

УДК 004.5

Вопросы проектирования и создания эффективного пользовательского интерфейса

Ивченко Даниил Денисович^(*) danyaiv2010@gmail.com

Горячкин Борис Сергеевич bsgor@mail.ru

МГТУ им. Н.Э. Баумана, Москва, Россия

Аннотация. Рассмотрены вопросы проектирования и создания эффективных пользовательских интерфейсов. Особое внимание уделено роли взаимодействия различных категорий разработчиков, включая дизайнеров, программистов (фронтенд и бэкенд), тестировщиков и системных инженеров в данном процессе. Проанализированы проблемы, возникающие в процессе их взаимодействия при проектировании и разработке интерфейсов, а также предлагаются методы их решения, в том числе с использованием современных технологий, таких как искусственный интеллект. Рассмотрены перспективы повышения эффективности проектирования и разработки пользовательских интерфейсов за счет оптимизации взаимодействия специалистов и внедрения инновационных подходов.

Ключевые слова: взаимодействие разработчиков, пользовательский интерфейс, разработка пользовательского интерфейса, эргономика интерфейса, проблемы взаимодействия, искусственный интеллект

Economy of reusability of future cosmonautics and issues of efficiency

Ivchenko Daniil Denisovich^(*) danyaiv2010@gmail.com

Petrov Petr Petrovich bsgor@mail.ru

BMSTU, Moscow, Russia

Abstract. The article discusses the issues of designing and creating effective user interfaces. Particular attention is paid to the role of interaction between different categories of developers, including designers, programmers (frontend and backend), testers, and system engineers, in this process. The problems arising in the process of interaction during the design and development of interfaces are analyzed, and methods for solving them are proposed, including the use of modern technologies such as artificial intelligence. The prospects for increasing the efficiency of user interface design and development by optimizing the interaction of specialists and introducing innovative approaches are considered.

Keywords: developer interaction, user interface design, user interface development, interface ergonomics, interaction problems, artificial intelligence

Введение. В условиях постоянно растущих требований к пользовательским интерфейсам, проектирование и создание эффективных и удобных пользовательских интерфейсов становится одной из ключевых задач в современной

разработке программного обеспечения. Процесс создания таких интерфейсов требует согласованной работы специалистов различного профиля: дизайнеров, программистов, тестировщиков и системных инженеров. Каждый из них выполняет свои специфические задачи, однако успешная реализация проекта возможна только при условии эффективного взаимодействия между всеми участниками процесса. На практике между различными категориями разработчиков нередко возникают сложности, обусловленные различиями в понимании задач, подходах и методах работы. Эти сложности могут приводить к замедлению процесса разработки и негативно сказываться на качестве конечного продукта. Таким образом, крайне важно определить пути оптимизации взаимодействия между специалистами, что предполагает анализ их деятельности и выявление типичных проблем, возникающих в процессе взаимодействия.

Цель данной статьи — исследование взаимодействия разработчиков в контексте проектирования и создания пользовательских интерфейсов, выявление основных проблем, возникающих в процессе работы, и рассмотрение методов их решения. В статье также рассматривается применение современных технологий, таких как искусственный интеллект, для оптимизации процессов проектирования и разработки интерфейсов, а также повышения их качества.

Анализ взаимодействия разработчиков. Эффективное взаимодействие между различными категориями разработчиков играет ключевую роль в проектировании и создании эргономичных пользовательских интерфейсов. Дизайнеры, программисты, тестировщики и системные инженеры выполняют уникальные задачи, и их совместная работа определяет качество конечного продукта (рис. 1) [1].

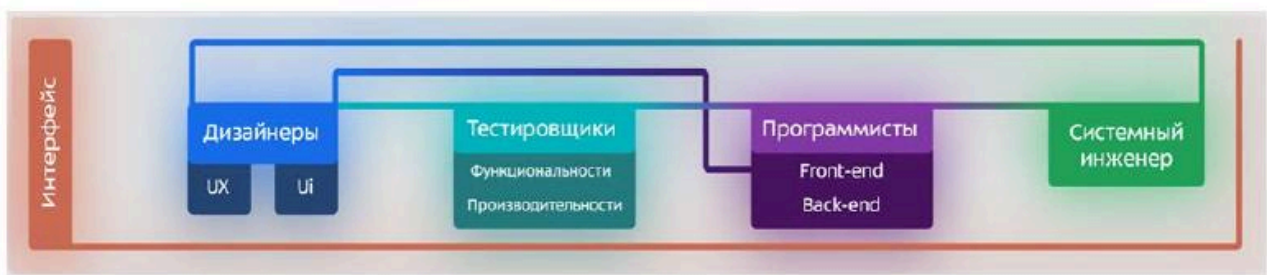


Рис. 1. Схема взаимодействия различных категорий разработчиков

Дизайнеры интерфейсов занимаются проектированием визуальных и функциональных аспектов взаимодействия пользователя с системой. Они создают интуитивно понятные и эргономичные интерфейсы, используя прототипирование и тестирование для выявления и устранения проблем. В качестве основных инструментов они используют Figma, Adobe XD и др.

