Касымова Афифе Наримановна
преподаватель кафедры прикладной информатики
ГБОУВО РК КИПУ имени Февзи Якубова
г. Симферополь, Россия

АВТОМАТИЗАЦИЯ РАЗРАБОТКИ ПРОГРАММНЫХ КОМПОНЕНТОВ С ПОМОЩЬЮ ИИ-АГЕНТОВ: ОПЫТ СОЗДАНИЯ И ВАЛИДАЦИИ СКИЛЛА ДЛЯ ПЛАТФОРМЫ 1С-БИТРИКС

Аннотация. В работе рассматривается практика автоматизации разработки программных компонентов средствами ИИ-агента на примере скилла для платформы 1С-Битрикс. Кратко проанализированы отечественные публикации о применении искусственного интеллекта и машинного обучения в разработке программного обеспечения. Описаны скрипт-генератор и скрипт-валидатор компонента, а также эксперимент из пяти тестовых сборок: эталонной и четырёх с намеренно внесёнными типовыми дефектами. Показано, что валидатор обнаружил все четыре дефекта и не выдал ложных срабатываний на эталонной сборке, что подтверждает практическую пользу связки «генерация — автоматическая проверка — исправление».

Ключевые слова: искусственный интеллект, ИИ-агенты, автоматизация разработки, генерация кода, валидация кода, 1С-Битрикс, программные компоненты, машинное обучение.

Abstract. The paper examines the practice of automating software component development with an AI agent, illustrated by a skill built for the 1C-Bitrix platform. Russian publications on the use of artificial intelligence and machine learning in software development are briefly reviewed. A scaffolding script and a validation script for the component are described, together with an experiment involving five test builds: one clean baseline and four builds with deliberately introduced typical defects. The validator detected all four defects and produced no false positives on the baseline, confirming the practical value of the generate — automatically validate — fix loop.

Keywords: artificial intelligence, AI agents, software development automation, code generation, code validation, 1C-Bitrix, software components, machine learning.

Разработка типового программного компонента для CMS-платформ, включая 1С-Битрикс, требует создания устойчивого набора файлов: файла

описания для визуального редактора, файла параметров, файла бизнес-логики (устаревший component.php либо современный class.php на основе CBitrixComponent), шаблона вывода и языковых файлов локализации. Эта структура почти не меняется от компонента к компоненту, что делает её удобным объектом автоматизации: ИИ-агент может взять на себя как генерацию каркаса, так и последующую формальную проверку результата на соответствие конвенциям платформы. Совмещение генерации и автоматической валидации в одном скилле — предмет настоящей работы.

Вопросы применения искусственного интеллекта в разработке программного обеспечения рассматривались в ряде отечественных публикаций. Топалов Н. К. и Федорченко С. Г. выделяют четыре направления такого применения — автоматизацию тестирования и отладки кода, оптимизацию программных компонентов, предиктивную аналитику ошибок и автоматическую генерацию кода для типовых задач — и приводят обобщённые показатели эффективности (табл. 1). Донской В. И. рассматривает более общую теоретическую рамку интеллектуальной оптимизации на основе машинного обучения, применимую в том числе к задачам автоматизации инженерных процессов. Бевзенко С. А. систематизирует направления применения искусственного интеллекта и машинного обучения в разработке ПО, отмечая рост интереса к инструментам, снижающим долю рутинных операций. Общее для рассмотренных работ — утверждение о снижении трудозатрат и числа ошибок при внедрении ИИ-инструментов, однако конкретный механизм проверки корректности кода, сгенерированного ИИ-агентом, в них подробно не рассматривается: этому вопросу посвящена практическая часть настоящей статьи.

Таблица 1 — Показатели эффективности использования нейросетевых технологий в разработке ПО (по данным: Топалов, Федорченко, 2024)

Показатель	Значение, %
Рост интегрального показателя удовлетворённости качеством	10
Рост эффективности процессов	14
Рост автоматизации типовых операций	19
Экономия издержек	5

Практическая часть исследования — скилл для ИИ-агента, автоматизирующий создание компонента 1С-Битрикс и его проверку. Скилл включает четыре элемента (табл. 2): файл SKILL.md с инструкцией по

применению, справочный файл references/component_structure.md с описанием структуры компонента и типовых ошибок, скрипт-генератор scripts/scaffold_component.py и скрипт-валидатор scripts/validate_component.py.

Таблица 2 — Файловая структура скилла bitrix-component

Файл	Назначение
SKILL.md	Инструкция для ИИ-агента: когда применять скилл и в каком порядке вызывать скрипты.
references/component_structure.md	Справочник по структуре компонента и типовым ошибкам платформы.
scripts/scaffold_component.py	Генерирует каркас компонента: обязательные файлы, класс, шаблон, локализацию.
scripts/validate_component.py	Проверяет каркас на соответствие конвенциям и находит типовые ошибки.

