УДК 004.9 **DOI** 10.15622/ia.21.6.7

А.Д. Обухов, А.А. Волков, А.О. Назарова

МИКРОСЕРВИСНАЯ АРХИТЕКТУРА ВИРТУАЛЬНЫХ ТРЕНАЖЕРНЫХ КОМПЛЕКСОВ

Обухов А.Д., Волков А.А., Назарова А.О. **Микросервисная архитектура виртуальных тренажерных комплексов.f**

Аннотация. В представленной работе рассматривается задача автоматизации и снижения сложности процесса разработки виртуальных тренажерных комплексов. Проведенный анализ предметной области показал необходимость перехода от монолитного подхода к сервис-ориентированному варианту архитектуры. Выявлено, что использование монолитной архитектуры при реализации виртуальных тренажерных комплексов ограничивает возможность модернизации системы, увеличивает ее программную сложность, затрудняет реализацию интерфейса для управления и мониторинга процесса подготовки. Представлена общая концепция микросервисной архитектуры виртуальных тренажерных комплексов, даны определения основных и второстепенных компонентов. Научная новизна исследования заключается в переходе от классической монолитной архитектуры в предметной области ВТК к микросервисной архитектуре и устранении недостатков данного подхода за счет реализации единого протокола обмена информацией между модулями и отделения процедур сетевого взаимодействия в программные библиотеки в каждом микросервисе для унификации и повышения надежности работы системы. Применение изолированных, слабо связанных микросервисов позволяет использовать оптимальные технологии, платформы и фреймворки для их реализации, отделить графический интерфейс инструктора тренажера от системы визуализации и виртуальной реальности, обеспечить возможность гибкой замены основных компонентов (визуализации, интерфейса, взаимодействия с виртуальной реальностью) без изменения архитектуры и влияния на остальные модули. Осуществлена декомпозиция структурной модели микросервисной архитектуры, представлена специфика функционирования основных компонентов. Рассмотрена реализация библиотек сетевого взаимодействия микросервисов и протокола обмена данных на основе JSON. Практическая значимость предложенной архитектуры состоит в возможности распараллеливания и снижения сложности процесса разработки и модернизации тренажерных комплексов. Проанализированы особенности функционирования систем, реализованных на предложенной микросервисной архитектуре.

Ключевые слова: микросервисная архитектура, микросервисы, виртуальные тренажерные комплексы, межмодульное взаимодействие, оптимизация передачи данных.

1. Введение. Виртуальные тренажерные комплексы (ВТК) включают широкий класс программных, аппаратных и программно-аппаратных систем, направленных на решение задач профессиональной подготовки [1], реабилитации (физической и психологической) [2, 3] и диагностики [4, 5]. Например, ВТК успешно применяются при подготовке шахтеров [6], так как позволяют смоделировать в виртуальной реальности штатные процессы трудовой деятельности и аварийные ситуации (пожаротушение, эвакуация после

затопления, обрушение, задымление). Современные ВТК основываются на передовых технологиях визуализации, сбора, анализа и обработки информации, включая следующие:

– технологии дополненной (AR) и виртуальной (VR) реальности для погружения пользователя в цифровое пространство [7];

– нейронные сети и машинное обучение для решения задач анализа, обработки и управления [8];

– компьютерное зрение для отслеживания состояния окружающей среды, объектов наблюдения, пользователя, процессов, протекающих в предметной области [9];

– имитационное оборудование для моделирования воздействия на органы чувств человека, создания физических нагрузок [10];

– медицинское оборудование для отслеживания текущего состояния пользователя [11].

При разработке ВТК с реализацией вышеперечисленных технологий, интеграцией дополнительных модулей и оборудования неизбежно возрастет программная сложность системы и возникает потребность в привлечении большого количества человеческих ресурсов [12]. Реализация ВТК как монолитного проекта имеет определенные преимущества для небольших команд разработчиков и систем с ограниченным количеством связей между модулями. С другой стороны, высокая связность компонентов ограничивает модернизацию и внедрение новых технологий, добавление, удаление или изменение модулей. Со временем общая программная база расширяется и становится громоздкой, а структура слишком тяжелой для понимания, что приводит к сложности поддержки [13]. Кроме того, существуют значительные сложности с реализацией интерфейса для инструктора тренажерного комплекса. Этот модуль должен быть реализован отдельно от системы визуализации, но постоянно обмениваться с ней информацией, что даже в условиях монолитной архитектуры приводит к реализации нескольких связанных модулей или даже отдельных проектов.

Таким образом, для устранения обозначенных недостатков при реализации ВТК, обладающих высокой сложностью, необходимо осуществить переход к новой архитектуре – сервис-ориентированной, в которой система разделяется на множество отдельных модулей (сервисов), слабо связанных между собой стандартизированными протоколами, легко заменяемых, решающих отдельные конкретные задачи [14]. Каждый модуль самодостаточен и независим, поэтому можно изменять фрагменты ВТК, не затрагивая остальные элементы

системы, распределять сервисы по разным серверам, реализовывать их на разных языках программирования и с применением различных технологий.

Среди сервис-ориентированных архитектур можно выделить направление микросервисных архитектур [15, 16], где приложение состоит из множества небольших сервисов, взаимодействующих между собой (чаще всего по протоколу HTTP). Каждый микросервис самодостаточен и независим, поэтому можно изменять только фрагменты ВТК, не затрагивая остальные элементы системы, распределять сервисы по разным серверам, реализовывать их на разных языках программирования и с применением различных технологий. На практике существует несколько стилей (стандартов) разработки приложении в соответствии с микросервисной идеологией, среди которых можно выделить REST (REpresentational State Transfer) [17]. REST – это архитектурный стиль программного обеспечения, включающий набор правил и ограничений на механику взаимодействия сервера и клиента в сети, основанный на использовании существующих стандартов: HTTP, URL, JSON, XML и других.

Достоинствами микросервисной архитектуры для предметной области ВТК являются [18]:

– масштабируемость приложений: добавление новых функций может быть реализовано через новый сервис, а удаление – через отключение ненужных сервисов;

– распределение задач: каждый сервис – это отдельный небольшой проект, который можно реализовать на любой платформе или языке программирования в соответствии с теми задачами, которые он решает;

– безопасность: сервисы изолированы и обмениваются данными по API, что позволяет обеспечить доставку только корректной информации;

– распределение ресурсов на каждый сервис: часть сервисов может требовать большой вычислительной нагрузки, часть – больших объемов памяти, тогда можно разместить их на разных серверах, ориентируясь на потребности каждого сервиса;

– безопасность отладки: поиск и исправление осуществляется в рамках изолированного от остальных компонентов модуля, что исключает возможность нарушения работы всей системы в процессе отладки;

– снижение степени зацепления: разделение проекта на изолированные, решающие конкретные задачи модули позволяет

снизить уровень зависимости между ними, упростить логику модулей, их тестирование и модернизацию;

– повышение связности: изолированность и фокус на решении узкого класса задач в рамках каждого микросервиса позволяет обеспечить высокую взаимосвязанность элементов внутри модуля, обеспечить их соответствие главной цели микросервиса.

Среди известных недостатков микросервисной архитектуры стоит выделить следующие: сложность тестирования из-за сложных маршрутов передачи информации между различными модулями, необходимость согласованности информации из-за задержки при передаче данных; рост объема передачи информации [19–21].