Программисты, в свою очередь, делятся на два основных направления: фронтенд-разработчики и бэкенд-разработчики. Фронтенд-разработчики реализуют визуальную часть интерфейса с использованием HTML, CSS

и JavaScript, а также платформ разработки, таких как React или Angular. Бэк-енд-разработчики занимаются серверной логикой, взаимодействием с базами данных и поддержанием стабильности системы.

Тестировщики занимаются выявлением ошибок и дефектов интерфейсов, а также проводят тестирование удобства использования для оценки эргономичности взаимодействия. Применение методов А/В-тестирования позволяет оперативно оптимизировать элементы интерфейса и взаимодействие с системой.

Системные инженеры автоматизируют процессы разработки и развертывания, внедряя практики непрерывной интеграции и непрерывной поставки (CI/CD) для повышения качества и скорости разработки. Они управляют инфраструктурой и обеспечивают интеграцию различных компонентов системы, сотрудничая с программистами и тестировщиками для обеспечения стабильности работы системы.

Взаимодействие между различными категориями разработчиков при создании пользовательских интерфейсов начинается с четкого распределения ролей и синхронизации задач с использованием систем управления проектами, таких как Jira, Asana или Trello. Обмен данными между фронтенд- и бэк-енд-разработчиками, тестировщиками и дизайнерами осуществляется через общие репозитории кода, например Git и GitHub, что обеспечивает прозрачность процесса разработки и предотвращает дублирование усилий. В рамках гибких методологий, таких как Scrum и Kanban, регулярная оценка прогресса с использованием ключевых метрик, таких как время выполнения задач, количество обнаруженных ошибок и результаты интеграционных тестов, позволяет оптимизировать рабочие процессы. Однако использование различных инструментов для разных категорий разработчиков, таких как Figma для дизайнеров, React для фронтенда, и специфических платформ разработки для бэкэнда, а также неопределенности на стадии планирования, создают сложности в координации. Эти трудности в интеграции и коммуникации могут существенно замедлить процесс разработки и потребовать поиска эффективных методов их решения [2].

Проблемы взаимодействия и методы их решения. В процессе проектирования и создания эффективных пользовательских интерфейсов сохраняется ряд значимых проблем, которые продолжают оказывать влияние на эффективность работы команд. Одной из таких проблем является сложность коммуникации между разработчиками различного профиля, что напрямую влияет на качество проектирования и конечный вид пользовательского интерфейса. Несмотря на использование средств управления проектами, таких как Jira и Git, синхронизация между дизайнерами, программистами и тестировщиками зачастую затруднена. Примерно 45 % команд сталкиваются с проблемами передачи требований, что приводит к ошибкам в проектировании и увеличению времени, затрачиваемого на разработку интерфейса. Различия в средах разработки, например Figma для дизайнеров и React для фронтенд-разработчиков, также осложняют интеграцию компонентов систе-

мы, что негативно сказывается на целостности и функциональности пользовательского интерфейса [3].

Проблемы совместимости программного обеспечения, особенно на стыке фронтенд- и бэкенд-разработки, также продолжают приводить к задержкам проектов, что в конечном итоге отражается на сроках создания пользовательского интерфейса. Интеграция различных технологий требует значительных усилий, и около 38 % проектов испытывают сложности на этапе интеграции, что может привести к несоответствию интерфейса первоначальным проектным решениям. Наряду с этим, несвоевременное развертывание новых компонентов и недостаточная автоматизация процессов непрерывной интеграции и непрерывной поставки (CI/CD) часто приводят к задержкам, что затягивает процесс создания пользовательского интерфейса и его вывод на рынок. Недостаточная автоматизация тестирования остается еще одной проблемой, увеличивая затраты на исправление ошибок, что также влияет на сроки и стоимость разработки пользовательского интерфейса [3].

Существует множество методов решения этих проблем, некоторые из них стали неотъемлемой частью современной разработки, такие как практики DevOps, а некоторые применяются ситуативно. Фактически, решения можно разделить на методологические и программные.

