УДК 004.5

Взаимодействие различных категорий разработчиков для создания эргономичного интерфейса взаимодействия системы «человек - машина»

Ивченко Даниил Денисович ivchenkodd@student.bmstu.ru
Горячкин Борис Сергеевич bsgor@bmstu.ru

МГТУ им. Н.Э. Баумана, Москва, Россия

Аннотация. *В статье анализируется взаимодействие различных специалистов, участвующих в разработке эргономичных интерфейсов для систем «человек-машина». Особое внимание уделяется задачам и обязанностям дизайнеров, программистов (как фронтенд, так и бэкенд), тестировщиков и системных инженеров. Рассматриваются основные проблемы, возникающие в процессе их сотрудничества, и предлагаются подходы для их решения, включая использование современных технологий, таких как искусственный интеллект. Также обсуждаются возможности улучшения взаимодействия и повышения эффективности разработки за счет внедрения инновационных методов и метрик.*

Ключевые слова: *взаимодействие разработчиков, человеко-машинный интерфейс, эргономика интерфейсов, проблемы взаимодействия, искусственный интеллект*

Interaction of Different Categories of Developers for Creating an Ergonomic “Human - Machine” Interface

Ivchenko Daniil Denisovich ivchenkodd@student.bmstu.ru
Goryachkin Boris Sergeevich[1] bsgor@bmstu.ru

BMSTU, Moscow, Russia

Abstract. This article analyzes the interaction of various specialists involved in the development of ergonomic interfaces for “human-machine” systems. The focus is on the tasks and responsibilities of designers, programmers (both frontend and backend), testers, and system engineers. The main issues arising during their collaboration are examined, and solutions are proposed, including the use of modern technologies such as artificial intelligence. The article also discusses opportunities for improving interaction and enhancing development efficiency through the integration of innovative methods and metrics.

Keywords: developer interaction, human-machine interface, interface ergonomics, interaction problems, artificial intelligence

Введение. С развитием требований к пользовательским интерфейсам разработка эффективных и удобных человеко-машинных интерфейсов стала одной из центральных задач в работе разработчиков. Этот процесс требует слаженной работы специалистов разных профилей — дизайнеров, программистов,

тестировщиков и системных инженеров. Каждый из них решает свою уникальную задачу, но успешная реализация проекта возможна только при эффективном взаимодействии всех этих специалистов.

На практике между различными категориями разработчиков нередко возникают проблемы, связанные с различием в восприятии задач, подходах и методах работы. Эти трудности могут замедлить процесс разработки и снизить качество конечного продукта. Поэтому важно понимать, как улучшить взаимодействие между специалистами, что требует анализа их работы и выявления ключевых проблем.

Цель этой статьи — изучить взаимодействие разработчиков в процессе создания человеко-машинных интерфейсов, выявить основные проблемы и предложить возможные пути их решения. Особое внимание будет уделено использованию современных технологий, таких как искусственный интеллект, для улучшения процессов разработки и повышения качества интерфейсов.

Материалы и методы. Для разработки эргономичного интерфейса системы «человек-машина» критически важно эффективное взаимодействие различных категорий разработчиков. Дизайнеры, программисты, тестировщики и системные инженеры выполняют уникальные задачи, и их совместная работа определяет качество конечного продукта (рис. 1) [1].

Рис. 1. Схема взаимодействия различных категорий разработчиков

Дизайнеры интерфейсов ответственны за проектирование визуальных и функциональных аспектов взаимодействия пользователя с системой. Их задача — создать интуитивно понятные и удобные интерфейсы, используя методы прототипирования и тестирования для выявления и устранения проблем. В качестве основных инструментов применяются Figma, Adobe XD и многих других.

Категория программистов делится на два направления: фронтенд и бэкенд разработчики. Фронтенд специалисты реализуют визуальную часть интерфейса с помощью HTML, CSS и JavaScript, а также фреймворков, таких как React или Angular. Бэкенд-разработчики отвечают за серверную логику, взаимодействие с базами данных и поддержание стабильности системы.

