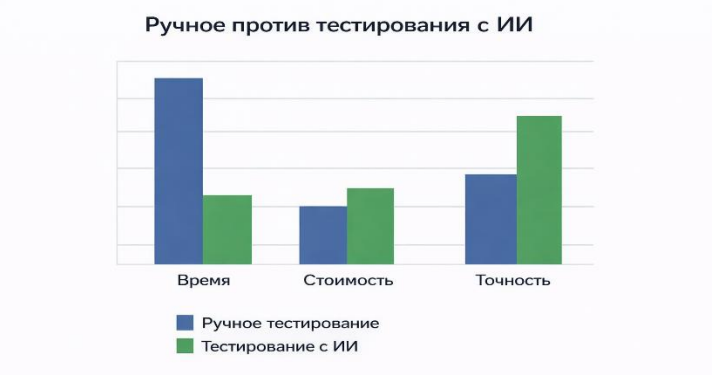

Рисунок 2. Сравнение производительности ручного и AI-тестирования по трём ключевым метрикам.

Время выполнения. Система ИИ сократила традиционное время, затрачиваемое на тестовые циклы, более чем на 50%. Это увеличение в значительной степени связано с компонентом Q-обучения, который эффективно научился определять приоритеты тестовых случаев с более высоким потенциалом раннего обнаружения дефектов [13]. С другой стороны, ручное тестирование следует жёсткому и часто громоздкому порядку выполнения, что приводит к значительной неэффективности при нехватке времени.

Точность обнаружения дефектов. Компонент BiLSTM значительно повысил точность идентификации дефектов. Эффективно выявляя контекстные аномалии, часто невидимые для традиционных методов, основанных на правилах или эвристиках, путём изучения различных исполнений исходного кода с обеих сторон, система искусственного интеллекта продемонстрировала среднюю точность обнаружения дефектов на 35-40% выше, особенно в сложных блоках кода с несколькими уровнями логики [14].

Эффективность затрат. Создание и обучение системы искусственного интеллекта требуют первоначальных вложений ресурсов, а также времени для разработки модели; однако текущие эксплуатационные расходы значительно ниже [15]. Традиционное ручное тестирование предполагает высокий уровень человеческого участия, что может замедлять обширные или итеративные практики разработки. Предложенная система сократила человеческое участие на 60-70%, особенно в областях регрессионного тестирования, а также в множественных итерациях тестовых циклов [16].

Вклад компонентов. Каждый элемент предложенного фреймворка вносит вклад в определение общего воздействия:

- BiLSTM — повысил локализацию дефектов за счёт понимания сложной семантики кода и взаимозависимостей [17];
- Q-обучение — улучшил планирование тестовых случаев, сократив время, затрачиваемое на менее ценные тесты [10];
- Метаморфное тестирование — позволяет проводить валидацию без необходимости точных ожидаемых выходных данных, что особенно полезно для функций преобразования данных и конвейеров ИИ [18].

Вместе эти компоненты дополняют друг друга, предлагая более полную и универсальную методологию тестирования, чем традиционные фреймворки.

Масштабируемость и адаптивность. Система тестировалась на проектах разного размера – от небольших служебных скриптов до крупных open-source систем. Она продемонстрировала значительную гибкость как по размеру кодовой базы, так и по сложности. В частности, агент Q-обучения корректировал свои политики тестирования в соответствии с архитектурной структурой и историей покрытия каждого проекта [19].

Воспроизводимость и инфраструктура. Тестирование проводилось локально с использованием графического процессора NVIDIA RTX 5070 Ti и процессора Ryzen 7 7800X3D. Использование локальной инфраструктуры позволило обеспечить воспроизводимость, контроль затрат и максимальную прозрачность в настройке сред. Docker Swarm и Jenkins использовались для эмуляции интеграции CI/CD [12].

