УДК 004.896

А.Г. Атагулов, маг. (КарУ им. Е.А. Букетова, г. Караганда, Казахстан)

# СОВРЕМЕННЫЕ ПРОБЛЕМЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ И ПОТЕНЦИАЛЬНЫЕ РЕШЕНИЯ

Программная инженерия – это активно развивающаяся отрасль сферы IT. С увеличением требований для стабильности и масштабов программ и систем появляются новые проблемы архитектуры, которые будут рассмотрены в этой работе.

Микросервисный подход – подход, который стал стандартом для разработки крупных и сложных систем. Он предлагает разбиение монолитного приложения на набор небольших, автономных сервисов, которые взаимодействуют через API [1]. Выделяют основные проблемы микросервисной архитектуры:

1. Сложность управления: оркестрация большого числа микросервисов требует применения сложных инструментов, таких как Kubernetes.

2. Надёжность: отказ одного из микросервисов может привести к деградации или остановке всей системы, если не предусмотрены механизмы устойчивости.

3. Тестирование: интеграционное тестирование микросервисов становится более сложным из-за распределённой природы системы [2].

Событийно-ориентированная архитектура. Системы, построенные на основе событий, используют механизмы публикации и подписки для обмена данными, что позволяет уменьшить связность компонентов и повысить масштабируемость. Основными проблемами событийно-ориентированной архитектуры являются:

1. Отслеживание состояния: Сложно управлять состоянием системы,

2. Обеспечение согласованности: Модели согласованности данных, такие как eventual consistency, требуют продуманных решений для предотвращения ошибок [3].

Edge Computing и распределённые системы. С ростом IoT и 5G технологий распределённые вычисления стали важным аспектом современных архитектур. Обработка данных на периферии сети снижает задержки и нагрузку на центральные серверы. Проблемы использования распределённых систем:

1. Безопасность: Большое количество точек входа увеличивает уязвимость системы;

2. Обновления: Управление обновлениями на тысячах распределённых узлов остаётся серьёзной задачей [4].

Важно упомянуть, что существуют общие проблемы программной инженерии, которые являются актуальными:

– масштабируемость. С ростом числа пользователей и данных возникает необходимость в масштабируемости приложений. Проблемы включают: выбор между вертикальной и горизонтальной масштабируемостью, управление затратами на инфраструктуру [3];

– устойчивость программного обеспечения. Системы должны быть устойчивы к ошибкам и иметь механизмы самовосстановления. Внедрение таких механизмов увеличивает сложность разработки и требует глубокого понимания работы всей системы [4, 5];

– энергопотребление. Рост вычислительных мощностей привёл к увеличению энергопотребления. Это становится критичной проблемой для дата-центров, особенно с учётом необходимости экологической устойчивости;

– этические проблемы. С ростом применения AI в программной инженерии важным становится вопрос этики. Становятся актуальными проблемы обеспечения прозрачности и объяснимости решений, предотвращение дискриминации и предвзятости алгоритмов [5];

– исследования и инновации. Исследования в области программной инженерии вносят значительный вклад в развитие технологий. Например, разработка новых протоколов и методов передачи данных, таких как QUIC от Google, улучшает производительность сетей и снижает задержки. Развитие технологий баз данных, таких как NewSQL или системы распределённого хранения, позволяет достичь новых уровней масштабируемости и надёжности [6].

Для преодоления вышеупомянутых проблем программной инженерии используются следующие подходы [4]:

1. Автоматизация: Использование CI/CD, инфраструктуры как кода и тестирования для упрощения разработки;

2. Документирование: Введение стандартов описания систем и архитектурных решений;

3. Обучение специалистов: Регулярное повышение квалификации разработчиков и архитекторов.

Алгоритмизация и программирование остаются ключевыми составляющими успешной разработки ПО. Однако с развитием технологий появляются новые вызовы, которые требуют современных подходов и инновационных решений. Только гибкость, адаптивность и стремление к совершенствованию позволят справиться с актуальными проблемами программной инженерии.

ЛИТЕРАТУРА

1. Fowler M. Microservices: A Definition of this New Architectural Term". – URL: martinfowler.com.

2. Newman S. Building Microservices. – O'Reilly Media, 2021.
3. Kleppmann M. Designing Data-Intensive Applications. – O'Reilly Media, 2017.
4. Bass L., Clements P., Kazman R. Software Architecture in Practice. – Addison-Wesley, 2012.
5. Articles and case studies from IEEE Software and ACM Digital Library on software architecture trends and challenges.
6. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. – McGraw-Hill, 2011.

УДК 004.89

Б.Қ. Рахимжан, маг.
(КарУ им. Е.А. Букетова, г. Караганда, Казахстан)

## ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ ДЛЯ ПЕРСОНАЛИЗАЦИИ ОБУЧЕНИЯ

Современные системы образования пытаются адаптироваться к потребностям цифровой эпохи. Создание условий для персонализированного обучения, которое учитывает требования и особенности каждого ученика, является важной задачей. Адаптивное обучение, основанное на искусственном интеллекте, позволяет значительно улучшить этот процесс. Это позволяет учащимся получать задания, соответствующие их уровню знаний, и получать обратную связь в режиме реального времени.

ИИ может учитывать множество вещей, включая уровень подготовки ученика, предпочтения и когнитивные особенности, что позволяет создавать траектории обучения, адаптированные к их потребностям. Такой метод открывает новые горизонты, делая образование более доступным и эффективным и обеспечивая равные возможности для каждого ученика.

В этой статье рассматривается роль искусственного интеллекта в адаптивном обучении и возможности, которые он предоставляет для персонализации учебного процесса.

Искусственный интеллект - это одна из многих технологий, которые могут значительно изменить образовательный процесс. Машинное обучение (ML) может анализировать успеваемость учащихся и предлагать рекомендации, адаптированные к конкретным потребностям. Такие технологии могут помочь определить, какие задания требуют больше усилий для ученика, а затем предложить более сложные или простые задания, соответствующие его уровню. Это обеспечивает эф-