Тестировщики занимаются поиском ошибок и дефектов интерфейса, а также в проведении юзабилити-тестирования для оценки удобства использования. Методы A/B тестирования активно применяются для оперативной оптимизации элементов интерфейса и взаимодействия с системой.

Системные инженеры автоматизируют процессы разработки и развертывания, внедряя CI/CD для повышения качества и скорости разработки. Они управляют инфраструктурой и обеспечивают интеграцию различных компонентов системы, тесно сотрудничая с программистами и тестировщиками для обеспечения стабильной работы системы.

Эффективное взаимодействие между различными категориями разработчиков начинается с четкого распределения ролей и синхронизации задач с помощью систем управления проектами, таких как Jira, Asana или Trello. Обмен данными между фронтенд и бэкенд разработчиками, тестировщиками и дизайнерами происходит через общие репозитории кода, например, Git и GitHub, что позволяет обеспечить прозрачность разработки и избежать дублирования усилий.

В рамках гибких методологий, таких как Scrum и Kanban, регулярная оценка прогресса с использованием ключевых метрик (время выполнения задач, количество открытых багов, количество интеграционных тестов) помогает оптимизировать рабочие процессы. Однако использование разных инструментов для разных категорий разработчиков (например, Figma для дизайнеров, React для фронтенда и специфические фреймворки для бэкенда), а также неопределенности на стадии планирования, создают сложности в координации. Эти проблемы в интеграции и коммуникации могут замедлить процесс разработки, требуя поиска эффективных методов их решения [2].

Результаты. В процессе разработки интерфейсов «человек-машина» продолжают оставаться несколько значимых проблем, которые влияют на эффективность работы команд. Одна из них — трудности коммуникации между разработчиками разных ролей. Несмотря на использование инструментов управления проектами, таких как Jira и Git, синхронизация между дизайнерами, программистами и тестировщиками зачастую оказывается затрудненной. Около 45 % команд сталкиваются с проблемами передачи требований, что увеличивает время разработки. Различия в рабочих средах, например, использование Figma для дизайнеров и React для фронтенда, также усложняют интеграцию компонентов системы [3].

Проблемы совместимости программного обеспечения, особенно на стыке фронтенда и бэкенда, продолжают задерживать проекты. Интеграция различных технологий требует значительных усилий, и примерно 38 % проектов сталкиваются с трудностями на этапе интеграции. Несвоевременное развертывание новых компонентов и недостаточная автоматизация процессов CI/CD также приводят к задержкам. Так, 27% задержек обусловлены недостаточной автоматизацией, а недостаточная автоматизация тестирования повышает затраты на исправление ошибок на 30%.

Существует множество методов решения этих проблем, некоторые из которых уже стали стандартом современной разработки, а другие применяются ситуативно. Эти решения можно разделить на методологические и программные.

Методологические решения включают гибкие подходы, такие как Agile и Scrum, которые способствуют улучшению коммуникации и синхронизации

работы между командами. Внедрение регулярных стендапов и ретроспектив позволяет оперативно устранять проблемы в процессе разработки, а методики управления рисками, такие как анализ причин и последствий, помогают глубже понять источник проблем и минимизировать их повторение. Практики, как Pair Programming и mob-программирование, способствуют повышению качества кода за счет постоянной обратной связи внутри команды.

Программные решения ориентированы на оптимизацию совместимости и тестирования компонентов системы. Использование контейнеризации с Docker и Kubernetes помогает обеспечить совместимость различных компонентов и упростить их интеграцию. Автоматизация тестирования с использованием фреймворков, таких как TestNG и Cypress, позволяет ускорить процесс тестирования и повысить надежность за счет автоматической проверки кода на каждом этапе. Инструменты для мониторинга, такие как Prometheus и Grafana, позволяют оперативно выявлять и устранять проблемы с производительностью и доступностью системы.

