

«АРС.СИМ»

Инженерная платформа цифровых двойников

Документация, содержащая описание технических средств хранения исходного текста и объектного кода программного обеспечения, а также технических средств компиляции исходного текста в объектный код программного обеспечения

ООО «АРС»

1. Общие положения

1.1. Назначение документа

Данный документ описывает высокоуровневую архитектуру программного обеспечения инженерной платформы цифровых двойников «АРС.СИМ» (далее – Платформа, АРС.СИМ, ПО). Техническая архитектура является гибридной и модульной, разделяя систему на несколько ключевых уровней для обеспечения масштабируемости, производительности и удобства сопровождения. Настоящий документ описывает технологии и инструменты, используемые для разработки, хранения исходного кода и компиляции, а также как обеспечивается их взаимодействие.

1.2. Термины и определения

В настоящем документе применяются следующие термины с соответствующими определениями, обозначения и сокращения:

Инженерная платформа цифровых двойников -	Программное обеспечение, обеспечивающее унифицированный запуск, развертывание и эксплуатацию виртуальных копий физических объектов, процессов и систем различной природы (оборудования, робототехнических комплексов, систем, технологических процессов, зданий и сооружений и пр.) и предоставление инструментов для работы с цифровыми двойниками
ПО -	Программное обеспечение
ODE	Высокопроизводительная библиотека с открытым исходным кодом для моделирования динамики твердого тела
SITL (Software in the loop)	Технология моделирования, заключающаяся в имитации аппаратных систем с помощью программного обеспечения
HITL (Hardware In The Loop)	Технология тестирования реального оборудования и программного обеспечения, путём программной имитации окружающей среды, с которой взаимодействует оборудование и сравнения выходных и входных данных оборудования и среды с ожидаемыми значениями
Исполнитель -	Разработчик, владелец платформы, ООО «АРС»
Заказчик -	Пользователь Платформы

2. Техническая архитектура программного обеспечения

2.1. Общая структура и уровни архитектуры

Уровень клиентского приложения. Основной пользовательский интерфейс. Представляет

собой нативное приложение для Windows, Linux, WebGL-версию для браузеров. Здесь происходит вся 3D-визуализация, рендеринг цифровых двойников и первичная обработка пользовательского ввода. Приложение отвечает за отображение текущего состояния объектов, анимацию и взаимодействие с моделями.

Среда моделирования реализована в Unity, отвечает за создание, настройку и выполнение имитационных и аналитических моделей цифровых двойников (физика, гидравлика, электрика, ML-модели), выступает в качестве интерактивного визуализатора и симулятора. Ее ключевые возможности, используемые в платформе:

- 3D-движок реального времени (Real-time 3D): используется для создания детализированных и физически корректных визуальных копий промышленных объектов, оборудования, робототехнических комплексов, процессов, систем. Это позволяет инженерам наблюдать за процессами в наглядной, интуитивно понятной форме.

- Модульная архитектура: система GameObject и Component позволяет создавать гибкую и переиспользуемую структуру цифрового двойника. Каждый физический объект (насос, датчик, робот) представлен в виде GameObject с набором компонентов, отвечающих за его визуализацию, логику, анимацию и связь с внешними данными.

- Физический движок (Physics Engine, ODE): используется для симуляции физических процессов, столкновений и взаимодействий объектов в виртуальной среде, что повышает точность имитационных моделей.

Ядро среды моделирования реализовано в виде набора C# скриптов и компонентов Unity, работающих внутри Unity-клиента. Это позволяет выполнять моделирование непосредственно в среде визуализации с минимальной задержкой.

Серверный уровень. Реализован на бэкенд-технологиях (C#, Python). Он отвечает за: сбор, агрегацию и хранение телеметрии от реальных объектов, управление пользователями и правами доступа, выполнение тяжелых аналитических и имитационных моделей, взаимодействие с внешними системами (ERP, MES), предоставление API (RESTful, WebSocket или gRPC) для подключения клиента на Unity, передача данных и интеграция, обеспечение асинхронной связи.

WebSockets / SignalR: для двунаправленной связи в реальном времени, что необходимо для отображения постоянно меняющихся показателей датчиков.

REST API: для выполнения административных и фоновых задач, таких как аутентификация пользователя, загрузка статической конфигурации сцены и 3D-моделей, а также отправка отчетов.

