Известия Гомельского государственного университета
имени Ф. Скорины, № 6 (141), 2023

УДК 004.9 EDN: LFCEPV

Анализ системного дизайна и архитектуры современных приложений

Ю.А. Тарасова

Системная часть является основой успешного и эффективного использования программного обеспечения. При этом важно отметить относительную новизну такого понятия, как системный дизайн. В связи с этим актуализируются задачи, связанные с созданием наиболее оптимального системного дизайна и архитектур современных приложений. Цель представленной статьи заключается в анализе наиболее эффективных инструментов разработки системной части программного обеспечения. Научная ценность работы состоит в предпринимаемой попытке систематизации знаний относительно представленных вопросов. Материалы статьи могут стать полезны для современных разработчиков, предоставляя комплексную информацию, которая поможет при выборе инструментов и способов реализации системного дизайна и архитектуры программного обеспечения.
Ключевые слова: системный дизайн, архитектура приложения, программное обеспечение, разработка приложений.

The system part is the basis for successful and efficient use of software. At the same time, it is important to note the relative novelty of such a concept as system design. In this regard, the tasks related to the creation of the most optimal system design and architectures of modern applications are being update. The purpose of the presented article is to analyze the most effective tools for developing the system part of the software. The scientific value of the work consists in an attempt to systematize the knowledge about the issues presented. The materials of the article can be useful for modern developers, providing comprehensive information that will help in choosing tools and ways to implement system design and software architecture.
Keywords: system design, application architecture, software, application development.

Введение. Системный дизайн представляет собой процесс определения архитектуры, интерфейсов и данных для системы, которая должна удовлетворять определенным требованиям. Именно системный дизайн включает в себя задачи, связанные с обдумыванием инфраструктуры, данных и способов их хранения. Проектирование информационной системы с использованием данных аспектов позволяет определить наиболее эффективную и удовлетворяющую всем бизнес-требованиям платформу [1].

Важно отметить, что системный дизайн – это комплексное понятие. При разработке программного обеспечения с целью получения наиболее оптимального и эффективного итогового решения важно и необходимо использование принципов системного дизайна. Системный дизайн при разработке современных приложений подразумевает решение нескольких последовательных задач, направленных на поиск и формирования наиболее оптимальных решений конечного продукта.

Вместе с этим стоит подчеркнуть, что это относительно новое понятие и его интерпретация в различных источниках может быть разной. Автором в рамках текущей работы преследуется цель, связанная с комплексным анализом данного вопроса и формированием методологического аппарата, отражающего анализ системного дизайна и архитектуры современного программного обеспечения.

Результаты и обсуждение. Проектирование системы представляет собой фазу, которая устраняет разрыв между проблемной областью и существующей системой управляемым образом. На этом этапе основное внимание уделяется области решения, то есть, вопросу: «как реализовать?». Это фаза, когда документ SRS (software requirements specification – набор требований к программному обеспечению) преобразуется в формат, который может быть реализован, и решает, как будет работать система. На этом этапе сложная деятельность по разработке системы делится на несколько мелких подвидов деятельности, которые координируются друг с другом для достижения основной цели разработки системы [2].

Системный дизайн подразумевает необходимость для всех участников процесса работы над программным обеспечением прийти к единому пониманию перед началом процесса разра-

ботки. Одним из наиболее распространенных результатов работы над системным дизайном является диаграмма, описывающая ключевые части конечного продукта и их взаимодействие друг с другом. Так, хороший результат над системным дизайном должен выявить такие аспекты, как:

– составные части (отвечает на вопрос «какие основные элементы и объекты есть в системе?»);

– взаимосвязи (отвечает на вопросы: «как соединены элементы?»; «какие существуют взаимосвязи?»; «где располагаются входы и выходы?»);

– цель конечного продукта (отвечает на вопрос «что нужно достичь?») [3].

Работа над данными задачами должна производиться в командном режиме с использованием специализированного программного обеспечения. При этом до недавнего времени наибольшей популярностью пользовалась обычная магнитная доска. Однако в результате последующей разработки инструментов выбор перешел в сторону специальных программ, позволяющих автоматизировать взаимодействие между всеми участниками процесса разработки. Примером одного из таких и наиболее популярным инструментом является Miro. Конечным результатом является четкая диаграмма, иллюстрирующая объекты и отношения между ними. На рисунке 1 представлен пример итога разработки системного дизайна посредством Miro [4].