Хорошим примером эффективного применения этих методов являются крупные мировые компании. Например, в Netflix внедрение практик DevOps позволило значительно ускорить процесс развертывания, снизив время с нескольких часов до нескольких минут благодаря автоматизации интеграции и доставки [4]. В Microsoft использование GitHub Copilot повысило производительность разработчиков на 30-40%, снизив нагрузку за счет автоматизации рутинных задач [5]. В Figma внедрение функций для совместной работы, таких как комментарии и редактор в реальном времени, улучшило координацию между дизайнерами и разработчиками, ускорив внедрение изменений [6]. В Spotify внедрение Agile подходов позволило ускорить выпуск обновлений, обеспечив гибкость разработки и возможность быстрого реагирования на изменения благодаря коротким итерациям и регулярной обратной связи [7] (рис. 2).

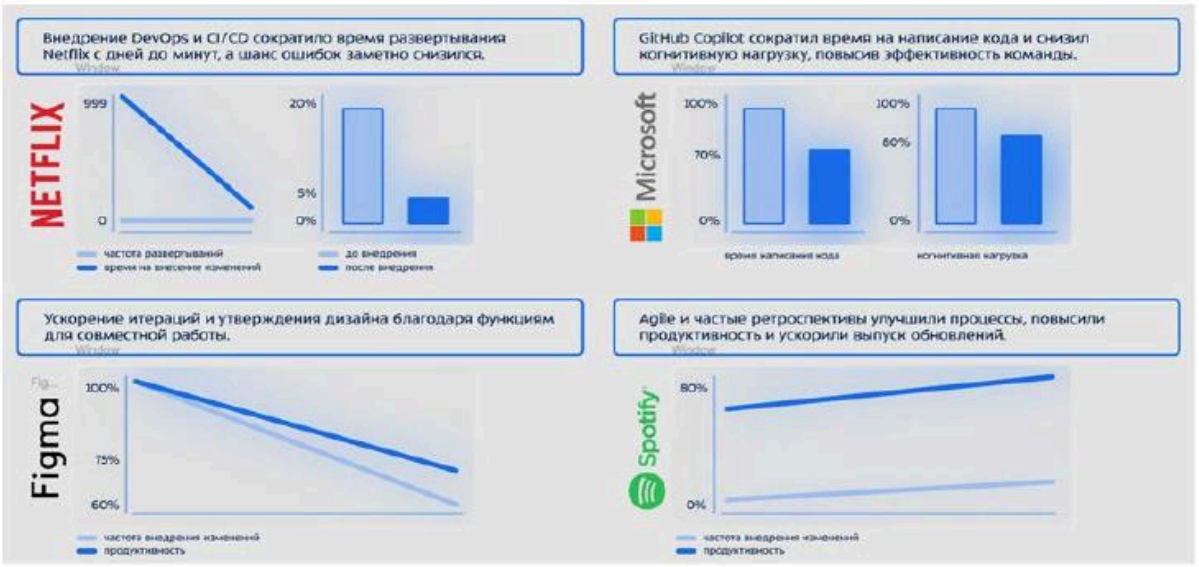

Рис. 2. Визуализация роста эффективности процессов при применении методов решения проблем

Обсуждение результатов. Для дальнейшего улучшения взаимодействия между разработчиками необходимо внедрение новых подходов, направленных на повышение гибкости и эффективности команд.

Одной из ключевых тенденций является развитие и внедрение адаптивных сред разработки. В отличие от традиционных статичных интеграционных платформ, такие среды будут динамично подстраиваться под потребности проекта, учитывая его уникальные характеристики. Интеграция таких сред, поддерживающих автоматическое масштабирование ресурсов и гибкую настройку под различные рабочие процессы, может значительно ускорить время разработки и снизить количество ошибок. Адаптивные среды разработки, которые могут автоматически настраиваться под командные и проектные задачи, становятся важным шагом к более гибким и персонализированным процессам [8].