Проведенный анализ существующих решений, построенных на основе микросервисной архитектуры, показал, что многие крупные компании в сфере информационных технологий в последние годы осуществили переход от монолитной архитектуры к микросервисам (Amazon, eBay, Netflix, Spotify, Walmart и другие). Это позволило им реализовывать кроссплатформенные высоконагруженные, устойчивые сервисы. Положительный эффект от подобного перехода отмечается во многих отраслях, например, в ходе опроса 10 компаний из отраслей информационных технологий, торговли, логистики и туризма [22]. Выяснено, что влияние микросервисов на качество и поддержку программного обеспечения оценено как положительное в исследовании [23] на основе опроса 52 специалистов из банковской сферы, ИТ-индустрии и разработки программного обеспечения. Положительный эффект отмечен в сфере образования при реализации информационных систем для тестирования, обучения, автоматизации внутренних процессов университетов [24]. В настоящее время микросервисная архитектура в предметной области ВТК не применялась, однако ввиду обозначенных ранее преимуществ данный подход является перспективным направлением.

Таким образом, задача исследования состоит в необходимости разработки микросервисной архитектуры для ВТК, учитывающей влияние вышеперечисленных недостатков и реализующей все обозначенные достоинства для автоматизации и снижения сложности синтеза систем различного масштаба, обеспечения высокой связности и снижения степени зацепления программного кода.

2. Концепция микросервисной архитектуры ВТК. На основе проведенного литературного обзора и выявленных преимуществ микросервисного подхода к построению архитектуры информационных систем сформулирована концепция микросервисной

архитектуры ВТК (рисунок 1). В рамках разработанной архитектуры будем использовать следующие понятия.

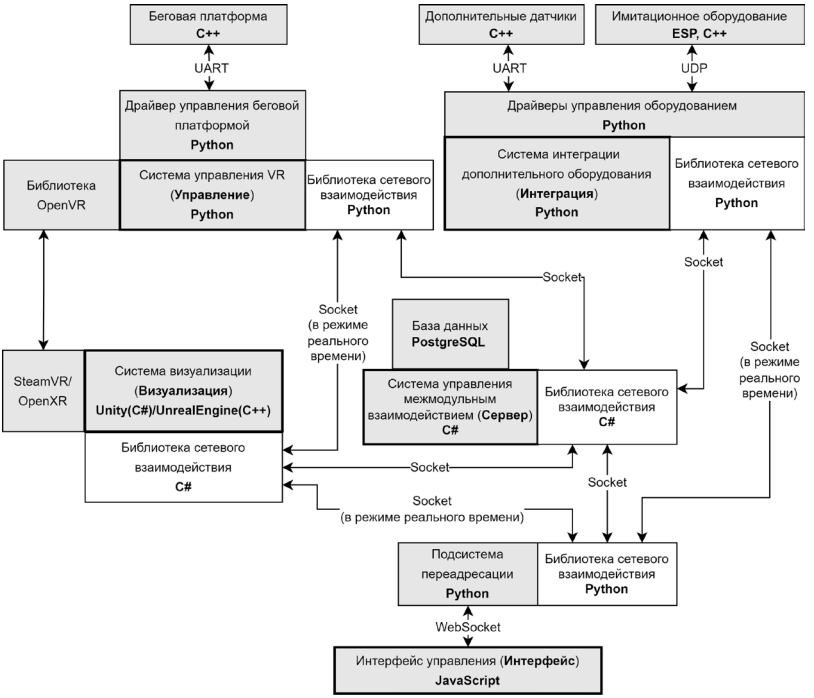

Рис. 1. Структурная модель микросервисной архитектуры ВТК

Микросервисная архитектура – сервис-ориентированная структура информационной системы, при которой осуществляется декомпозиция единого проекта на набор модулей с низким зацеплением и высоким уровнем внутренней связности.

Тогда для предметной области ВТК микросервисная архитектура должна обеспечивать выполнение следующих принципов, что позволит декомпозировать изначально монолитную архитектуру ВТК на набор микросервисов [25]:

– системный анализ объектов предметной области: необходимо осуществить анализ предметной области для последующего формализации моделей объектов и их программных абстракций, которые являются функциональной и структурной основой каждого микросервиса;

– изолированность: обеспечение независимости и автономности каждого микросервиса за счет инкапсуляции деталей его реализации от остальных компонентов системы, недопустимость выполнения отдельных функций или копирования элементов одного модуля в других микросервисах;

– автоматизация типовых процессов: использование унифицированных протоколов обмена информации, инструментов сетевого взаимодействия, средств тестирования и отладки, протоколирования;

– децентрализация процесса разработки: каждый микросервис должен разрабатываться независимо, принятие решений относительно программной реализации модуля осуществляется его разработчиками без внешнего согласования;

– прозрачность процесса разработки: наличие и использование эффективного инструментария для протоколирования, тестирования и отладки для обеспечения высокой скорости разработки и модернизации, возможности оперативного устранения неполадок.

Под термином *Система управления межмодульным взаимодействием* (Сервер) будем понимать обязательный микросервис архитектуры, реализующий функциональность сервера в ВТК, отвечающий за координацию остальных модулей, управление каналами связи и организацию передачи данных между модулями, работу с БД, хранение и обработку информации о процессе обучения и взаимодействия с виртуальной средой [26].

Под термином *Интерфейс управления* (Интерфейс) будем понимать обязательный микросервис архитектуры, реализующий процесс управления ВТК через графическую оболочку и следующую функциональность: авторизация пользователей, формирование ВТК (за счет компоновки из множества подключенных в сеть микросервисов), запуск тренировки, ее мониторинг и протоколирование.

Под термином *Система визуализации* (Визуализация) будем понимать обязательный микросервис архитектуры, отвечающий за визуальное представление виртуального пространства ВТК пользователю с использованием различных игровых трехмерных движков и фреймворков, включая работу с SteamVR или OpenXR для поддержки устройств виртуальной реальности.

Под термином *Система управления VR* (Управление) будем понимать обязательный микросервис архитектуры, обеспечивающий подключение, мониторинг и интеграцию компонентов VR (контроллеров, трекеров) для организации взаимодействия с

виртуальной средой и перемещения пользователя в сцене (в том числе, с помощью активных беговых платформ [27]).

Под термином *Система интеграции дополнительного оборудования* (Интеграция) будем понимать набор дополнительных микросервисов, реализующих управление и сбор данных с интегрированного в ВТК имитационного оборудования, дополнительных датчиков по проводным и беспроводным протоколам для повышения качества подготовки или сбора дополнительных данных о состоянии обучаемого.

Далее рассмотрим специфику реализации и функционировании каждой из сущности в рамках микросервисной архитектуры ВТК, так как разделение обозначенных компонентов, ранее тесно интегрированных между собой в монолитной архитектуре, неизбежно придет к значительным изменениям. Итак, определены 4 обязательных микросервиса и один вспомогательный, отражающий возможность функционального расширения архитектуры за счет применения различного имитационного оборудования и дополнительных датчиков. Разделение обозначенных сущностей на отдельные микросервисы позволило, с одной стороны, расширить их функциональность за счет возможности использования произвольных программных платформ, языков программирования и фреймворков, однако привело к необходимости их модернизации.