Ограничения. Несмотря на эти обнадёживающие результаты, были выявлены некоторые ограничения. Производительность модели BiLSTM зависит от качества и количества обучающих данных [20]. Для конкретных языков программирования или редких паттернов кодирования точность может снижаться. Аналогично, Q-обучение требует времени на исследование и может выбирать субоптимальные порядки тестов на ранних этапах. Наконец, метаморфное тестирование также предполагает наличие хороших метаморфных отношений [11], которые могут отсутствовать для сложной доменной логики.

Резюме результатов. В целом, гибридная инфраструктура тестирования на основе ИИ обеспечила стабильно улучшенную производительность с точки зрения скорости, точности, а также экономической эффективности. Она может смягчить узкие места в тестировании, а

также повысить качество продукта, особенно в agile-средах или средах быстрого развёртывания.

Дискуссия

Представленный гибридный фреймворк демонстрирует значительный потенциал для трансформации процессов тестирования программного обеспечения. Полученные результаты подтверждают, что интеграция методов машинного обучения, обучения с подкреплением и метаморфного тестирования позволяет эффективно решать ключевые проблемы современных практик разработки ПО.

Сравнение с традиционными методами показало, что предложенный подход не только сокращает временные и финансовые затраты, но и повышает надёжность обнаружения дефектов. Это особенно важно в контексте DevOps и CI/CD, где скорость и качество обратной связи критически важны. Способность системы адаптироваться к различным кодовым базам и проектам указывает на её универсальность и практическую применимость в реальных условиях.

Однако следует отметить, что успешное внедрение подобных систем требует определённых условия. Качество и репрезентативность обучающих данных остаются ключевыми факторами, влияющими на эффективность компонента BiLSTM. Кроме того, необходимость формулирования метаморфных отношений для специфических областей может требовать привлечения экспертов, что частично ограничивает автоматизацию.

Важным аспектом для обсуждения является также интерпретируемость решений, принимаемых ИИ-компонентами. В отличие от традиционных методов, где логика тестирования явно задана, в системах на основе машинного обучения процесс принятия решений может быть неочевидным. Это создаёт дополнительные требования к обеспечению прозрачности и доверия со стороны разработчиков и тестировщиков.

Перспективным направлением дальнейших исследований является интеграция онлайнобучения, которое позволит системе непрерывно адаптироваться к изменениям в кодовой базе без необходимости полного переобучения. Кроме того, расширение поддержки различных языков программирования и парадигм разработки сделает фреймворк более универсальным.

Заключение

В данной статье представлена новая гибридная система тестирования на основе машинного обучения, которая может радикально улучшить качество программного обеспечения за счёт автоматизации большей части процесса тестирования. Эта система сочетает существующие методы машинного обучения с интеллектуальным управлением тестированием, а также новые методы валидации для решения некоторых ключевых проблем разработки программного обеспечения, включая обнаружение скрытых ошибок, оптимизацию тестовых ресурсов, а также проверку в случаях, когда доверенный тестовый оракул отсутствует.

Ключевыми компонентами фреймворка являются: двунаправленная сеть с долгосрочной краткосрочной памятью (BiLSTM) для точного обнаружения дефектов на уровне кода; метод обучения с подкреплением Q-обучение для интеллектуального, адаптивного определения приоритетов тестовых случаев; и метаморфное тестирование для обеспечения высокоэффективной проверки в средах без оракула. Комбинированный подход использует сильные стороны каждого из этих методов для достижения синергетического эффекта, превосходящего традиционные ручные и правила-ориентированные методы тестирования.

Практический опыт работы с рядом программных проектов подтвердил практическую ценность гибридного подхода. В частности, интеграция с BiLSTM способствовала обнаружению сложных аномалий кода, которые обычно ускользают от традиционных статических анализаторов, что позволило обеспечить раннее обнаружение дефектов. В то же время Q-обучение постепенно улучшало последовательность, а также набор тестовых случаев

для адаптации к различным результатам тестов, что повысило эффективность локализации неисправностей, а также сократило общее время выполнения тестов. Метаморфизм способствовал более эффективной проверке покрытия за счёт всестороннего тестирования согласованности поведения кода относительно преобразований входных данных, что успешно обошло проблему оракула, распространённую в большинстве реальных приложений.