Библиотеки IoT: Интеграция с сервисами для прямого подключения к потокам данных от оборудования.

2.2. Хранение исходного и объектного кода. Компиляция

Исходные коды C# скриптов моделирования хранятся в Git-репозитории вместе с

основным проектом Unity (папка Assets/Scripts/Modeling и ProjectSettings).

Визуальные модели (графы узлов) сохраняются в текстовых форматах (JSON, YAML) или в сценах Unity (.unity), что позволяет версионировать их в Git.

Объектный код (скомпилированные сборки) - для внешних сервисов моделирования создаются отдельные репозитории (Python-модели упаковываются в Docker-образы и хранятся в реестре контейнеров) и хранятся в облачном хранилище S3.

Компиляция включает в себя импорт ассетов, компиляцию скриптов, упаковку ресурсов и, создание исполняемого файла для целевой платформы. Для этого используются следующие средства и подходы. C# - часть среды моделирования компилируется вместе со всем Unity-проектом через IL2CPP для релизных сборок. Внешние Python/C++ модели компилируются в рамках отдельного CI/CD конвейера (сборка Docker-образа при каждом изменении модели).

2.3. Веб расширение. Личный кабинет

Личный кабинет обеспечивает доступ к функциональности платформы через веб-браузер (дашборды, просмотр данных, управление лицензиями, версиями ПО). Для 3D-визуализации в браузере используется WebGL-версия. Веб-расширение развёртывается как статический сайт (в Azure Storage Static Web App, S3 + CloudFront) и обслуживается через API Gateway. Исходный код веб-приложения хранится в отдельном Git-репозитории. Общается с бэкендом, через REST API и WebSocket. Результат сборки (JS, CSS, HTML, ассеты) загружается как артефакт CI/CD в облачное хранилище (S3).

2.4. Мобильная версия

Мобильное приложение для оперативного мониторинга и получения уведомлений (просмотр статуса цифровых двойников, базовое управление). Единая кодовая база с десктопной версией Unity (используется тот же проект Unity, но с отдельными сценами и оптимизациями под мобильные устройства). Сборка под iOS и Android из того же Unity-проекта с использованием платформозависимых настроек (разрешения, сенсоры, push-уведомления). Для связи с бэкендом используются WebSockets (для реального времени) и REST API.

2.5. База данных

Централизованное хранилище всех данных платформы: метаданных цифровых двойников, конфигураций, телеметрии, пользовательской информации. Гибридная модель хранения: реляционная БД – для метаданных, пользователей, прав доступа, проектов, структуры двойников. Доступ к базам данных осуществляется только через бэкенд-сервисы (прямые подключения из клиента не допускаются по соображениям безопасности). Резервное копирование настроено на уровне СУБД (архивация) и объектного хранилища (репликация, бэкапы).

2.6. Технологическое ПО

Специализированное программное обеспечение, необходимое для функционирования платформы. От технологического ПО поступают сигналы PWR. Из среды физического моделирования поступает текущая телеметрическая информация о цифровом двойнике.

В режиме SITL и HITL, среда физического моделирования взаимодействует с контроллером оборудования (далее - КО), представленным в виде программы – эмулятора реального оборудования. Можно выделить 4 шага обработки данных:

- эмулятор КО выдает в Платформу сигнал PWR, который интерпретируется в среде физического моделирования в вектор тяги от электродвигателей устройства;
- среда физического моделирования вычисляет вектора перемещений и скоростей устройства относительно точки размещения устройства с предыдущей итерации;
- на основе вычисленных текущих значений векторов перемещений и скоростей устройства, с учетом первоначальных установочных значений вольт–амперной характеристики и емкости аккумуляторной батареи устройства, вычисляются значения скорости и ускорения устройства, давления среды, координат устройства, потребляемого тока, напряжения батареи, характеристики магнитного поля (текущая телеметрическая информация о действии устройства);
- рассчитанная средой физического моделирования текущая телеметрическая информация об устройстве передается в эмулятор КО, который после обработки выдает в среду физического моделирования сигналы PWR (SITL), который воспринимает и обрабатывает их как показания от физически реальных датчиков, с выдачей в среду физического моделирования сигналов PWR (HITL).