Микросервис Сервер реализуется на высокопроизводительных языках программирования, так как от его быстродействия напрямую зависит производительность работы всего ВТК. Необходимо отметить, что в отличие от сервис-ориентированной архитектуры, где каждый модуль системы связан с общим модулем интеграции, в микросервисной предполагается прямая связь модулей друг с другом и горизонтальная иерархия всех составляющих системы. В том случае, если одному модулю требуется постоянно обмениваться информацией с другим модулем, не оказывая на Сервер лишнюю нагрузку, то он может организовать с ним прямое подключение. В результате Сервер выполняет роль маршрутизатора, управляет доступом, обрабатывает ошибки, а также ведет записи в централизованном журнале. Связь модулей с Сервером и между собой оформлена в виде архитектуры клиент-сервер, в которой любой микросервис должен хранить состояние подключенных клиентов и выполняемых с ними процедур. Каждый микросервис обменивается с другим сообщениями, которые могут содержать данные, команды, события, ошибки, либо являться частью процедур. Помимо маршрутизации и контроля состояний всех

компонентов ВТК, Сервер отвечает за хранение и обработку информации из базы данных и их передачу другим микросервисам.

Микросервис Интерфейс реализует взаимодействие пользователя с ВТК. При использовании игровых движков разработка интерфейса приводит к значительному усложнению проекта, особенно, когда виртуальная сцена транслируется в системы виртуальной реальности. Кроме того, при организации процесса обучения требуется обеспечить контроль со стороны инструктора, что требует отдельного интерфейса. Поэтому полное отделение виртуальной сцены от интерфейса ее управления позволяет значительно упростить процесс разработки. Отделение интерфейса от остальных микросервисов дает возможность удаленного управления несколькими ВТК, быстрое переподключение между ними, отслеживание ошибок от всех остальных микросервисов. Интерфейс получает всю информацию от Сервера, но также может устанавливать подключение типа «клиент-клиент» с другими микросервисами (также при участии Сервера). Для обеспечения кроссплатформенности модуля Интерфейса его реализация предлагается в качестве Web-приложения на основе двух языков программирования: Python (для использования унифицированной системы сетевого взаимодействия на основе TCP-сокетов) и JavaScript (для реализации взаимодействия с интерфейсом и его динамического формирования на основе концепции Активностей). Передача информации внутри микросервиса между двумя составляющими осуществляется посредством технологии WebSocket. Концепция Активностей (Activity) заключается в повторном использовании программного кода логики и компонентов интерфейса, реализации жизненного цикла страницы, аналогично принятой в операционной системе Android при разработке приложений.

Микросервис Визуализация тесно взаимодействует с модулем Управление. За счет отделения визуальной составляющей (трёхмерные объекты, виртуальное пространство, эффекты, интерактивное взаимодействие) непосредственно от логики работы с аппаратным обеспечением VR (подключение и мониторинг контроллеров и датчиков движения, реализация системы перемещения с помощью сторонних платформ) был достигнут следующий положительный эффект: модуль Визуализация может быть реализован на базе любых игровых движков и сред разработки, в свою очередь, микросервис Управление может использовать в своей основе любой аппаратный комплекс для организации перемещения пользователя. Данные подсистемы, несмотря на тесную и совместную работу, являются

полностью заменимыми на аналоги, что не приведёт к изменению в логике работы каждого из них. Кроме того, логика непосредственного взаимодействия с аппаратным обеспечением различных платформ вынесена отдельно, а в микросервис Управление посредством драйвера предоставляется высокоуровневый API для передачи управляющих команд и отслеживания состояния, что также позволяет повысить гибкость работы данного модуля.

Микросервисы категории Интеграция хотя и являются дополнительными, позволяют значительно расширить функциональные возможности профессиональной подготовки с использованием ВТК. Приведем примеры таких микросервисов. В рамках предыдущих исследований в качестве подобного модуля успешно выступала имитационная система изолирующих дыхательных аппаратов [23]. Для данного типа оборудования также требуется разработка специализированных драйверов, предоставляющих высокоуровневые функции для управления и мониторинга состояния. Связь с оборудованием может быть организована как проводным, так и беспроводным способом. Помимо имитационного оборудования в качестве микросервиса Интеграции могут выступать различные медицинские датчики (пульсоксиметры, энцефалографы, электромиографы и так далее), что позволяет осуществить сбор дополнительных данных о состоянии обучаемого в процессе выполнения упражнений в виртуальной реальности.

Таким образом, одним из ключевых моментов научной новизны предложенной архитектуры является возможность гибкой замены основных компонентов без изменения архитектуры и влияния на остальные модули, возможность расширения архитектуры за счет добавления новых микросервисов Разделение общей структуры тренажерного комплекса на отдельные микросервисы в соответствии с теми задачами, которые они решают, значительно повышает связность (за счет ориентирования микросервиса на одну конкретную задачу) и снижает степень зацепления (микросервисы независимы между собой). Следующая важная составляющая микросервисной архитектуры ВТК – это организация процесса взаимодействия модулей.

3. Процедура сетевого взаимодействия микросервисов ВТК. Недостатком микросервисной архитектуры является требование к высокой надежности и корректности процесса обмена данными между модулями, так как в отличие от монолитной архитектуры объем передаваемой информации между компонентами возрастает в несколько раз. Для решения данной проблемы и обеспечения высокой надежности функционирования ВТК на основе микросервисной

архитектуры предлагается реализация отдельных программных компонент – библиотек сетевого взаимодействия, разграничивающих функции обмена и передачи информации между микросервисами. Определим основные функциональные возможности данных библиотек:

– формирование набора абстракций, формализующих все основные сущности ВТК;

– механизм десериализации сетевых пакетов, в сущности, программных абстракций объектной модели и сериализации сущностей в пакеты для передачи по сети;

– механизм передачи и приема сетевых пакетов с возможностью проверки целостности и корректности получаемых данных;

– реализация основных методов API, общих для всех микросервисов;

– реализация общих процедур, регламентированных протоколом управления, включая обнаружение Сервера в локальной сети, установку соединения с Сервером и с другими клиентами, штатное закрытие соединения и т.д.;

– ведение журнала событий модуля в едином для всей системы формате и управление параметрами журналирования;

– обработка ошибок в едином пространстве ошибок всей системы;

– управление конфигурацией модуля на основе унифицированного формата конфигурационных файлов;

– процедура загрузки модуля.

Рассмотрим один из вариантов структурной модели библиотеки, отвечающий заданным требованиям. Объектная модель системы представляет собой набор классов, инкапсулирующих все основные сущности (программные абстракции) внутри тренажерного комплекса, такие как: информация о модуле (подсистеме), состояние подключения, ошибка, объект команды и объект события, пользователь, тренажер, тренировка, сессия пользователя и т.д. Классы модели должны предоставлять все базовые методы для работы с сущностями: чтение и запись данных, сравнение, преобразование одних объектов в другие (если это предполагается API), валидацию данных и т.д. Также классы модели должны предоставлять возможность преобразования сущностей из формата, стандартного для используемого языка программирования в формат модели и обратно.

Организация функционирования межмодульного взаимодействия в микросервисной архитектуре заключается в строгой

унификации управляющего протокола и требует единого плана обработки любых запросов от других модулей. При таком подходе каждый микросервис включает в себя, кроме заданной функциональности, еще и реализацию модуля обмена сообщениями. Структура сообщений, которые будут пересылаться между элементами системы, имеет стандартизированную классификацию типов сообщений. В зависимости от типа сообщения определяется соответствующая структура вложенных объектов.