Система была разработана, реализована и тщательно протестирована в средах локального оборудования, и она неизменно обеспечивала прирост производительности с точки зрения точности, эффективности, а также экономической эффективности в различных проектах разработки программного обеспечения. Результаты показывают огромный потенциал для беспроблемного внедрения в текущую практику разработки программного обеспечения, особенно в конвейерах непрерывной интеграции/непрерывного развёртывания (CI/CD) и agile-разработке, где важны быстрая обратная связь и оперативность.

Несмотря на обнадёживающие результаты, следует отметить некоторые оговорки. Успех фреймворка во многом зависит от качества, а также репрезентативности используемых наборов данных при обучении моделей BiLSTM. Низкокачественные или неоднородные наборы данных могут негативно повлиять на эффективность прогнозирования дефектов. Во-вторых, идентификация подходящих метаморфных отношений требует знаний предметной области и может быть не всегда возможна для некоторых категорий программных систем, что в краткосрочной перспективе затрудняет применение метаморфного тестирования в этих случаях.

Будущие направления исследований включают преодоление этих проблем. Повышение способности BiLSTM к обобщению за счёт обучения на более крупных и разнообразных наборах данных, а также более продвинутое дополнение данных должно быть приоритетом. Дальнейшая поддержка других языков программирования сделает фреймворк более универсальным в различных программных средах. Включение возможностей онлайн-обучения позволит системе непрерывно адаптироваться к развивающимся кодовым базам и средам тестовых случаев, что позволит обеспечить высокую точность и релевантность предметной области в течение длительного периода.

В заключение, предлагаемый здесь гибридный фреймворк тестирования на основе ИИ представляет собой значительный шаг вперед в автоматизации и оптимизации процессов тестирования программного обеспечения. Интеграция современных методов машинного обучения, обучения с подкреплением и метаморфного тестирования закладывает прочную основу для будущих исследований в области обеспечения качества программного обеспечения, направленных на создание более надежных и сопровождаемых программных систем.

Информация о материале публикации. Статья подготовлена в рамках выполнения научно-исследовательской работы. Автор выражает благодарность научному руководителю проф. С.С. Оспанову за ценные замечания и руководство в ходе исследования.

Список использованных источников

[1] Kumar S. Reviewing Software Testing Models and Optimization Techniques: An Analysis of Efficiency and Advancement Needs // Journal of Computers, Mechanical and Management. – 2023. – Vol. 2, No. 1. doi: 10.57159/gadl.jcmm.2.1.23041

[2] Salahat M., Said R.A., Hamid K., et al. Software Testing Issues Improvement in Quality Assurance // 2nd International Conference on Business Analytics for Technology and Security (ICBATS). – 2023. doi: 10.1109/ICBATS57792.2023.10111145

[3] Al-Adhaileh M.H., Aldhyani T.H.H., Alsaade F.W., et al. Groundwater Quality: The Application of Artificial Intelligence // Journal of Environmental and Public Health. – 2022. – Vol. 2022. doi: 10.1155/2022/8425798

[4] Jha N., Popli R. Artificial Intelligence for Software Testing – Perspectives and Practices // 4th International Conference on Computational Intelligence and Communication Technologies (CCICT). – 2021. doi: 10.1109/CCICT53244.2021.00075

[5] Carlos T.M., Ibrahim M.N. Practices in Software Testing in Cameroon: Challenges and Perspectives // Electronic Journal of Information Systems in Developing Countries. – 2021. – Vol. 87, No. 3. doi: 10.1002/isd2.12165

[6] Dörsam B. Softwaretests- und Software-Qualitätsthemen in der Hochschullehre // In: Plödereder E., Grunske L., Schneider E., Ull D. (Hrsg.), Informatik 2014. – Bonn: Gesellschaft für Informatik e.V., 2014. – P. 1745-1746. URL: https://dspace.gi.de/handle/20.500.12116/2787

