DOI: 10.25728/avtprom.2021.01.02

Проектирование распределенной экосистемы передачи данных цифрового двойника

М.Д. Пысин, Д.В. Зубов, Е.Б. Филиппова, В.С. Шушпанов, Э.М. Кольцова, Р.С. Крашенинников, А.В. Лобанов (РХТУ им. Д.И. Менделеева)

Рассмотрен процесс создания цифровых двойников технологических производств. Показана важность выбора архитектуры экосистемы передачи данных в системе цифрового двойника. Для решения этой задачи предложено использовать микросервисный подход. Даны рекомендации для построения экосистемы передачи данных. Предложенная архитектура может быть использована на реальных производствах путем интеграции реальных источников данных.

Ключевые слова: цифровой двойник, микросервисная архитектура, экосистема обмена данными.

Введение

Огромный прогресс вычисленных методов и математического моделирования резко расширил область применимости численных экспериментов. Часто различные аспекты какого-либо процесса на первый взгляд кажутся хорошо изученными (гидродинамика, теплопередача, механическая деформация и т.д.). Но в случаях взаимодействия большого числа аппаратов и технических систем в реальном технологическом процессе зачастую практически невозможно с достаточной достоверностью предсказать реакцию системы в целом на различные воздействия, особенно нештатные, выбрать оптимальный закон управления, подготовить операторов, не имеющих опыта практики на реальном производстве. Именно в таких ситуациях наиболее эффективно использование цифровых двойников — моделей, позволяющих как предсказывать поведение сложной технической системы, так и подбирать оптимальные воздействия для достижения заданных целей.

Цифровой двойник

Цифровой двойник — это интегральное мультифизическое, многомасштабное вероятностное моделирование комплексной системы, призванное наиболее полно и точно отобразить жизненный цикл и работу своего оригинала [1]. Цифровые двойники используются для экономии на проведении дорогостоящих экспериментальных исследований каких-либо систем, продуктов, агрегатов, комплексов, с целью выявления недостатков, прогнозирования аварийных ситуаций и масштабирования пилотных разработок до реальных производственных мощностей. При этом цифровой двойник — это уже значительно больше, чем просто моделирование, и даже больше, чем просто основанное на моделировании проектирование. Это система, для которой симуляция служит ядром для поддержки жизненного цикла всего продукта в целом [2]. Цифровые двойники широко применяются рядом крупных компаний [3].

Важным аспектом цифровых двойников является способ моделирования, который позволяет получить два вида моделей.

1. Теоретические (аналитические) модели, которые создаются на основе балансовых уравнений, уравнений состояния, законов физики, химии и т.д. Они по возможности наиболее детально и правильно описывают физические и химические процессы, происходящие в объекте, двойник которого создается.

2. Экспериментальные модели, создаваемые на основе анализа реальных данных, собранных с существующего объекта. Данные могут быть обработаны разными способами, например, при помощи методов машинного обучения.

Иногда используется комбинация этих двух подходов — вид уравнений модели создается на основе теоретического анализа, а значения коэффициентов — на основании анализа экспериментальных данных.

Оба подхода имеют как достоинства, так и недостатки. При теоретическом построении модели создание двойника может начинаться параллельно или даже задолго до создания реального объекта, тогда как экспериментальный метод построения модели требует наличия существующего объекта, и определенного времени его работы для того, чтобы были накоплены данные, которые будут использоваться в процессе моделирования. С другой стороны, наиболее приближенные к реальному процессу, теоретические модели часто являются очень громоздкими. Они включают в себя системы дифференциальных уравнений, иногда трудные для решения. Их расчеты требуют значительных вычислительных ресурсов, тогда как методы построения экспериментальных моделей позволяют получать простые модели, обладающие достаточной точностью и легко работающие в режиме реального времени. Экспериментально-аналитический метод требует наличия реального объекта и значительных вычислительных затрат при использовании модели, однако позволяет получить высокую точность моделирования.

Тем не менее, все методы, которые используются при построении цифровых двойников, объединяет то, что в процессе как их построения, так и их работы происходит передача и работа с огромным объемом данных, потоки которых нужно координировать и контролировать. Так в работе [2], посвященной описанию концепции цифрового двойника, и в работе

[4] Фрéймворк — программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

[4], посвященной выработке основ фреймворка[1] для построения систем цифровых двойников, отдельно выделяется пункт, связанный со сбором и обработкой данных, необходимых для функционирования таких систем. Отдельно вопрос данных поднимается и обсуждается в работе [5], посвященной концепции разработки продуктов, построенной на повсеместном применении цифрового двойника. Исходя из вышесказанного, можно сделать вывод, что описание базовой концепции и проектирование экосистемы передачи данных внутри системы цифровых двойников с множеством агентов, которые поставляют данные, и множеством различных потребителей данных, занимающихся их обработкой и визуализацией, является актуальной задачей.