Далее рассмотрим реализацию унифицированного протокола обмена данными между микросервисами ВТК, основанную на использовании формата данных JSON. В качестве основного пакета внутри ВТК используется сущность «Сообщение» («message») (рисунок 2). Сообщение включает в себя ряд заголовков, содержащих различную служебную информацию, а также основной объект с полезной нагрузкой. Заголовки содержат информацию об отправителе и получателе пакета, типе этого пакета, а также несколько дополнительных идентификаторов, требуемых протоколом API (рисунок 2а).

а) Общесистемный пакет

```
"Message": {
   "message_type": #тип сообщения
   "src":  #источник сообщения
   "dst":  #цель сообщения
   "datetime":  #время отправки сообщения
   "request_id":  #id запроса
   "session_id":  #id сессии
   "payload":  #вложенный объект
}
```

б) payload для типа пакета “command”

```
"payload": {
      "args": null,
      "name": "unity.getScenesList"
}
```

в) Общий пакет для подтверждения успешного выполнения команды

```
{
   "payload": {},
   "message_type": "ack",
   "src": {
      "agent_name": "motion-control",
      "agent_id": "b9f25439-11d5-41a7-9a9c-74a8ca2a7645"
   },
   "dst": {
      "agent_name": "server",
      "agent_id": "547e1bba-fc92-44a2-b4be-98162ea6955c"
   },
   "datetime": 1651220209523,
   "request_id": 0,
   "session_id": "8e52a262-790f-4c94-9101-ea8b42b30484"
}
```

г) Общий пакет ошибки

```
{
   "payload": {
      "code": 13,
      "description": "Access denied",
      "cause": {
         "code": 13,
         "description": "System.NullReferenceException: Object
reference not set to an instance of an object.",
         "cause": null
      }
   },
   "message_type": "error",
   "src": {
      "agent_name": "unity",
      "agent_id": "50dd11df-3d0f-470f-b5fe-d9a92538f449"
   },
   "dst": {
      "agent_name": "server",
      "agent_id": "547e1bba-fc92-44a2-b4be-98162ea6955c"
   },
   "datetime": 1651220604575,
   "request_id": 0,
   "session_id": "e17baed2-7c3b-4954-905d-7898938c7d3e"
}
```

Рис. 2. Примеры основных типов пакетов

Тип пакета определяет категорию сообщения. В зависимости от типа сообщения модули делегируют обработку соответствующей подсистеме:

– discovery – обнаружение Сервера в локальной сети;
– initialization – подключение клиентов к Серверу и друг к другу;
– disconnect – отключение клиентов от Сервера и друг от друга;
– ack – подтверждение выполнения запроса/команды;
– command – команда;
– data – пакет с данными;
– event – системное событие;
– error – сообщение об ошибке.

Каждый микросервис содержит собственный модуль обработки сообщений, в который сообщение будет отправлено после десериализации. Первым этапом обработки принятого сообщения является Firewall, в котором проверяется необходимость получения сообщения. Вторым этапом является его проверка на соответствие текущим актуальным для модуля процедурам. В том случае, если сообщение корректно, выполняются команды, соответствующие данному этапу процедуры, далее состояние процедуры меняется, генерируется сообщение-ответ и обработка сообщения завершается на этом этапе, иначе сообщение отправляется на следующий этап обработки. На третьем этапе в соответствии с типом сообщения выполняется система команд, в результате будет сгенерирован ответ или ошибка при невозможности выполнения команды. В заключительном четвертом этапе сгенерированный ответ будет отправлен запустившему цепочку действий микросервису.

Прием и передача пакетов реализуются в соответствии с протоколами обмена «клиент-сервер» и «клиент-клиент». Классы подсистемы обмена сообщениями предоставляют высокоуровневый интерфейс для отправки и получения сообщений, позволяющий скрывать детали реализации сетевого обмена. Выполнение базовых процедур, таких как подключение и отключение от других узлов сети также реализуется в рамках подсистемы обмена сообщениями. Обработка основных команд, общих для всех модулей, выполняется в базовом классе обработчика команд.

Рассмотрим систему запрос-ответ на Сервере. В случае получения сообщения типа «command» Сервер десериализует объект payload и распознает тип команды. Для определения функциональности команды разработана классификация команд,

которые делятся на группы в зависимости от специфики модуля. Полное имя команды формируется из имени группы и предназначения команды в группе, например, «group.action». Структура объекта payload для типа «command» представлена на рисунке 2б.

После получения сообщение перенаправляется в метод-обработчик соответствующей команды, где опционально может быть получена дополнительная информация из объекта payload. Далее выполняется запрошенное действие и отправляется результат выполнения команды. Структура подтверждения команды представлена на рисунке 2в.

Если модуль, получивший любой запрос, не может успешно его выполнить, то модулю, отправившему запрос, будет возвращено сообщение с соответствующей ошибкой. Система ошибок унифицирована и разделяется на несколько логических групп, заданных для каждого микросервиса. Структура общего пакета ошибки представлена на рисунке 2г.

Для согласования работы модулей при выполнении простых запросов (команды, ошибки, события и т.д.) либо процедур используется поле «request_id» основного пакета «message». Значение данного поля заполняется любым узлом сети при инициации запроса. Модуль, генерирующий запрос другому модулю отправляет сообщение с определенным значением «request_id» и ожидает ответа с таким же значением данного поля.

Таким образом, решена задача организации межмодульного взаимодействия в микросервисной архитектуре за счет реализации единого протокола обмена информацией и отделения процедур сетевого взаимодействия в программные библиотеки в каждом микросервисе для унификации и повышения надежности работы системы. Это также сказывается на общей сложности ВТК, так как сетевые библиотеки модернизируются и поддерживаются отдельно, а разработчики микросервисов используют для взаимодействия модулей готовые решения.

4. Оптимизация межмодульного взаимодействия микросервисов. Выявленные в ходе анализа предметной области недостатки микросервисной архитектуры (большой объем передаваемой информации и высокие требования к скорости ее передачи для обеспечения согласованной работы всех модулей) приводят к необходимости оптимизации процесса межмодульного взаимодействия в рамках предложенной архитектуры для обеспечения высокой пропускной способности и низких задержек при передаче данных. Эта задача становится особенно актуальной, когда данные

необходимо обрабатывать в реальном времени, например, при передаче данных о положении пользователя из микросервиса Управления в Визуализацию.

Для постановки задачи оптимизации межмодульного взаимодействия микросервисов необходимо рассмотреть основные метрики и параметры данного процесса, а также область их возможных значений.

Пропускная способность канала C (байты в секунду): $C = I/Tt$, где I – объем переданной информации, Tt – время этой передачи. Ввиду естественных ограничений, значение C всегда неотрицательное, $C \geq 0$.

Доля активного времени центрального процессора (ЦП) L (проценты): $L = Ta/Tr \times 100\%$, где Ta – активное время, затраченное на обработку пакета данных, Tr – общее время приема сообщений. Значения L находятся в диапазоне $0\% \leq L < 100\%$, причем значение в 100% является недопустимым, так как в этом случае система не успевает обработать все полученные данные, следовательно, теряет пакеты.

Объем приемного буфера сообщений в оперативной памяти (ОЗУ) $V \to \min$. Значения V являются целочисленными и кратными степени 2, в рамках данной работы ограничены 32 мегабайтами: $8 \leq V < 32 \times 2^{20}$ (байт).