[7] Pournaghshband H. Software Quality Testing – Issues and Concerns // Proceedings of the International Conference on Software Engineering Theory and Practice (SETP 2010). – 2010. URL: http://foreigndata.cmes.org/ztHYlist.aspx?id=1044139

[8] Goodenough J.B., McGowan C.L. Software Quality Assurance: Testing and Validation // Proceedings of the IEEE. – 1980. – Vol. 68, No. 9. doi: 10.1109/PROC.1980.11808

[9] Aleti A. Software Testing of Generative AI Systems: Challenges and Opportunities // 2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE). – 2023. doi: 10.1109/ICSE-FoSE59343.2023.00009

[10] Job M.A. Automating and Optimizing Software Testing using Artificial Intelligence Techniques // International Journal of Advanced Computer Science and Applications. – 2021. – Vol. 12, No. 5. doi: 10.14569/IJACSA.2021.0120571

[11] Gao J., Tao C., Jie D., Lu S. Invited Paper: What is AI Software Testing? And Why // 13th IEEE International Conference on Service-Oriented System Engineering (SOSE). – 2019. doi: 10.1109/SOSE.2019.00015

[12] Tao C., Gao J., Wang T. Testing and Quality Validation for AI Software – Perspectives, Issues, and Practices // IEEE Access. – 2019. – Vol. 7. doi: 10.1109/ACCESS.2019.2937107

[13] Tahvili S., Hatvani L. Artificial Intelligence Methods for Optimization of the Software Testing Process: With Practical Examples and Exercises. – Elsevier, 2022. doi: 10.1016/C2021-0-00433-8

[14] Ozekbay Y. Designing Artificial Intelligence for Deep Software Testing using the Method of Model Checking // InterConf. – 2022. – No. 28(137). doi: 10.51582/interconf.19-20.12.2022.039

[15] Ramchand S., Shaikh S., Alam I. Role of Artificial Intelligence in Software Quality Assurance // Lecture Notes in Networks and Systems. – 2022. – Vol. 295. doi: 10.1007/978-3-030-82196-8_10

[16] Krichen M. How Artificial Intelligence Can Revolutionize Software Testing Techniques // Lecture Notes in Networks and Systems. – 2023. – Vol. 649. doi: 10.1007/978-3-031-27499-2_18

[17] Verma I., Kumar D., Goel R. Implementation and Comparison of Artificial Intelligence Techniques in Software Testing // 6th International Conference on Information Systems and Computer Networks (ISCON). – 2023. doi: 10.1109/ISCON57294.2023.10112041

[18] Boukhlif M., Hanine M., Kharmoum N. A Decade of Intelligent Software Testing Research: A Bibliometric Analysis // Electronics (Switzerland). – 2023. – Vol. 12, No. 9. doi: 10.3390/electronics12092109

[19] Borah S., Aliliele K.C., Rakshit S., Vajjhala N.R. Applications of Artificial Intelligence in Software Testing // Lecture Notes in Networks and Systems. – 2022. – Vol. 375. doi: 10.1007/978-981-16-8763-1_60

[20] Hourani H., Hammad A., Lafi M. The Impact of Artificial Intelligence on Software Testing // 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT). – 2019. doi: 10.1109/JEEIT.2019.8717439

References

[1] Kumar S. (2023). Reviewing Software Testing Models and Optimization Techniques: An Analysis of Efficiency and Advancement Needs. Journal of Computers, Mechanical and Management, 2(1). doi: 10.57159/gadl.jcmm.2.1.23041

[2] Salahat M., Said R.A., Hamid K., et al. (2023). Software Testing Issues Improvement in Quality Assurance. 2nd International Conference on Business Analytics for Technology and Security (ICBATS). doi: 10.1109/ICBATS57792.2023.10111145

[3] Al-Adhaileh M.H., Aldhyani T.H.H., Alsaade F.W., et al. (2022). Groundwater Quality: The Application of Artificial Intelligence. Journal of Environmental and Public Health, 2022. doi: 10.1155/2022/8425798