Цифровой двойник как распределенная система

Из постановки задачи вытекают следующие особенности проектируемой системы.

• Система содержит некоторое не нулевое число участников, которые предоставляют данные для других участников, то есть являются поставщиками данных (агенты).

• Система содержит некоторое, не нулевое, число участников, которые могут получать данные и заниматься какой-либо дальнейшей их обработкой, то есть являются потребителями данных.

• Потребитель данных может являться также и поставщиком других данных, при этом поставщик может являться потребителем, если иное не указано.

Таким образом, проектируемая система обмена данными является распределенной. В ней фигурируют множество составляющих, каждая из которых выполняет возложенные на нее функции, и предоставляет некоторую долю своей информации, для общего использования, или, что является эквивалентным, часть общей информации, модифицирует по своему усмотрению и получает общую информацию для выполнения своих функций.

Одним из возможных решений для построения распределенной системы является концепция, называемая Enterprise Service Bus (ESB), или сервисная шина предприятия, описанная во множестве работ, как например [6], посвященной использованию этой концепции для решения задачи построения, ориентированной на множество сервисов архитектуры, или [7], в главе о сервис-ориентированных архитектурах. У этого решения есть свои преимущества, такие как:

• упрощение и стабилизация управления структурой обмена данными, порядком вызовов сервисов и их взаимодействий за счет обладания большим объемом информации о присутствующих в системе сервисах;

• стандартизация протоколов обмена, формата и управления передачей данных между сервисами, что обеспечивает стабильность передачи, интегрируемость новых сервисов в общую структуру обмена и контролируемость всего процесса передачи;

• защищенность доступа к данным и простоту разграничения прав на доступ ко всей системе обмена данными.

Для решения поставленной задачи необходимо учитывать, что концепция ESB плохо применима для создания непрерывно модифицируемой, эволюционирующей системы. Этот недостаток вызван рядом факторов.

• Избыточной связностью системы. Каждый отдельный сервис в такой системе не является в полном смысле самостоятельным — сервис жестко интегрирован во всю последовательность обработки информации, то есть в случае выхода из строя одного сервиса все последующие по цепочки вызова за ним могут оказаться не способны выполнять возложенные на них функции, так как они полностью зависят от всего что происходило с данными до них, и, как следствие, — вся последовательность вызовов не выполнится.

• Жесткой последовательностью вызовов. Поскольку последовательностью вызовов сервисов для обработки информации управляет шина передачи данных, то обеспечение отказоустойчивости и толерантности к отказу отдельных операций в последовательности вызова сервисов требует значительных усилий как на этапе проектирования, так и на этапе эксплуатации системы.

• Сложностью изменений. Поскольку весь процесс завязан на шине, то изменения базовых компонентов, возникающие при необходимости изменения отдельных частей системы или интеграции новых сервисов, могут вызывать значительные трудности и приводить к отказу системы.

Вышеописанные недостатки перекрывают те преимущества, которые предоставляет концепция ESB для построения системы обмена данными цифрового двойника, так как цифровой двойник, являясь не стандартизированной на данный момент концепцией, не имеет постоянных и устоявшихся правил и общепринятых подходов для решения задач проектирования и эксплуатации, а, следовательно, системы, построенные на этой концепции, будут постоянно изменяться и адаптироваться под новые реалии работы.

Исходя из этого свойства цифровых двойников (готовность к изменению), экосистема обмена данными между частями в такой системе также должна быть хорошо приспособлена к постоянной модификации и адаптации. Подходящей для решения этой задачи концепцией построения распределенных систем, которая в последнее время активно развивается и широко используется, является концепция микросервисной архитектуры. Микросервисная архитектура описана в работе [8] как архитектурный стиль для построения распределенной системы, в которой каждый сервис является небольшим, легко развертываемым процессом, который выполняет возложенные на него функции, является независимым процессом,

и взаимодействует с другими сервисами при помощи легковесного протокола обмена данными. Микросервисная архитектура является удобной для постоянно эволюционирующих систем, и, соответственно, оптимальным образом подходит для решения проблемы проектирования экосистемы обмена данными цифрового двойника.

Спроектированная экосистема обмена данными цифрового двойника

В книге [7] и документации [8] описана концепция микросервисной архитектуры. На основании этой концепции спроектирована архитектура экосистемы передачи данных системы построения цифровых двойников, приведенная на рисунке.

Экосистема состоит из следующих элементов: супервизор, источники данных, потребители данных.