Доля больших (более 16 КБ) пакетов D (проценты): $D = NL/N \times 100\%$, где NL – число больших пакетов, N – общее число пакетов на протяжении одного сеанса обмена данными. Значения D находятся в диапазоне от 0 до 100%.

Таким образом, основной варьируемой переменной является размер буфера V, который будет напрямую влиять на метрики C и L, в качестве дополнительного входного параметра может использоваться значение D, которое будет отражать специфику передаваемых данных.

В формализованном виде постановка задачи оптимизации будет иметь вид: необходимо найти такой размер приемного буфера V и долю больших пакетов D, при которых целевые функции пропускной способности канала и доли активного времени ЦП стремятся к экстремальным значениям:

$$C(V, D) = I/Tt \to \max,$$

$$L(V, D) = Ta/Tr \times 100\% \to \min,$$

при выполнении ограничений:

$$C \geq 0, 0\% \leq L < 100\%, 8 \leq V < 32 \times 2^{20}, 0\% \leq D \leq 100\%.$$

Для решения задачи оптимизации необходимо выявить взаимосвязь между критериями и варьируемыми переменными. Поскольку современные вычислительные комплексы и операционные системы не предоставляют пользователю непосредственного контроля над ресурсами системы, выявить такие закономерности аналитически не представляется возможным из-за большого количества факторов (например, динамически изменяемая тактовая частота ЦП, алгоритмы работы планировщика операционной системы). Наиболее подходящим способом выявления взаимосвязей в данных условиях является вычислительный эксперимент.

Для проведения такого эксперимента в библиотеку сетевого взаимодействия добавлен алгоритм измерения времени приема пакета, подсистема для генерации тестового набора данных и программа для замера метрик C и L. При проведении эксперимента переменная V изменялась по всей области определения по экспоненциальному закону с интервалом в 4 шага на 1 порядок (в двоичной системе счисления). Переменная D варьировалась от 0 до 100% линейно с шагом в 10%. В результате проведения эксперимента получена таблица из 803-х значений. Замеры производились на следующем оборудовании: Intel I7 3770k (4 ГГц), 32Гб ОЗУ, использовался интерпретатор Python 3.8.

В результате отдельно рассмотрим две ситуации (рисунок 3): при отсутствии пакетов большой величины (D = 0%) и при обмене только большими пакетами (D = 100%). Для первого случая получено, что из Парето-оптимального множества буфер размером порядка V = 32 КБ является оптимальным. При обмене большими пакетами во втором случае требуется буфер V = 1 МБ. Как можно увидеть из графиков, Парето-оптимальные решения для обоих случаев не совпадают. Получено, что среди множества возможных вариантов решения не существует наиболее оптимальной комбинации входных переменных, пригодных для всех возможных вариантов использования. Высокая скорость передачи данных и низкие задержки достигаются только на достаточно большом размере буфера. С другой стороны, это негативно влияет на объем потребляемой ОЗУ.

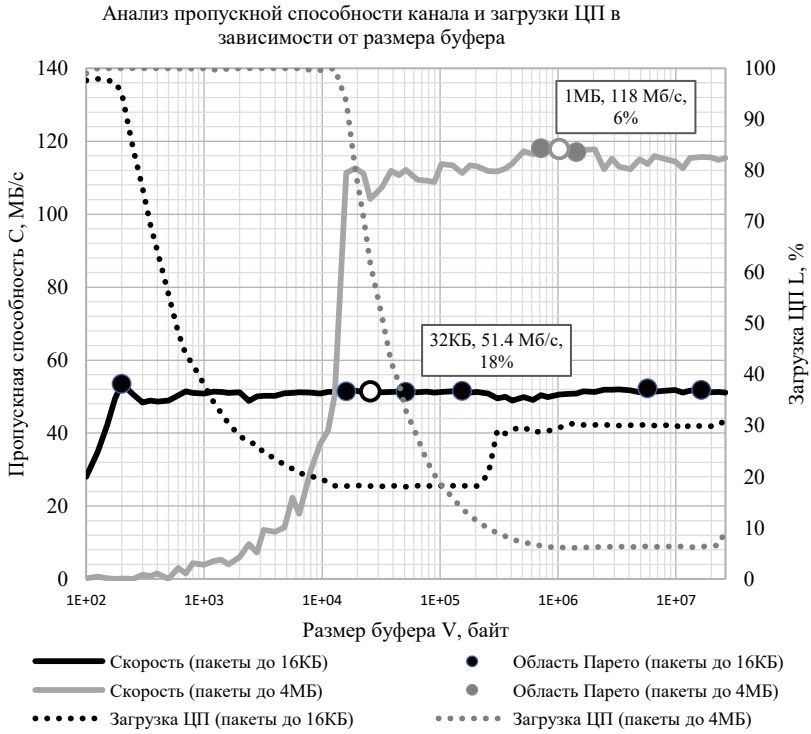

Рис. 3. Анализ метрик при различных размерах буфера V

В рамках данного исследования предлагается разделить каналы передачи данных между микросервисами на две категории, соответствующие наличию и отсутствию больших пакетов. Первая категория использует буфер, равный 32Кб, вторая – 1Мб. Тогда получим следующее решение задачи оптимизации: для буфера 32Кб $C = 51.4$ Мб/с, $L = 18\%$, для буфера 1Мб $C = 118$ Мб/с, $L = 6\%$.

Определенные в ходе решения задачи значения буфера обеспечивают наименьшую загрузку ЦП и наивысшую скорость среди области Парето. Примером связи первой категории являются Управление-Визуализация (передаются только управляющие команды и сведения о состоянии), второй – Визуализация-Сервер (передается полный протокол тренировки, список сцен с изображениями и описанием). В дальнейшем возможно использование адаптивного алгоритма выбора буфера, подстраивающего его размер под конкретную межмодульную связь.

5. Специфика функционирования ВТК, построенных на микросервисной архитектуре. Функционирование ВТК, реализованного в соответствии с микросервисной архитектурой, значительно отличается от проектов, выполненных по монолитному принципу. Компоненты системы могут быть как распределены между различными терминалами, так и запускаться на одном компьютере. Поэтому одним из важнейших процессов при функционировании системы является процедура инициализации всех микросервисов и объединение их в единую связанную структуру.

Процесс инициализации начинается с запуска всех микросервисов. Каждый из клиентских модулей запускает процедуру обнаружения Сервера в локальной сети. После обнаружения Сервера с ним устанавливается соединение, сначала на статически заданном порту, а затем на динамически выделенном (чтобы освободить статический порт для других клиентов). Изначально Сервер отвечает на запросы обнаружения только модулю Интерфейса. После подключения Интерфейса к Серверу пользователь проходит процедуру авторизации и может либо создать новый тренажер, либо выбрать существующий (в зависимости от прав доступа). После выбора тренажера сервер начинает отвечать на входящие запросы обнаружения всем клиентам, участвующим в работе этого тренажера.

Собранный тренажер сохраняется в базу данных для последующего быстрого запуска. В случае наличия всех микросервисов в системе пользователь может перейти к тренировке: выбрать доступные сцены, отправленные через Сервер микросервисом Визуализация, задать необходимые настройки трекеров или контроллеров посредством компонента Управление и дополнительного оборудования в микросервисе Интеграция. Далее пользователь через Интерфейс запускает тренировку.

