УДК 004

ПРИНЦИПЫ И МЕТОДЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ В ЭПОХУ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА И МАШИННОГО ОБУЧЕНИЯ: ВЫЗОВЫ И АДАПТАЦИЯ

Раевский В.А.

канд. техн. наук,

Калужский государственный университет им. К.Э. Циолковского,

Калуга, Россия

Калашников А.С.

магистрант,

Калужский государственный университет им. К.Э. Циолковского, Калуга, Россия

Аннотация

Представленная статья посвящена комплексному исследованию адаптации, трансформации и дополнения фундаментальных принципов программной инженерии для эффективного решения задач создания и поддержки систем с искусственным интеллектом и машинным обучением (ИИ/МО). Приводится анализ возникновения и ключевых компонентов новой парадигмы Machine Learning Operations. Рассматриваются уникальные особенности и современные вызовы в разработке систем с ИИ/МО; предлагаются методы адаптации классических принципов программной инженерии к современным условиям.

Ключевые слова: искусственный интеллект, машинное обучение, программная инженерия, проблематика разработки.

SOFTWARE ENGINEERING PRINCIPLES AND METHODS IN THE ERA OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING: CHALLENGES AND ADAPTATION

Raevsky V.A.
Candidate of Technical Sciences,
Kaluga State University named after K.E. Tsiolkovski,
Kaluga, Russia

Kalashnikov A.S.
Master's Degree Student,
Kaluga State University named after K.E. Tsiolkovski,
Kaluga, Russia

Abstract

The presented article is devoted to a comprehensive study of the adaptation, transformation and complementation of the fundamental principles of software engineering for the effective solution of the tasks of creating and maintaining systems with artificial intelligence and machine learning (AI/ML). The analysis of the emergence and key components of the new Machine Learning Operations paradigm is presented. The unique features and modern challenges in the development of AI/ ML systems are considered; methods for adapting classical principles of software engineering to modern conditions are proposed.

Keywords: artificial intelligence, machine learning, software engineering, development issues.

Введение. Искусственный интеллект (ИИ) и машинное обучение (МО) перестали быть областью чистой науки или технологическим экспериментом, став мощными двигателями цифровой трансформации. Алгоритмы ИИ/МО

стремительно интегрируются в критически важные аспекты современной жизни и бизнес-процессы: от персональных рекомендаций и систем распознавания образов до автономного транспорта и медицинской диагностики [1; 5; 6; 7; 8]. Создание не просто исследовательского прототипа, а надежных, масштабируемых, безопасных и поддерживаемых промышленных систем на основе ИИ/МО предъявляет беспрецедентные требования к процессам их разработки, развертывания и сопровождения [2; 9; 11]. Таким образом, на первый план выходит программная инженерия как дисциплина, обеспечивающая предсказуемость и качество. Традиционный жизненный цикл разработки программного обеспечения (Software Development Life Cycle, SDLC) [3] сталкивается с уникальными вызовами применительно к системам с ИИ/МО. Существует значительный разрыв между экспериментальной, итеративной природой работы Data Scientist'a и жесткими требованиями к промышленной эксплуатации программного обеспечения, что требует глубокой переоценки классических подходов в программной инженерии.

Уникальные особенности разработки систем с ИИ/МО и новые вызовы. Разработка программных систем, интегрирующих компоненты искусственного интеллекта и машинного обучения, принципиально отличается от создания традиционного программного обеспечения [4; 11]. Эта специфика порождает ряд уникальных инженерных вызовов, требующих переосмысления классических подходов программной инженерии.

1) *Данные как «первоклассный» артефакт.* В отличие от традиционного программного обеспечения, где логика жестко закодирована, поведение систем ИИ/МО в решающей степени определяется данными, на которых они обучаются. Это возводит данные в статус критически важного, «первоклассного» артефакта наравне с исходным кодом.

– Качество данных, «Мусор на входе – мусор на выходе» (Garbage In, Garbage Out, GIGO). Неточные, неполные, несбалансированные или смещенные (biased) данные приводят к некорректным или несправедливым предсказаниям

модели. Обеспечение и поддержание высокого качества данных – постоянная задача.

– Подготовка данных (Feature Engineering) – процесс извлечения, очистки, трансформации и создания признаков (features) из «сырых» данных зачастую занимает до 80% времени проекта МО, требует глубокого предметного знания и является скорее искусством, чем строгой инженерией.

– Версионирование данных. Необходимо отслеживать не только версии кода и моделей, но и версии наборов данных и процессов их подготовки, чтобы гарантировать воспроизводимость экспериментов и результатов.