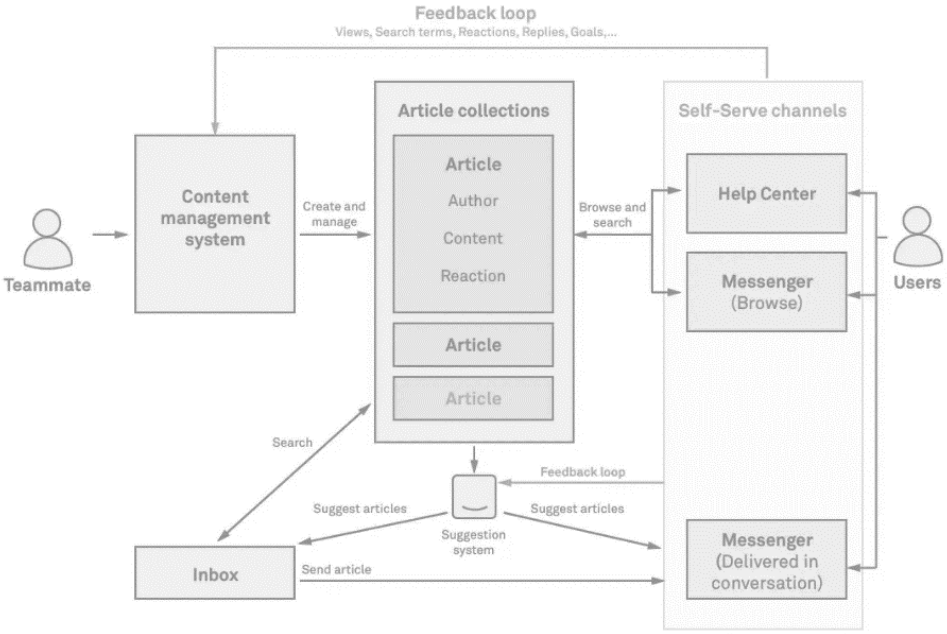

Рисунок 1 – Пример работы над системным дизайном в Miro

Главной особенностью использования цифровых инструментов создания системного дизайна является возможность перенастройки и изменения. Зачастую не только на стадии проектирования, но при разработке возникают задачи, подразумевающие расширение функционала программы и ее возможностей. Благодаря использованию специализированных инструментов, как Miro, разработчики получают возможность быстрого внесения изменений в исходный проект программного обеспечения.

Также важно отметить, что системный дизайн при разработке приложений играет ключевую роль в обеспечении их эффективной работы и удовлетворении потребностей пользователей. Он охватывает широкий спектр аспектов, начиная от архитектурных решений и взаимодействия компонентов приложения до управления данными, производительности и безопасности. На рисунке 2 представлены основные компоненты, определяемые и вносимые в приложение на этапе работы над системным дизайном [5].

На этапе системного дизайна разработчики определяют структуру приложения, включая его модули, интерфейсы и взаимосвязи между ними. Важно учесть масштабируемость системы, чтобы она могла расти и развиваться с увеличением нагрузки и объема данных. Архитектурные решения также включают выбор технологий и платформ, которые поддерживают нужные функциональности.

Системный дизайн охватывает и вопросы управления данными, включая выбор баз данных, их оптимизацию и обеспечение целостности информации. Он также включает в себя стратегии обработки ошибок, обеспечение безопасности данных и пользовательской конфиденциальности. Все эти аспекты вместе обеспечивают надежность, производительность и удобство использования разрабатываемого приложения.