В процессе прохождения тренировки Визуализация отправляет в Интерфейс данные о скорости перемещения пользователя, его действиях и ошибках. Также все микросервисы могут оповестить Сервер об ошибках в их работе, что будет отражено в Интерфейсе. После завершения тренировки все данные сохраняются в базу данных в виде протокола и могут быть проанализированы в дальнейшем.

Реализация ВТК для подготовки шахтеров в соответствии с микросервисной архитектурой представлена на рисунке 4. Микросервис Интерфейс (рисунок 4а) обеспечивает удаленное управление и мониторинг ВТК через веб-браузер. Микросервис Визуализация (рисунок 4б) на основе Unity включает виртуальное окружение и оборудование шахты, а также реализует штатные и

аварийные процессы. Микросервисы Управление и Интеграция (рисунок 4в) представлены управляемой беговой дорожкой и имитатором дыхательных аппаратов.

Таким образом, предложенный подход позволит декомпозировать структуру ВТК на отдельные микросервисы, тем самым повысив его гибкость, масштабируемость и возможность модернизации. Предложенное разделение на микросервисы позволяет реализовать их на оптимальных платформах, языках программирования, технологиях и фреймворках, эффективно распределить трудоемкость между разработчиками, тем самым снизив общую сложность проекта. Применение унифицированного протокола передачи данных, сетевых библиотек для организации межмодульного взаимодействия, а также оптимизация размера буфера при передаче информации позволяет минимизировать известные недостатки микросервисных архитектур и в полной мере реализовать их достоинства.

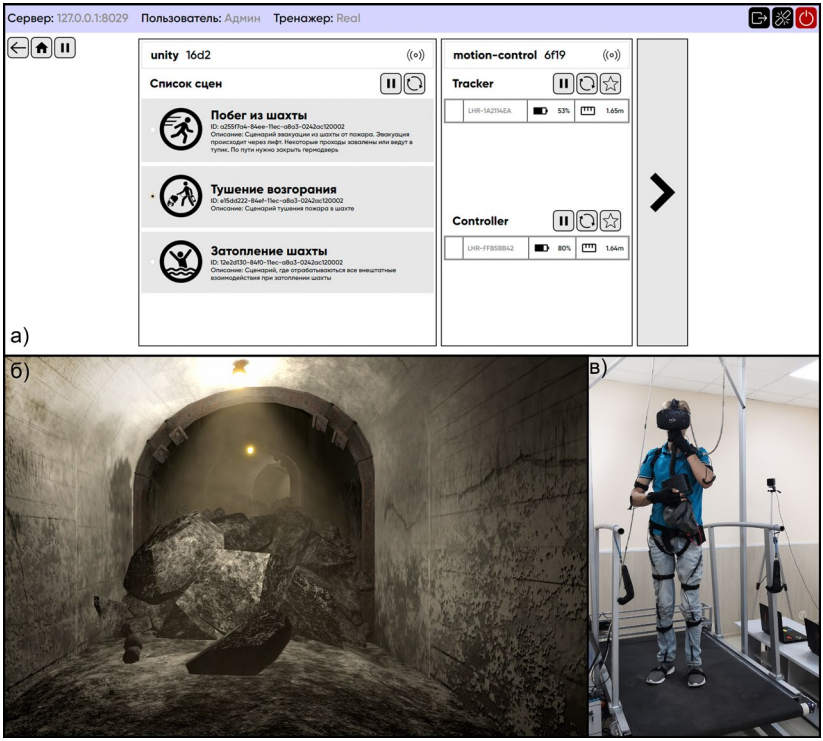

Рис. 4. Пример ВТК, реализованного на основе микросервисной архитектуры

6. Заключение. В работе рассмотрена микросервисная архитектура тренажерных комплексов, ее декомпозиция, особенности организации межмодульной связи и функционирования. Анализ специфики микросервисных архитектур выявил ряд достоинств и недостатков данного подхода, которые были учтены при адаптации данной архитектуры к предметной области ВТК. Предлагаемый подход позволяет разработчикам уйти от монолитного приложения, что повышает гибкость системы в плане модификации, ее расширяемость и стабильность, так как аварийное завершение одного из сервисов может быть корректно обработано остальными компонентами архитектуры (в том числе, с возможностью перезапуска или восстановления отдельных сервисов).

Реализованная в соответствии с предложенной микросервисной архитектурой система управления ВТК имеет ряд особенностей, связанных с распределением функциональности по отдельным узлам и необходимостью реализации постоянного обмена информацией между ними. Предлагаемая архитектура может использоваться для реализации тренажеров различных масштабов. Научная новизна предложенной микросервисной архитектуры заключается в переходе от классической монолитной архитектуры при синтезе тренажерных комплексов к использованию изолированных, слабо связанных микросервисов; реализации единого протокола обмена информацией между модулями; отделении процедур сетевого взаимодействия в программные библиотеки в каждом микросервисе для унификации и повышения надежности работы системы; возможности гибкой замены основных компонентов без изменения архитектуры и влияния на остальные модули. Также предложенная архитектура решает распространенную проблему реализации графического интерфейса инструктора тренажера, так как изначально отделяет интерфейс в отдельный модуль от системы визуализации.

Рассмотрена реализация библиотек сетевого взаимодействия микросервисов и унифицированного протокола обмена данных на основе JSON. Поставлена и решена задача оптимизации размера буфера для максимизации пропускной способности канала передачи данных между микросервисами и снижения нагрузки на центральный процессор. Предлагаемый подход позволяет снизить вероятность возникновения ошибок на уровне сетевого взаимодействия и обмена сообщениями между микросервисами, а также поддерживать API в актуальном состоянии во всех модулях, основанных на библиотеке.

Представлена специфика функционирования тренажерного комплекса на основе микросервисной архитектуры и приведены

преимущества микросервисной концепции в виде снижения программной сложности и возможности распределения вычислительной нагрузки. В соответствии с изложенной архитектурой реализован ВТК для подготовки шахтеров, обладающий следующими преимуществами: изолированный интерфейс управления для оператора, интеграция управляемой беговой дорожки для перемещения в виртуальной реальности и иного дополнительного оборудования (имитатор дыхательных аппаратов). В ходе дальнейших исследований планируется расширение функциональных возможностей архитектуры за счет добавления новых микросервисов, дальнейшая оптимизация процессов передачи информации, в том числе, за счет адаптивного буфера.