Методологические решения включают в себя гибкие подходы, такие как Agile и Scrum, которые способствуют улучшению коммуникации и синхронизации работы между командами, что положительно сказывается на процессе проектирования и создания пользовательского интерфейса. Внедрение регулярных собраний 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, что способствует созданию более качественного и эффективного пользовательского интерфейса. Применение таких практик, как парное программирование и коллективное программирование, повышает качество кода за счет постоянной обратной связи внутри команды, что также положительно влияет на надежность и функциональность пользовательского интерфейса.

Программные решения ориентированы на оптимизацию совместимости и тестирования компонентов системы, что напрямую влияет на качество и стабильность пользовательского интерфейса. Использование контейнеризации с Docker и Kubernetes помогает обеспечить совместимость различных компонентов и упростить их интеграцию, что облегчает процесс создания целостного и эффективного пользовательского интерфейса. Автоматизация тестирования с применением платформ, таких как TestNG и Cypress, позволяет ускорить процесс тестирования и повысить надежность за счет автоматической проверки кода на каждом этапе, что способствует созданию более качественного и безошибочного пользовательского интерфейса. Средства для мониторинга Prometheus и Grafana позволяют оперативно выявлять и устранять проблемы с производительностью и доступностью системы, что обеспечивает стабильную и эффективную работу пользовательского интерфейса.

Хорошим примером служат крупные, всемирно известные компании, которые активно применяют вышеперечисленные методы. Например, в Netflix внедрение практик DevOps позволило значительно ускорить процесс развертывания, снизив время с нескольких часов до нескольких минут благодаря автоматизации интеграции и доставки, что положительно сказалось на скорости обновления и улучшения пользовательского интерфейса [4]. В Microsoft использование GitHub Copilot снизило нагрузку на разработчиков, повысив их производительность на 30–40 % [5]. Этот инструмент, основанный на ИИ, помогает ускорить написание кода и автоматизирует рутинные задачи, что также способствует более быстрому и эффективному созданию пользовательских интерфейсов. В Figma внедрение функций для совместной работы, таких как комментарии и редактор в реальном времени, позволило ускорить внедрение изменений и улучшить координацию между дизайнерами и разработчиками, что привело к созданию более удобных и функциональных пользовательских интерфейсов [6]. Эти улучшения стали возможны благодаря единой платформе, интегрирующей все процессы. В свою очередь, в Spotify внедрение подходов Agile позволило ускорить выпуск обновлений, обеспечив гибкость разработки и возможность быстрого реагирования на изменения благодаря коротким итерациям и регулярной обратной связи, что позволило компании оперативно адаптировать пользовательский интерфейс к потребностям пользователей и изменениям рынка (рис. 2) [7].

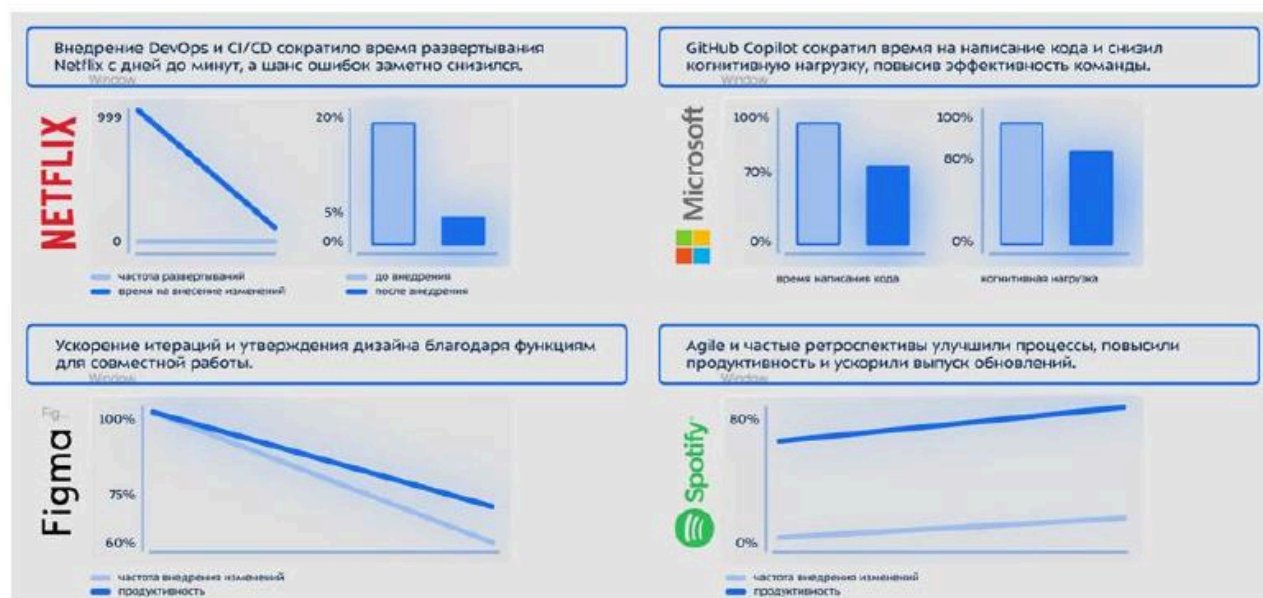


Рис. 2. Визуализация роста эффективности процессов при применении методов решения проблем