– Концептуальный Дрейф (Data Drift). Статистические свойства реальных данных, на которых модель работает после развертывания, неизбежно дрейфуют со временем относительно данных, на которых модель обучалась. Это приводит к постепенной деградации качества предсказаний модели (устареванию модели).

2) Недетерминизм и статистическая природа. Поведение систем ИИ/МО носит вероятностный, а не детерминированный характер. Один и тот же код (модель + данные) может давать разные результаты при повторных запусках (из-за стохастичности алгоритмов обучения, различий в инициализации, аппаратных особенностях). Это создает фундаментальные сложности:

– Тестирование и верификация. Традиционные методы модульного и интеграционного тестирования, основанные на проверке строгого соответствия ожидаемому результату, неприменимы напрямую. Требуется проверять статистические метрики (точность, полнота, F1-score и т.д.) на репрезентативных валидационных и тестовых наборах данных.

– Воспроизводимость (Reproducibility). Крайне сложно гарантировать, что эксперимент, проведенный одним разработчиком, даст точно такие же результаты у другого разработчика. Необходима фиксация всех зависимостей, окружения, сидов (seeds) генераторов случайных чисел.

– Гарантии поведения. Невозможно формально доказать корректность модели во всех возможных сценариях, как это иногда делается для критического ПО. Поведение модели в экстремальных или неучтенных входных данных непредсказуемо.

3) *Двойная эволюция жизненного цикла систем ИИ/МО*. Эволюция модели и данных порождает постоянную необходимость переобучать или заменять модели из-за дрейфа данных, появления новых данных, улучшения алгоритмов, изменения бизнес-требований.

4) *Проблемы интерпретируемости (Explainability) и доверия*. Многие современные модели ИИ/МО, например глубокие нейронные сети, работают как «черные ящики»: их внутренняя логика принятия решений сложна для понимания человеком. Это порождает следующие проблемы проблемы:

– Доверие пользователей и стейкхолдеров. Каким образом убедить пользователя, врача, регулятора или судью в правильности решения, если невозможно объяснить, почему оно было принято.

– Отладка и улучшение модели. Сложно выявить причины ошибочных предсказаний без понимания внутренних механизмов модели.

– Выявление смещений (Bias): Невозможность «заглянуть внутрь» модели затрудняет обнаружение и устранение скрытых смещений в данных или алгоритме, ведущих к дискриминационным результатам. Требования к интерпретируемости особенно критичны в таких областях как финансы, медицина, юриспруденция, уголовное право.

5) *Этические и регуляторные аспекты*. Разработка и внедрение систем ИИ/МО несут значительную этическую нагрузку и привлекают внимание регуляторов [10].

– Смещения (Bias) и справедливость (Fairness). Модели могут усиливать и масштабировать «предубеждения», присутствующие в обучающих данных,

приводя к дискриминационным исходам. Обеспечение справедливости алгоритмов – ключевая этическая задача.

– Конфиденциальность данных (Privacy). Обучение моделей часто требует больших объемов персональных или чувствительных данных. Необходимо строго соблюдать законодательство и применять методы защиты приватности.

– Ответственность. Кто несет ответственность за вред, причиненный решением автономной системы ИИ; четкое определение зон ответственности разработчиков, операторов и пользователей.

– Регуляторное давление. Появление законов и стандартов, регулирующих разработку и использование ИИ (например, Европейский Акт об ИИ - EU AI Act), накладывает дополнительные юридические и технические требования на инженерные процессы.

Эти уникальные особенности и возникающие по их причине вызовы позволяют сделать вывод, что применение классических принципов и методов программной инженерии к разработке систем ИИ/МО недостаточно. Требуется их глубокая адаптация и развитие новых специализированных практик.

Адаптация классических принципов программной инженерии. Специфические вызовы разработки систем с ИИ/МО, рассмотренные в предыдущем разделе, требуют их творческой адаптации и расширения классических принципов программной инженерии [11].

1) Управление жизненным циклом (Software Development Life Cycle, SDLC). Классические модели SDLC (Waterfall, V-Model) слишком ригидны для итеративной природы Data Science. Гибкие методологии (Agile, Scrum) лучше подходят, но также требуют модификации:

– Специфика итераций в МО. Спринты в проектах ИИ/МО часто включают не только разработку функциональности, но и этапы исследования данных, экспериментов с моделями, оценки их качества. Необходимо планировать время на непредсказуемые по длительности эксперименты.