Литература

1. Zahabi M., Abdul Razak A.M. Adaptive virtual reality-based training: a systematic literature review and framework // Virtual Reality. 2020. vol. 24. no. 4. pp. 725-752.
2. Saldana D. et al. Applications of head-mounted displays for virtual reality in adult physical rehabilitation: a scoping review // The American Journal of Occupational Therapy. 2020. vol. 74. no. 5. pp. 7405205060p1-7405205060p15.
3. Jerdan S.W. et al. Head-mounted virtual reality and mental health: critical review of current research // JMIR serious games. 2018. vol. 6. no. 3. pp. e9226.
4. Zulueta A. et al. Virtual reality-based assessment and rating scales in ADHD diagnosis // Psicología Educativa. Revista de los Psicólogos de la Educación. 2019. vol. 25. no. 1. pp. 13-22.
5. Alcañiz M. et al. Eye gaze as a biomarker in the recognition of autism spectrum disorder using virtual reality and machine learning: A proof of concept for diagnosis // Autism Research. 2022. vol. 15. no. 1. pp. 131-145.
6. Obukhov A.D. et al. The study of virtual reality influence on the process of professional training of miners // Virtual Reality. 2022. pp. 1-25
7. Drossis G., Birliraki C., Stephanidis C. Interaction with immersive cultural heritage environments using virtual reality technologies // International Conference on Human-Computer Interaction. Springer, Cham. 2018. pp. 177-183.
8. Shinde P.P., Shah S. A review of machine learning and deep learning applications // 2018 Fourth international conference on computing communication control and automation (ICCUBEA). IEEE. 2018. pp. 1-6.
9. Qi X. et al. Applying neural-network-based machine learning to additive manufacturing: current applications, challenges, and future perspectives // Engineering. 2019. vol. 5. no. 4. pp. 721-729.
10. Delazio A. et al. Force jacket: Pneumatically-actuated jacket for embodied haptic experiences // Proceedings of the 2018 CHI conference on human factors in computing systems. 2018. pp. 1-12.
11. Andrews C. et al. Extended reality in medical practice // Current treatment options in cardiovascular medicine. 2019. vol. 21. no. 4. pp. 1-12.
12. Obukhov A. et al. Methodology for the Development of Adaptive Training Systems Based on Neural Network Methods // Proceedings of the Computational Methods in Systems and Software. Springer, Cham. 2021. pp. 238-253.
13. Tapia F. et al. From monolithic systems to microservices: A comparative study of performance // Applied sciences. 2020. vol. 10. no. 17. p. 5797.

14. Niknejad N. et al. Understanding Service-Oriented Architecture (SOA): A systematic literature review and directions for further investigation // Information Systems. 2020. no. 91. p. 101491.
15. Cerny T., Donahoo M.J., Trnka M. Contextual understanding of microservice architecture: current and future directions // ACM SIGAPP Applied Computing Review. 2018. vol. 17. no. 4. pp. 29-45.
16. Rushani L. et al. Differences between Service-Oriented Architecture and Microservices Architecture // International Journal of Natural Sciences: Current and Future Research Trends. 2022. vol. 13. no. 1. pp. 30-48.
17. Maurya R. et al. Application of Restful APIs in IOT: A Review // Int. J. Res. Appl. Sci. Eng. Technol. 2021. vol. 9. pp. 145-151.
18. Taibi D., Lenarduzzi V., Pahl C. Architectural patterns for microservices: a systematic mapping study // CLOSER 2018: Proceedings of the 8th International Conference on Cloud Computing and Services Science; Funchal, Madeira, Portugal, 19-21 March 2018. SciTePress. 2018.
19. Li S. et al. Understanding and addressing quality attributes of microservices architecture: A Systematic literature review // Information and software technology. 2021. vol. 131. pp. 106449.
20. Cerny T. et al. On code analysis opportunities and challenges for enterprise systems and microservices // IEEE Access. 2020. vol. 8. pp. 159449-159470.
21. Velepucha V., Flores P. Monoliths to microservices-Migration Problems and Challenges: A SMS // 2021 Second International Conference on Information Systems and Software Technologies (ICI2ST). IEEE. 2021. pp. 135-142.
22. Городничев М.Г., Полонский Р.В. Оценка возможности использования микросервисной архитектуры при разработке пользовательских интерфейсов клиент-серверного программного обеспечения // Экономика и качество систем связи. 2020. № 3 (17). С. 33-43.
23. Bogner J. et al. Microservices in industry: insights into technologies, characteristics, and software quality // 2019 IEEE international conference on software architecture companion (ICSA-C). – IEEE, 2019. pp. 187-195.
24. Auer F. et al. From monolithic systems to Microservices: An assessment framework // Information and Software Technology. 2021. vol. 137. pp. 106600.
25. Huang L., Zhang C., Zeng Z. Design of a public services platform for university management based on microservice architecture // Microsystem Technologies. 2021. vol. 27. №. 4. pp. 1693-1698.
26. Krasnyanskiy M.N., Obukhov A.D., Dedov D.L. Control System for an Adaptive Running Platform for Moving in Virtual Reality // Automation and Remote Control. 2022. Т. 83. №. 3. С. 355-366.
27. Obukhov A. et al. Mobile Simulator Control System for Isolating Breathing Apparatus of Software-Hardware Platform // International Journal of Interactive Mobile Technologies. 2020. Т. 14. №. 8. С. 32-42.

Обухов Артем Дмитриевич — д-р техн. наук, руководитель лаборатории, лаборатория медицинских VR тренажерных систем для обучения, диагностики и реабилитации, Тамбовский государственный технический университет. Область научных интересов: адаптивные информационные системы, структурно-параметрический синтез, нейронные сети, машинное обучение. Число научных публикаций — 151. obuhov.art@gmail.com; улица Советская, 106, 392000, Тамбов, Россия; р.т.: 8(915)867-6915.

Волков Андрей Андреевич — младший научный сотрудник, лаборатория медицинских VR тренажерных систем для обучения, диагностики и реабилитации, Тамбовский государственный технический университет. Область научных интересов:

информационные технологии, архитектуры информационных систем, распределенные системы управления. Число научных публикаций — 15. didim@eclabs.ru; улица Советская, 106, 392000, Тамбов, Россия; р.т.: 89537030619.

Назарова Александра Олеговна — техник-программист, лаборатория медицинских VR тренажерных систем для обучения, диагностики и реабилитации, Тамбовский государственный технический университет. Область научных интересов: информационные технологии, разработка программного обеспечения, виртуальная реальность. Число научных публикаций — 11. nazarova.al.ol@yandex.ru; улица Советская, 106, 392000, Тамбов, Россия; р.т.: +79204963910.

Поддержка исследований. Работа выполнена при финансовой поддержке Министерства науки и высшего образования РФ в рамках проекта «Разработка медицинских VR тренажерных систем для обучения, диагностики и реабилитации» (№ 122012100103-9).

DOI 10.15622/ia.21.6.7

A. OBUKHOV, A. VOLKOV, A. NAZAROVA

MICROSERVICE ARCHITECTURE OF VIRTUAL TRAINING COMPLEXES

Obukhov A., Volkov A., Nazarova A. **Microservice Architecture of Virtual Training Complexes.**

Abstract. The task of automating and reducing the complexity of the process of developing virtual training complexes is considered. The analysis of the subject area showed the need to move from a monolithic to a service-oriented version of the architecture. It is found that the use of a monolithic architecture in the implementation of virtual training complexes limits the possibility of modernizing the system, increases its software complexity, and makes it difficult to implement an interface for managing and monitoring the training process. The general concept of the microservice architecture of virtual training complexes is presented, and definitions of the main and secondary components are given. The scientific novelty of the research lies in the transition from the classical monolithic architecture in the subject area of the HTC to the microservice architecture; eliminating the shortcomings of this approach by implementing a single protocol for the exchange of information between modules; separation of network interaction procedures into software libraries to unify and improve the reliability of the system. The use of isolated, loosely coupled microservices allows developers to use the best technologies, platforms and frameworks for their implementation; separate the graphical interface of the simulator instructor from the visualization and virtual reality system; provide the ability to flexibly replace the main components (visualization, interface, interaction with virtual reality) without changing the architecture and affecting other modules. The decomposition of the structural model of the microservice architecture is carried out, and the specifics of the functioning of the main components are presented. The implementation of microservices networking libraries and a JSON-based data exchange protocol is considered. The practical significance of the proposed architecture lies in the possibility of parallelization and reducing the complexity of the development and modernization of training complexes.