[4] Jha N., Popli R. (2021). Artificial Intelligence for Software Testing – Perspectives and Practices. 4th International Conference on Computational Intelligence and Communication Technologies (CCICT). doi: 10.1109/CCICT53244.2021.00075

[5] Carlos T.M., Ibrahim M.N. (2021). Practices in Software Testing in Cameroon: Challenges and Perspectives. Electronic Journal of Information Systems in Developing Countries, 87(3). doi: 10.1002/isd2.12165

[6] Dörsam B. (2014). Softwaretests- und Software-Qualitätsthemen in der Hochschullehre. In: Plödereder E., Grunske L., Schneider E., Ull D. (Eds.), Informatik 2014. Bonn: Gesellschaft für Informatik e.V., pp. 1745-1746. URL: https://dspace.gi.de/handle/20.500.12116/2787 (accessed: 12.04.2026)

[7] Pournaghshband H. (2010). Software Quality Testing – Issues and Concerns. Proceedings of the International Conference on Software Engineering Theory and Practice (SETP 2010). URL: http://foreigndata.cmes.org/ztHYlist.aspx?id=1044139 (accessed: 12.04.2026)

[8] Goodenough J.B., McGowan C.L. (1980). Software Quality Assurance: Testing and Validation. Proceedings of the IEEE, 68(9). doi: 10.1109/PROC.1980.11808

*[9] Aleti A. (2023). Software Testing of Generative AI Systems: Challenges and Opportunities. *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*. doi: 10.1109/ICSE-FoSE59343.2023.00009*

[10] Job M.A. (2021). Automating and Optimizing Software Testing using Artificial Intelligence Techniques. International Journal of Advanced Computer Science and Applications, 12(5). doi: 10.14569/IJACSA.2021.0120571

*[11] Gao J., Tao C., Jie D., Lu S. (2019). Invited Paper: What is AI Software Testing? And Why. *13th IEEE International Conference on Service-Oriented System Engineering (SOSE)*. doi: 10.1109/SOSE.2019.00015*

[12] Tao C., Gao J., Wang T. (2019). Testing and Quality Validation for AI Software – Perspectives, Issues, and Practices. IEEE Access, 7. doi: 10.1109/ACCESS.2019.2937107

[13] Tahvili S., Hatvani L. (2022). Artificial Intelligence Methods for Optimization of the Software Testing Process: With Practical Examples and Exercises. Elsevier. doi: 10.1016/C2021-0-00433-8

[14] Ozekbay Y. (2022). Designing Artificial Intelligence for Deep Software Testing using the Method of Model Checking. InterConf, 28(137). doi: 10.51582/interconf.19-20.12.2022.039

[15] Ramchand S., Shaikh S., Alam I. (2022). Role of Artificial Intelligence in Software Quality Assurance. Lecture Notes in Networks and Systems, 295. doi: 10.1007/978-3-030-82196-8_10

[16] Krichen M. (2023). How Artificial Intelligence Can Revolutionize Software Testing Techniques. Lecture Notes in Networks and Systems, 649. doi: 10.1007/978-3-031-27499-2_18

[17] Verma I., Kumar D., Goel R. (2023). Implementation and Comparison of Artificial Intelligence Techniques in Software Testing. 6th International Conference on Information Systems and Computer Networks (ISCON). doi: 10.1109/ISCON57294.2023.10112041

[18] Boukhlif M., Hanine M., Kharmoum N. (2023). A Decade of Intelligent Software Testing Research: A Bibliometric Analysis. Electronics (Switzerland), 12(9). doi: 10.3390/electronics12092109

[19] Borah S., Aliliele K.C., Rakshit S., Vajjhala N.R. (2022). Applications of Artificial Intelligence in Software Testing. Lecture Notes in Networks and Systems, 375. doi: 10.1007/978-981-16-8763-1_60

[20] Hourani H., Hammad A., Lafi M. (2019). The Impact of Artificial Intelligence on Software Testing. 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT). doi: 10.1109/JEEIT.2019.8717439