– Инкрементальность в МО. Развертывание новой версии модели – это не просто замена кода, а замена ядра системы, что может кардинально изменить ее поведение. Требуется тщательное A/B тестирование, канареечные релизы и откат к предыдущей модели.

– DevOps для МО. Принципы DevOps (автоматизация, CI/CD) расширяются для включения этапов работы с данными и моделями, формируя основу MLOps. Ключевая задача – создание воспроизводимых конвейеров (pipelines) от сырых данных до работающей модели в продакшене.

2) Модульность и инкапсуляция. Критически важными для систем ИИ/МО являются:

– Разделение кода обучения и кода вывода (Inference). Код, используемый для обучения модели, принципиально отличается от оптимизированного, надежного кода, обслуживающего модель в реальном времени (REST API, gRPC сервис). Эти компоненты должны быть модульными и независимо развертываемыми.

– Инкапсуляция модели и логики обработки данных. Модель должна инкапсулировать внутреннюю логику предсказания. Код предварительной обработки данных (feature transformation) и постобработки результатов также должен быть выделен в отдельные, тестируемые модули. Это упрощает замену модели без переписывания всей логики приложения.

– Микросервисная архитектура. Выделение сервиса модели (Model Serving) в отдельный микросервис позволяет независимо масштабировать вычислительные ресурсы под нагрузку, обновлять модель без остановки всего приложения и использовать разные технологии стека.

3) Абстракция. Абстракция позволяет скрыть сложность реализации, предоставляя четкие интерфейсы.

– Хранилище признаков («фич») (Feature Store). Предоставляет единый интерфейс для инженерии и трансформации признаков; централизованное, версионируемое хранилище готовых «фич»; обеспечения одинаковых «фич»

как для обучения модели (offline), так и для вывода в реальном времени (online), устраняя тренировочно-сервисный перекос (training-serving skew).

– Реестр Моделей (Model Registry). Абстракция для управления жизненным циклом моделей. Позволяет хранить версии обученных моделей с метаданными (метрики, гиперпараметры, дата обучения, ссылка на данные/код), управлять стадиями жизненного цикла модели (Staging, Production, Archived), контролировать продвижение моделей из тестовой среды в продакшен.

4) Управление конфигурацией и версионирование. Принцип контроля версий должен распространяться за пределы исходного кода:

– Версионирование данных (Data Versioning). Инструменты DVC (Data Version Control) или интегрированные возможности платформ (MLflow, Weights & Biases) позволяют привязать конкретную версию набора данных (или даже отдельного файла) к коду эксперимента и обученной модели, что является основой воспроизводимости.

– Версионирование моделей. Каждая обученная модель рассматривается уникальный артефакт. Реестр моделей хранит эти артефакты и их версии, позволяя «откатиться» к предыдущей, рабочей модели при проблемах с новой.

– Версионирование конфигураций и гиперпараметров. Настройки обучения (learning rate, batch size, архитектура сети) и конфигурации окружения (версии библиотек, зависимости) должны быть зафиксированы вместе с кодом, данными и моделью для полной воспроизводимости.

Воспроизводимость окружений. Инструменты контейнеризации (Docker) и управления инфраструктурой как кодом (IaC - Terraform, Ansible) обеспечивают идентичность окружений для обучения, тестирования и продакшена.

5) Обеспечение качества (QA). Тестирование систем ИИ/МО должно выходить за рамки проверки функциональности кода:

– Качество данных. Проверки на наличие пропусков, выбросов, несбалансированности классов, смещений на этапе загрузки/подготовки.

– Дрейф данных. Автоматизированный мониторинг статистических характеристик входящих данных в продакшене и сравнение с характеристиками тренировочных данных.

– Функциональное тестирование (на уровне модели). Проверка модели на репрезентативном тестовом наборе данных на достижение целевых метрик качества (accuracy, precision, recall, F1, AUC-ROC и т.д.).

– Тестирование на смещения (Bias) и справедливость (Fairness). Оценка работы модели на различных подгруппах данных для выявления дискриминационных предсказаний.

– Тестирование устойчивости (Robustness). Проверка реакции модели на зашумленные, искаженные или слегка измененные (adversarial) входные данные.

– Тестирование интерпретируемости (Explainability). Проверка модели что модель или вспомогательных методов на генерацию удовлетворительных объяснений предсказаний (если требуется).

– Регрессионное тестирование модели. Сравнение метрик новой версии модели с предыдущей на фиксированном валидационном наборе для выявления деградации («регрессии» модели).