Keywords: microservice architecture, microservices, virtual training complexes, intermodule interaction, inter-module interaction, data transfer optimization.

References

1. Zahabi M., Abdul Razak A.M. Adaptive virtual reality-based training: a systematic literature review and framework. Virtual Reality. 2020. vol. 24. no. 4. pp. 725-752.
2. Saldana D. et al. Applications of head-mounted displays for virtual reality in adult physical rehabilitation: a scoping review. The American Journal of Occupational Therapy. 2020. vol. 74. no. 5. pp. 7405205060p1-7405205060p15.
3. Jerdan S.W. et al. Head-mounted virtual reality and mental health: critical review of current research. JMIR serious games. 2018. vol. 6. no. 3. pp. e9226.
4. Zulueta A. et al. Virtual reality-based assessment and rating scales in ADHD diagnosis. Psicología Educativa. Revista de los Psicólogos de la Educación. 2019. vol. 25. no. 1. pp. 13-22.
5. Alcañiz M. et al. Eye gaze as a biomarker in the recognition of autism spectrum disorder using virtual reality and machine learning: A proof of concept for diagnosis. Autism Research. 2022. vol. 15. no. 1. pp. 131-145.
6. Obukhov A.D. et al. The study of virtual reality influence on the process of professional training of miners. Virtual Reality. 2022. pp. 1-25.

7. Drossis G., Birliraki C., Stephanidis C. Interaction with immersive cultural heritage environments using virtual reality technologies. International Conference on Human-Computer Interaction. Springer, Cham. 2018. pp. 177-183.
8. Shinde P.P., Shah S. A review of machine learning and deep learning applications. 2018 Fourth international conference on computing communication control and automation (ICCUBEA). IEEE. 2018. pp. 1-6.
9. Qi X. et al. Applying neural-network-based machine learning to additive manufacturing: current applications, challenges, and future perspectives. Engineering. 2019. vol. 5. no. 4. pp. 721-729.
10. Delazio A. et al. Force jacket: Pneumatically-actuated jacket for embodied haptic experiences. Proceedings of the 2018 CHI conference on human factors in computing systems. 2018. pp. 1-12.
11. Andrews C. et al. Extended reality in medical practice. Current treatment options in cardiovascular medicine. 2019. vol. 21. no. 4. pp. 1-12.
12. Obukhov A. et al. Methodology for the Development of Adaptive Training Systems Based on Neural Network Methods. Proceedings of the Computational Methods in Systems and Software. – Springer, Cham. 2021. pp. 238-253.
13. Tapia F. et al. From monolithic systems to microservices: A comparative study of performance. Applied sciences. 2020. vol. 10. no. 17. p. 5797.
14. Niknejad N. et al. Understanding Service-Oriented Architecture (SOA): A systematic literature review and directions for further investigation. Information Systems. 2020. no. 91. p. 101491.
15. Cerny T., Donahoo M.J., Trnka M. Contextual understanding of microservice architecture: current and future directions. ACM SIGAPP Applied Computing Review. 2018. vol. 17. no. 4. pp. 29-45.
16. Rushani L. et al. Differences between Service-Oriented Architecture and Microservices Architecture. International Journal of Natural Sciences: Current and Future Research Trends. 2022. vol. 13. no. 1. pp. 30-48.
17. Maurya R. et al. Application of Restful APIs in IOT: A Review. Int. J. Res. Appl. Sci. Eng. Technol. 2021. vol. 9. pp. 145-151.
18. Taibi D., Lenarduzzi V., Pahl C. Architectural patterns for microservices: a systematic mapping study. CLOSER 2018: Proceedings of the 8th International Conference on Cloud Computing and Services Science; Funchal, Madeira, Portugal, 19-21 March 2018. SciTePress. 2018.
19. Li S. et al. Understanding and addressing quality attributes of microservices architecture: A Systematic literature review. Information and software technology. 2021. vol. 131. pp. 106449.
20. Cerny T. et al. On code analysis opportunities and challenges for enterprise systems and microservices. IEEE Access. 2020. vol. 8. pp. 159449-159470.
21. Velepucha V., Flores P. Monoliths to microservices-Migration Problems and Challenges: A SMS. 2021 Second International Conference on Information Systems and Software Technologies (ICI2ST). IEEE. 2021. pp. 135-142.
22. Gorodnichev M.G., Polonskij R.V. Ocenka vozmozhnosti ispol'zovaniya mikroservisnoj arhitektury pri razrabotke pol'zovatel'skih interfejsov klient-servernogo programmnogo obespecheniya. [Evaluation of the possibility of using micro-service architecture in the development of user interfaces of client-server software]. Economics and Quality of Communication Systems. 2020. vol. 3. no. 17. pp. 33-43. (In Russ.).
23. Bogner J. et al. Microservices in industry: insights into technologies, characteristics, and software quality. 2019 IEEE international conference on software architecture companion (ICSA-C). IEEE. 2019. pp. 187-195.

24. Auer F. et al. From monolithic systems to Microservices: An assessment framework. Information and Software Technology. 2021. vol. 137. pp. 106600.
25. Huang L., Zhang C., Zeng Z. Design of a public services platform for university management based on microservice architecture. Microsystem Technologies. 2021. vol. 27. №. 4. pp. 1693-1698.
26. Krasnyanskiy M.N., Obukhov A.D., Dedov D.L. Control System for an Adaptive Running Platform for Moving in Virtual Reality. Automation and Remote Control. 2022. vol. 83. no. 3. pp.355-366.
27. Obukhov A. et al. Mobile Simulator Control System for Isolating Breathing Apparatus of Software-Hardware Platform. International Journal of Interactive Mobile Technologies. 2020. vol. 14, no. 8. pp. 32-42.

Obukhov Artem — Ph.D., Dr.Sci., Head of the laboratory, Laboratory of medical VR simulator systems for training, diagnostics and rehabilitation, Tambov State Technical University. Research interests: adaptive information systems, structural-parametric synthesis, neural networks, machine learning. The number of publications — 151. obuhov.art@gmail.com; 106, Sovetskaya St., 392000, Tambov, Russia; office phone: 8(915)867-6915.

Volkov Andrey — Junior researcher, Laboratory of medical VR simulator systems for training, diagnostics and rehabilitation, Tambov State Technical University. Research interests: information technologies, information system architectures, distributed control systems. The number of publications — 15. didim@eclabs.ru; 106, Sovetskaya St., 392000, Tambov, Russia; office phone: 89537030619.

Nazarova Alexandra — Programmer technician, Laboratory of medical VR simulator systems for training, diagnostics and rehabilitation, Tambov State Technical University. Research interests: information technologies, software development, virtual reality. The number of publications — 11. nazarova.al.ol@yandex.ru; 106, Sovetskaya St., 392000, Tambov, Russia; office phone: +79204963910.

Acknowledgements. Acknowledgements. The work was carried out with the financial support of the Ministry of Science and Higher Education of the Russian Federation within the framework of the project «Development of medical VR simulator systems for training, diagnosis and rehabilitation» (No. 122012100103-9).