Перспективной областью является интеграция машинного обучения и искусственного интеллекта для более умного управления проектами. Такие технологии смогут не только оптимизировать процесс тестирования и обеспечения качества, но и автоматизировать принятие решений в ходе разработки. Например, системы на основе ИИ смогут анализировать баги и ошибки, выделять наиболее важные задачи для разработчиков и предсказывать вероятные проблемы на основе предыдущих данных о проекте. Это приведет к сокращению времени на устранение дефектов и повышению общей производительности команд. Исследования показывают, что применение ИИ для оптимизации рабочего процесса может сократить время на решение задач на 20–30 % [9].

Продолжается развитие модернизированных инструментов для синхронизации работы мультидисциплинарных команд, в которых программисты, дизайнеры и тестировщики смогут более эффективно взаимодействовать. В таких инструментах процессы тестирования и проверки соответствия будут значительно ускорены. Использование визуальных интерфейсов и платформ для совместной работы, таких как Figma, а также аналогичных разработок, будет продолжать развиваться в направлении создания «умных» инструментов. Эти инструменты позволят ускорить создание проектов и снизить затраты на их реализацию [10].

Заключение. В ходе исследования взаимодействия различных категорий разработчиков при создании человеко-машинных интерфейсов было выявлено, что эффективное сотрудничество между дизайнерами, программистами, тестировщиками и системными инженерами является ключевым фактором успеха в разработке высококачественных интерфейсов. Полученные результаты показывают, что согласованная работа всех участников процесса разработки существенно влияет на конечный результат.

В рамках исследования были рассмотрены несколько решений для улучшения взаимодействия команд, включая внедрение адаптивных сред разработки, которые динамично подстраиваются под особенности проекта, а также использование искусственного интеллекта для оптимизации рабочих процес-

сов. Эти подходы могут значительно повысить гибкость разработки, ускорить процессы тестирования и автоматизировать многие аспекты работы.

Предложенные решения имеют потенциал стать основой для дальнейших исследований и практических шагов по улучшению разработки человеко-машинных интерфейсов, а также могут внести значительный вклад в развитие этой области в будущем.

Список источников

[1] Kotler I., James J. Problems of team interaction during interface development. Journal of Software Development, 2021, vol. 15, no. 3, pp. 123–134.

[2] Bertolino M., Lee J. Compatibility of software components and integration methods. International Journal of Software Engineering, 2019, vol. 28, no. 7, pp. 210–221.

[3] Smith A., Thompson R. Problems of testing and automation of development processes. Software Engineering Review, 2022, vol. 37, no. 9, pp. 215–225.

[4] Netflix. How Netflix built its in-house advertising technology platform. TechCrunch, 2024. May 15. URL: https://techcrunch.com/2024/05/15/netflix-in-house-advertising-technology-platform/ (accessed 09.12.2024).

[5] Microsoft. Drive efficient software development and boost DevEx with GitHub Copilot. Microsoft Azure Blog, 2023. November 15. URL: https://azure.microsoft.com/en-us/blog/drive-efficient-software-development-and-boost-devex-with-github-copilot/ (accessed 09.12.2024).

[6] Figma. How Figma builds software faster than most companies. FastCompany, 2020. July 13. URL: https://www.fastcompany.com/90548906/how-figma-builds-software-faster-than-most-companies (accessed 09.12.2024).

[7] Spotify. How Spotify scaled its agile framework to develop faster and react to change. Spotify Blog. URL: https://www.spotify.com/dk-da/agile-development/ (accessed 09.12.2024).

[8] Gonçalves S., Silva K. Flexible development environments and process automation. Journal of Adaptive Systems, 2020, vol. 24, no. 1, pp. 56–65.

[9] Liu C., Sun F. Application of artificial intelligence in optimizing development processes. Artificial Intelligence Review, 2020, vol. 32, no. 5, pp. 234–248.

[10] Bury K., Lee J. The role of collaborative tools in accelerating development. Journal of Software Engineering Practices, 2021, vol. 29, no. 2, pp. 121–130.