6) Интеграционное и системное тестирование. Проверка корректного взаимодействия сервиса модели с другими компонентами системы, обработки ошибок, времени отклика, нагрузки.

Заключение. Адаптация классических принципов программной инженерии является необходимым условием для создания промышленных систем ИИ/МО. Предложенные адаптированные принципы формируют фундамент, на котором могут быть построены современные специализированные практики иных уровней для методов программной

инженерии применительно к системам искусственного интеллекта и машинного обучения.

Библиографический список:

1. Белоглазов, Д.А. Разработка автоматизированной системы управления зданием на основе искусственного интеллекта / Д.А. Белоглазов, В.Ю. Евтушенко, А.А. Пушнина, Е.Н. Жидченко // Проблемы автоматизации. региональное управление. Связь и автоматика - ПАРУСА-2015 : сборник трудов IV Всероссийской научной конференции молодых ученых, аспирантов и студентов : том 1, 2015. – Ростов-на-Дону : Южный федеральный университет, 2015. – С. 54-56.

2. ГОСТ Р 56939-2024. Защита информации. Разработка безопасного программного обеспечения. Общие требования : национальный стандарт Российской Федерации : утвержден и введен в действие Приказом Федерального агентства по техническому регулированию и метрологии от 24 октября 2024 г. №1504-ст : взамен ГОСТ Р 56939-2016 / разработан ФСТЭК России [и др.]. – Москва : Российский институт стандартизации, 2024. – С. 36.

3. ГОСТ Р ИСО/МЭК 12207-2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств : национальный стандарт Российской Федерации : утвержден и введен в действие Приказом Федерального агентства по техническому регулированию и метрологии от 30 ноября 2010 г. №631-ст : взамен ГОСТ Р ИСО/МЭК 12207-99 / подготовлен ФГУП «Научно-исследовательский институт «Восход». – Москва : Стандартинформ, 2011. – 105 с.

4. ГОСТ Р ИСО/МЭК 25010-2015. Информационные технологии. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов : национальный стандарт Российской Федерации : утвержден и введен в

действие Приказом Федерального агентства по техническому регулированию и метрологии от 29 мая 2015 г. №464-ст : введен впервые / подготовлен ООО «ИАВЦ». – Москва : Стандартинформ, 2015. – 36 с.

5. Донецков, А.М. Использование средств автоматизации в операционной складской деятельности на предприятии АО «ТБФ» / А.М. Донецков, В.М. Рожков // Проблемы и перспективы развития технических наук в условиях кризиса глобализации : сборник научных статей, 2024. – Ульяновск : «Зебра», 2024. – С. 56-61. – URL: https://elibrary.ru/download/elibrary_77127051_17474496.pdf (дата обращения 23.06.2025).

6. Кузнецов, П.П. Системы поддержки принятия врачебных решений на основе искусственного интеллекта – стратегия развития персонализированной медицины следующего этапа / П.П. Кузнецов, Е.П. Какорина, А.А. Алмазов // Терапевт. – 2020. – №1. – С. 48-53.

7. Мосолов, В.В. Автоматизация процесса обновления курсов валют в приложении для анализа и прогнозирования будущих трендов / В.В. Мосолов, А.Л. Ткаченко // Вестник Калужского университета. – 2024. – №1(62). – С. 29-35. – URL: https://elibrary.ru/download/elibrary_65511262_54642617.pdf (дата обращения 24.06.2025).

8. Сидоркин, А.Е. Исследование возможности автоматизации управленческого и производственного учёта в условиях импортозамещения / А.Е. Сидоркин, В.В. Сорочан, В.А. Раевский // Вестник калужского университета. – 2024. – №1(62). – С. 36-38. – URL: https://elibrary.ru/download/elibrary_65511263_74688546.pdf (дата обращения 25.06.2025).

9. Феоктистов, А.Г. Классификация масштабируемых программных комплексов / А.Г. Феоктистов, А.С. Корсуков, О.Ю. Башарина // Вестник Иркутского государственного технического университета. – 2017. – Т.21, №11(130). – С. 92-103.

10. Laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU)

2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act) : regulation (EU) 2024/1689 of the European Parliament and of the Council : 13 June 2024 // EUR-Lex.europa.eu : An Official Website of the European Union : [site]. – URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:L_202401689 (дата обращения 27.06.2025).

11. Sculley, D. Hidden Technical Debt in Machine Learning Systems / D. Sculley, G. Holt, D. Golovin [et al.] // Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS 2015). – 2015. – Vol. 2. – P. 2503–2511. – URL: https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf (дата обращения 26.06.2025).

Оригинальность 78%