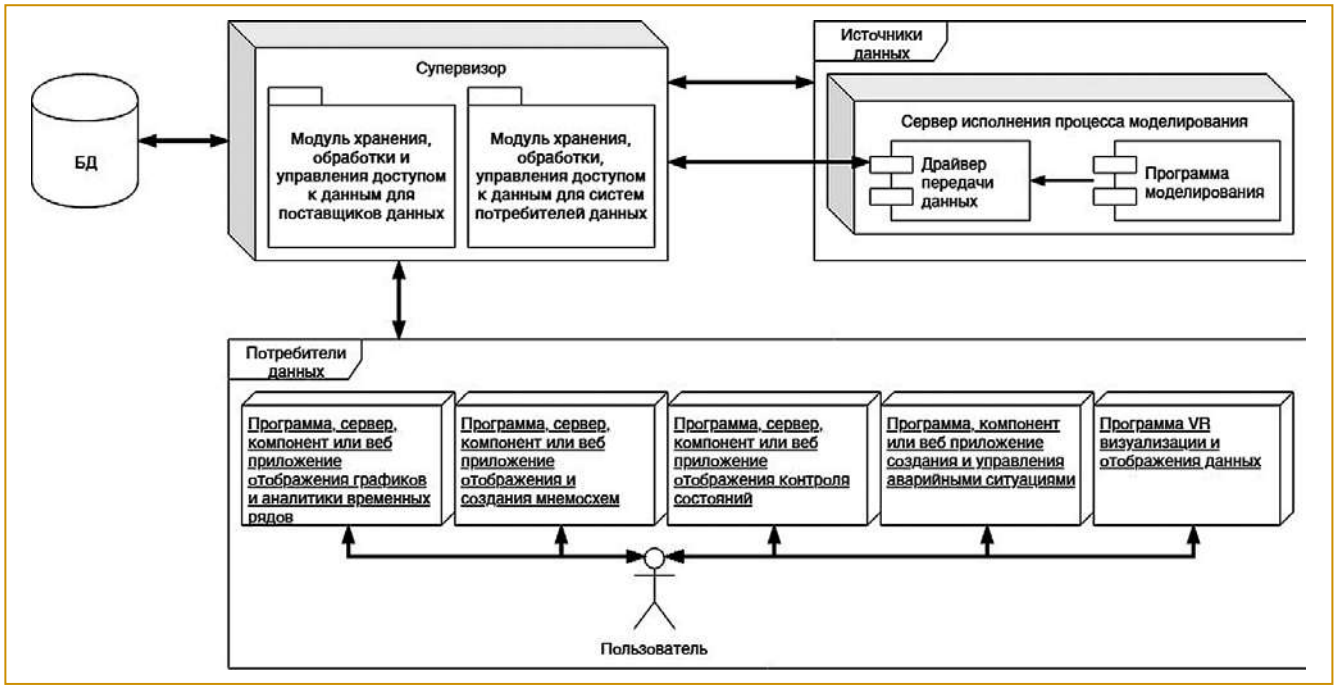
Супервизор

В рамках проектируемой системы супервизором называется сервер обработки и хранения оперативных данных цифровых двойников, в задачи которого в базовой конфигурации входит:

- хранение оперативных, то есть соответствующих текущему положению дел, данных;
- предоставление доступа к оперативным данным как потребителям данных, так и источникам данных, с обеспечением контроля на предмет доступности этих данных для каждой конкретной части системы;
- предоставление механизмов модификации оперативных данных как для потребителей, так и для источников с обеспечением контроля на предмет доступности этих данных для каждой конкретной части системы;
- предоставление механизмов регистрации и учета новых моделируемых цифровых двойников и их связывания с другими двойниками для совместной работы;
- предоставления механизмов регистрации и учета новых потребителей данных.

Таким образом, при помощи супервизора решаются задачи по управлению всей системой передачи данных, а также задачи по защите и авторизации всех участников обмена данными, что не позволит неавторизованному потребителю получить доступ к передаваемой информации и внести в нее изменения, таким образом нарушив работу и дискредитировав моделирование. С другой стороны, если предоставить супервизору слишком много функций контроля над системой и завязать все потоки данных исключительно на него, итоговая архитектура по своей сути будет напоминать уже упомянутую сервисную шину предприятия, а, соответственно, потеряет преимущества легкости эволюционного развития. Для устранения данной проблемы супервизор агрегирует в себя только оперативные данные по текущим значениям параметров моделируемого процесса и только по тем параметрам, которые источники самостоятельно решают ему предоставить. Источник может перестать предоставлять определенные параметры либо расширить список предоставляемых параметров в любой момент. Таким образом, супервизор, хоть и является ключевой частью экосистемы передачи данных, но система способна работать и без него. Все потребители данных способны передавать и получать информацию напрямую от источников, если источники предоставляют такую функциональность.