Перспективы развития методов решения проблем взаимодействия. Для достижения большей эффективности в проектировании и создании пользовательских интерфейсов необходимы новые подходы, которые обеспечат не только гибкость работы команд, но и высокое качество конечного продук-

та. В этом контексте, развитие адаптивных сред разработки становится ключевой тенденцией. Если традиционные платформы интеграции остаются статичными, то такие среды смогут динамически подстраиваться под потребности конкретного проекта, учитывая его уникальные характеристики. Интеграция таких сред, поддерживающих автоматическое масштабирование ресурсов и гибкую настройку под разные рабочие процессы, может значительно ускорить время разработки и снизить количество ошибок. Например, помимо общего упрощения разработки, которое достигается, в частности, за счет low-code платформ, адаптивные среды могут быть использованы для реализации интеллектуальной персонализации интерфейсов. Представьте себе интерфейсы, которые автоматически адаптируются под индивидуальные потребности и предпочтения пользователей, основываясь на контекстных данных, таких как местоположение, время суток или история взаимодействия. Адаптивные среды разработки, с возможностью автоматической настройки под командные и проектные задачи, становятся важным шагом к более гибким и персонализированным процессам [8].

Параллельно с этим, огромный потенциал имеет интеграция машинного обучения и искусственного интеллекта для интеллектуального управления проектами на всех этапах проектирования и создания пользовательских интерфейсов. ИИ может не только оптимизировать процесс тестирования и обеспечения качества, но и автоматизировать принятие решений в процессе разработки. Системы, основанные на ИИ, смогут анализировать дефекты в интерфейсах, выделять наиболее важные задачи для разработчиков и даже предсказывать возможные проблемы, опираясь на предыдущие данные о проекте. Применение ИИ для оптимизации рабочего процесса может сократить время на решение задач на 20–30 % [9]. И если генеративный дизайн уже находит применение, то в будущем ИИ может быть использован для создания «эмпатичных интерфейсов». Такие интерфейсы будут автоматически адаптироваться к эмоциональному состоянию пользователя, анализируя биометрические данные, такие как мимика или пульс, что позволит сделать их не только функциональными, но и более интуитивными и приятными в использовании.

Наконец, важную роль в повышении эффективности разработки пользовательских интерфейсов играет развитие средств синхронизации работы мультидисциплинарных команд. Это направлено на ускорение процессов проектирования, разработки и тестирования. Здесь помимо развития существующих инструментов, таких как инструменты для дизайн-систем, актуальным направлением является создание иммерсивных сред для совместной работы. Представьте себе VR/AR-платформы, позволяющие командам проектировать и тестировать интерфейсы в трехмерном пространстве. Такой подход может значительно улучшить понимание будущего продукта и взаимодействие с ним на ранних этапах разработки [10].

Заключение. В данной работе было проведено исследование вопросов проектирования и создания эффективных пользовательских интерфейсов.

Полученные результаты подтверждают, что эффективное взаимодействие между дизайнерами, программистами, тестировщиками и системными инженерами играет ключевую роль в успешном создании высококачественных интерфейсов. В работе предложены решения, направленные на оптимизацию взаимодействия команд, включая развитие адаптивных сред разработки и использование искусственного интеллекта для оптимизации рабочих процессов. Эти подходы могут стать основой для дальнейших исследований и практических улучшений в области проектирования и создания пользовательских интерфейсов.

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. URL: <https://techcrunch.com/2024/05/15/netflix-in-house-advertising-technology-platform> (accessed March 10, 2025).
- [5] Microsoft. Drive efficient software development and boost DevEx with GitHub Copilot // Microsoft Azure Blog. URL: <https://azure.microsoft.com/en-us/blog/drive-efficient-software-development-and-boost-devex-with-github-copilot> (accessed March 20, 2025).
- [6] Figma. How Figma builds software faster than most companies. URL: <https://www.fastcompany.com/90548906/how-figma-builds-software-faster-than-most-companies> (accessed March 20, 2025).
- [7] Spotify. How Spotify scaled its agile framework to develop faster and react to change. URL: <https://www.spotify.com/dk-da/agile-development/> (accessed March 21, 2025).
- [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.