Для проверки практической пользы связки «генерация — валидация» поставлен небольшой эксперимент. Скриптом-генератором создано пять экземпляров одного компонента (пространство имён testco, компонент widget.card): один — без изменений (эталон), четыре — с единичным намеренно внесённым дефектом каждый, воспроизводящим реальные ошибки, зафиксированные ранее при разработке скилла: отсутствие импорта use Bitrix\Main\Localization\Loc; при вызове Loc::getMessage(), отсутствие папки локализации lang/, точка в имени CSS-класса шаблона и отсутствие защиты B_PROLOG_INCLUDED. К каждому экземпляру применён scripts/validate_component.py; результаты приведены в табл. 3.

Таблица 3 — Результаты проверки пяти тестовых сборок

№	Внесённый дефект	Результат validate_component.py
1 (эталон)	Без изменений	0 ошибок — «критических проблем не найдено»
2	Удалён импорт use Bitrix\Main\Localization\Loc;	Обнаружено 1/1 — FAIL: Loc:: без use
3 (legacy)	Удалена папка lang/	Обнаружено 1/1 — FAIL: нет lang/<код>/
4	Точка в имени CSS-класса шаблона	Обнаружено 1/1 — FAIL: класс содержит точку
5	Удалена защита B_PROLOG_INCLUDED	Обнаружено 1/1 — FAIL: нет защиты B_PROLOG_INCLUDED

Валидатор обнаружил все четыре внесённых дефекта (4 из 4, 100 %) и не выдал ни одного ложного срабатывания на эталонной сборке. Общая схема цикла показана на рисунке. Важно, что все четыре дефекта относятся к ошибкам, которые трудно заметить при беглом визуальном просмотре кода — импорт класса, наличие отдельной папки, символ в имени CSS-класса, — однако каждая из них приводит к фатальной ошибке выполнения либо к поломке вёрстки в боевом окружении. Формальная проверка снимает этот класс ошибок ещё до этапа код-ревью, оставляя человеку проверку бизнес-логики и семантики — тех аспектов, которые статический анализ по регулярным выражениям заведомо не покрывает.

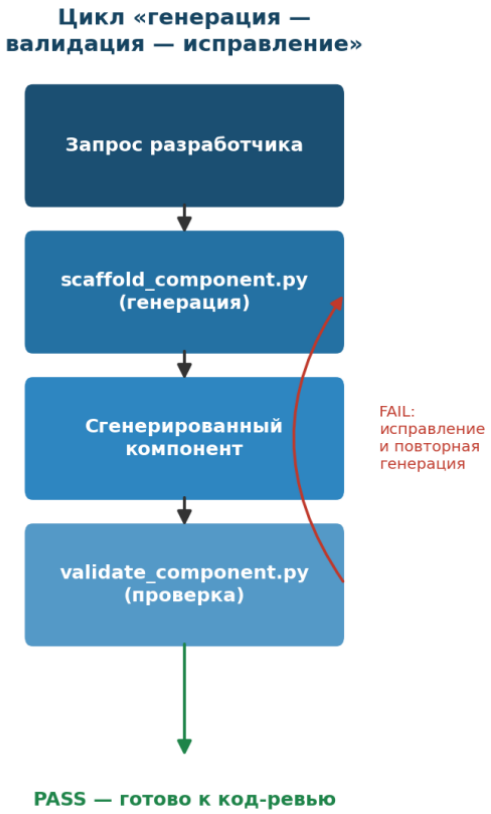

Рисунок — Цикл «генерация — автоматическая валидация — исправление»

Принцип «генерация каркаса + формальная проверка по правилам платформы» не специфичен для 1С-Битрикс и переносим на любые компонентные архитектуры со схожей структурой — шорткоды WordPress, модули Drupal, компоненты собственных CMS. Ограничение проведённого эксперимента — небольшой набор из пяти сборок и четырёх типов дефектов; для

практического внедрения в конвейер разработки целесообразно расширить корпус дефектов и включить проверку в CI-процесс, чтобы валидация выполнялась автоматически при каждом коммите, а не по запросу разработчика.

Таким образом, ИИ-агент, дополненный скиллом со скриптом-генератором и скриптом-валидатором, способен не только ускорять создание типовых программных компонентов, но и системно снижать долю ошибок определённого класса за счёт немедленной формальной проверки результата. Дальнейшая работа предполагает расширение набора проверяемых правил и количественную оценку эффекта на более крупной выборке реальных компонентов.

Список литературы

1. Топалов Н. К., Федорченко С. Г. Автоматизация разработки программного обеспечения с помощью искусственного интеллекта: как нейросети могут изменить процессы разработки // Молодой учёный. — 2024. — № 46 (545). — С. 23–25.
2. Донской В. И. Интеллектуальная оптимизация на основе машинного обучения: современное состояние и перспективы (обзор) // Таврический вестник информатики и математики. — 2020. — № 1. — С. 32–63.
3. Бевзенко С. А. Применение искусственного интеллекта и машинного обучения в разработке программного обеспечения // Инновации и инвестиции. — 2023. — № 8. — С. 187–191.