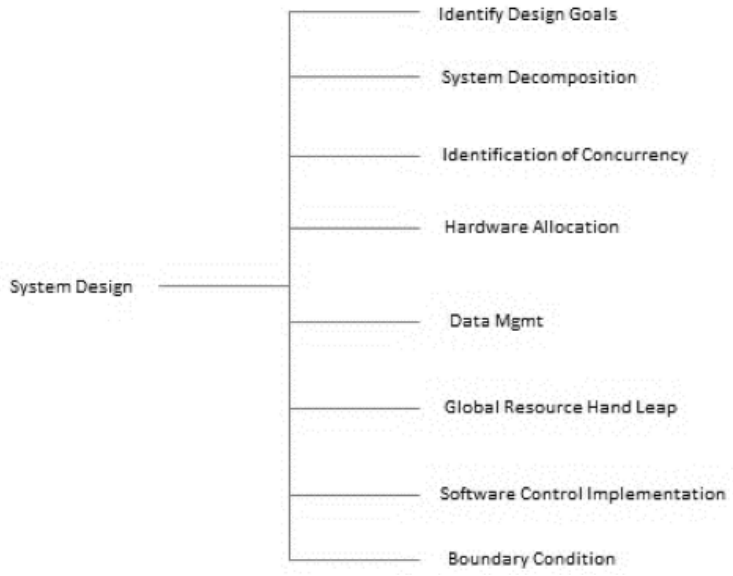

Рисунок 2 – Компоненты, определяемые на этапе системного дизайна

Важно отметить и взаимосвязь между системным дизайном и архитектурой приложения. Системный дизайн и архитектура приложения тесно взаимозависимы. Системный дизайн определяет общую структуру и организацию приложения, включая его компоненты, модули, интерфейсы и взаимосвязи между ними. Он управляет тем, как различные части приложения взаимодействуют друг с другом и с внешними системами.

Архитектура приложения, с другой стороны, фокусируется на высокоуровневом планировании и организации системы в целом. Это включает выбор подходящих технологий, платформ, архитектурных шаблонов и паттернов для реализации функциональности, определенной в системном дизайне. Архитектурные решения влияют на производительность, масштабируемость, безопасность и общую эффективность приложения [6].

Таким образом, системный дизайн определяет, как разные компоненты приложения должны быть связаны и как они будут взаимодействовать, в то время как архитектура приложения предоставляет инструкции о том, как эти компоненты будут построены и какие технологии будут использоваться для их реализации. Когда эффективно сочетаются, системный дизайн и архитектура приложения помогают создать устойчивые, гибкие и эффективные программные решения, которые соответствуют требованиям проекта и удовлетворяют потребности пользователей.

Системные дизайны программ могут иметь различные архитектурные подходы в зависимости от конкретных требований, целей и характеристик проекта. Далее представлены некоторые из наиболее популярных и часто используемых типов системных дизайнов:

– монолитная архитектура. В этом подходе весь функционал приложения находится в одном монолитном блоке кода. Это простой способ разработки, но может вызвать проблемы масштабируемости и поддержки при росте проекта;

– архитектура клиент-сервер. Приложение разделяется на две части: клиентскую (интерфейс пользователя) и серверную (обработка данных и бизнес-логика). Этот подход позволяет легче масштабировать и обеспечивает более гибкий контроль над данными;

– архитектура микро-сервисов. Приложение разделяется на небольшие независимые сервисы, каждый из которых выполняет конкретные функции. Это способствует гибкости, масштабируемости и легкости обновления, но требует управления множеством компонентов;

– событийно-ориентированная архитектура. Приложение реагирует на события и сообщения, которые генерируются различными компонентами или внешними системами. Это позволяет создавать высоко отзывчивые и расширяемые системы;

– архитектура потоков данных. В этом подходе данные перемещаются через различные этапы обработки в виде потоков. Это полезно для обработки больших объемов данных в реальном времени;

– архитектура с участием контейнеров и оркестраторов. Использование контейнеров (например, Docker) и оркестраторов (например, Kubernetes) позволяет управлять развертыванием и масштабированием приложений с высокой степенью автоматизации [7].

Каждая из этих архитектур имеет особенности и степени актуальности своего использования при решении той или иной задачи. В таблице 1 отражены результаты анализа, по которому выделены ключевые преимущества и недостатки по архитектурам современных приложений.

Таблица 1 – Анализ архитектур современных приложений

Архитектура	Преимущества	Недостатки
Монолитная архитектура	Простота разработки и тестирования. Единое место для управления и обслуживания.	Ограниченная масштабируемость. Трудности в разделении ответственности между разработчиками. Большие изменения могут повлиять на всю систему.
Архитектура клиент-сервер	Разделение клиентской и серверной частей обеспечивает гибкость. Легче обновлять и масштабировать серверную часть.	Сложнее управление двумя отдельными компонентами. Требует более сложной сетевой инфраструктуры.
Архитектура микро-сервисов	Высокая масштабируемость и гибкость. Отдельные сервисы могут быть разработаны и масштабированы независимо.	Управление множеством сервисов может быть сложным. Сложнее обеспечить связь данных между сервисами.
Событийно-ориентированная архитектура	Высокая отзывчивость и расширяемость. Легкость интеграции с внешними системами.	Сложнее отслеживать и управлять потоками событий. Требует внимательного управления событиями и обработки ошибок.
Архитектура потоков данных	Обработка данных в реальном времени. Подходит для аналитики и потоковой обработки.	Сложнее в реализации и отладке. Требует специализированных инструментов.
Архитектура с участием контейнеров и оркестраторов	Упрощает развертывание и масштабирование приложений. Повышает надежность и отказоустойчивость.	Требует знания и опыта в управлении контейнерами и оркестраторами. Может быть избыточным для небольших проектов.

Каждый из представленных типов архитектур имеет свои преимущества и недостатки. При этом выбор зависит от конкретных потребностей проекта, его масштаба и целей разработки. Также важно отметить, что зачастую в больших и сложных приложениях используется комбинация нескольких архитектурных подходов [8].

Заключение. Таким образом, основной целью представленной статьи являлось выполнение анализа по вопросам системного дизайна и архитектур современных приложений. В результате работы актуализирована необходимость детальной проработки будущей программы на этапе системного дизайна. В работе представлен пример работы над системным дизайном с использованием специализированного инструмента, рассмотрены компоненты, задачи и связи приложения, определяемые на этапе системного дизайна. Также в рамках статьи представлены результаты анализа архитектур современных приложений в таких отношениях, как преимущества и недостатки каждой из них.

В заключение необходимо отметить, что именно посредством системного дизайна и соответствующих специальных инструментов представляется возможность создания наиболее устойчивых, масштабируемых и эффективных приложений, способных удовлетворить потребности современных пользователей. Этот подход помогает учесть комплексность взаимо-

действия компонентов приложения и обеспечить его согласованность и удобство использования. Из всего вышеописанного следует, что системный дизайн является ключевой и неотъемлемой частью при проектировании и разработке современных приложений.

Литература

1. Пащенко, Д. С. Как мировые тенденции в проектировании информационных систем используются в отечественной практике: результаты исследования / Д. С. Пащенко // Инновации. – 2018. – № 1 (231). – С. 58–63.

2. Караванов, А. В. Архитектура программного обеспечения для высоконадежных систем / А. В. Караванов, Н. Д. Иванов // Космические аппараты и технологии. – 2018. – № 2 (24). – С. 100–104.

3. Парамзина, А. А. Использование case-инструментов при разработке программного обеспечения / А. А. Парамзина, Е. Н. Тищенко // Экономика и социум. – 2022. – № 1-2 (92). – С. 118–122.

4. Шайтура, С. В. Системный анализ технологий компьютерных систем и систем связи / С. В. Шайтура, М. А. Жиделев, Д. Ю. Федоров // Известия ТулГУ. Технич. науки. – 2023. – № 3. – С. 290–296.

5. Игнатьева, О. В. Архитектурные приемы при разработке программного обеспечения, зависимого от интерфейса пользователя / О. В. Игнатьева, Н. А. Москат, С. С. Рубцова // ИВД. – 2022. – № 2 (86). – С. 25–33.

6. Калугян, К.Х. Моделирование процессов контроля качества программного обеспечения / К. Х. Калугян, К. Н. Щербакова, М. В. Глущенко // SAEC. – 2020. – № 3. – С. 308–314.

7. Кокурин, Е. А. Исследование и разработка архитектуры информационной системы анализа использования программного обеспечения предприятия / Е. А. Кокурин // Международный журнал гуманитарных и естественных наук. – 2022. – № 11-2. – С. 103–107.

8. Яровая, Е. В. Принципы построения архитектуры программного обеспечения / Е. В. Яровая // E-Scio. – 2022. – № 8 (71). – С. 20–24.

ООО «Антифишинг», Россия Поступила в редакцию 02.10.2023