При этом супервизор строго определяет и стандартизирует формат и механизмы передачи данных. Возможно создание собственного протокола, но собственный протокол передачи данных требует реализации драйверов, обеспечивающих его работу на всех задействованных в системе устройствах. Это



Принципиальная структура экосистемы передачи данных системы цифрового двойника

приведет к возрастающей сложности эволюционной интеграции новых частей в уже существующие двойники. Также возрастет сложность создания новых двойников, предполагающих связывание их со старыми. Отдельной сложной задачей при разработке собственного протокола будет реализация механизмов, гарантирующих сложность или невозможность компрометации пересылаемых данных, подмены источников этих данных, их перехвата. Каждая из упомянутых проблем не имеет простого и однозначного решения, что усложняет весь процесс разработки системы в целом и затягивает его завершение.

По вышеописанным причинам разработка собственного протокола передачи данных не целесообразна. Для достижения главной цели работы в качестве основного протокола передачи данных был выбран протокол HTTPS, совмещающий уже подтвердивший свою работоспособность и простоту в использовании протокол http, а также обеспечивающий в достаточной мере защищенный способ передачи протокол SSL/TSL. Исходя из выбранного протокола передачи данных, логичным форматом передаваемых данных будет формат файла JSON, так как работа с файлами данного формата имеет уже множество пригодных для использования библиотек во всех популярных языках программирования. В итоге, весь механизм и правила передачи данных между сервисами будут построены на концепции REST API в его вариации работы с форматом JSON. При этом, как уже упоминалось ранее, такое решение является основным, но в силу специфики выбранной базовой концепции архитектуры системы не является единственным, и допускается изменение протоколов общения непосредственно между отдельными частями системы без использования супервизора.

Источник данных

Источник данных в проектируемой системе — это в базовом варианте сервис, являющийся либо программой, либо модулем, либо сервером, который занимается непосредственно моделированием технологической схемы. В его основные задачи входит:

- определение передаваемых и получаемых данных и уведомление о их составе супервизора;
- генерация набора данных, распространяющихся по системе, для которых сервис является источником;
- самостоятельная передача этих данных в нужном формате супервизору с характерной и определяемой исключительно самим сервисом периодичностью;
- запрос, периодичность которого задается самим сервисом, у супервизора текущих значений тех данных, которые сам сервис определил для себя как изменяемые и использование их по своему усмотрению.

Таким образом, источником данных в базовой конфигурации может выступать любая программа для моделирования, развернутая в рамках какого-либо окружения, и все, что она дополнительно должна реализовывать — это передачу данных при помощи протокола HTTPS и работу с форматом JSON, которую в базовой версии предоставляет практически любой язык программирования. Если система моделирования по какой-либо причине не может обеспечить работу со средой передачи данных, то есть не поддерживает протокол передачи данных HTTPS или работу с данными формата JSON и не позволяет модифицировать себя для внесения их поддержки, допускается и предполагается создание отдельной программы драйвера на одном из распространенных языков программирования. Данная программа будет выступать в роли прослойки, которая каким-либо доступным для программного комплекса моделирования способом, получит данные результатов моделирования, и, преобразовав их в нужный формат, отправит на сервер супервизора. Такое построение процесса обмена данными источника с супервизором позволяет первому самостоятельно приемлемым для него образом контролировать процесс моделирования и обмена данными, в том числе отделяя их друг от друга и обеспечивая их независимость. При этом, какие-либо внутренние изменения не вызывают необходимости оповещения всей системы в целом и супервизора в частности.

Потребители данных

Потребителем данных в итоговой экосистеме считается какой-либо сервис, который, как и источник, может являться как отдельной программой, так и модулем другой программы или отдельным сервером. Этот сервис для того, чтобы быть участником системы, должен решать следующие задачи:

- определение необходимых для запроса параметров;
- самостоятельный запрос параметров с сервера супервизора со своей собственной характерной только для себя периодичностью;
- изменение и уведомление о необходимости внесения изменений в параметры, хранимые на сервере супервизора с отправкой модифицированных данных;
- способность работы в отсутствии полного состава всех необходимых данных.

Реализуя вышеперечисленные функции, потребитель данных способен получать необходимую ему информацию с определяемым им самим временем актуальности. Разумеется, в условиях распределенной системы, актуальность данных гарантируется только собственной частотой запросов этих данных, что периодически может перегружать общую систему обмена данными. Однако реализация супервизора должна учитывать такую нагрузку. Также в случае, если система разрастется в процессе своего эволюционного развития во время эксплуатации до такого состояния, что один сервер супервизора не сможет справляться с возросшим потом данных, вся концепция супервизора и способы общения с ним должны будут поддерживать различные виды контейнеризации и балансировки нагрузки. А также не исключается вариант жесткого нормативного ограничения максимальной частоты запроса данных.

Заключение

Предложена микросервисная архитектурная концепция экосистемы передачи данных в системе цифровых двойников. Ключевым, но не необходимым для работы узлом итоговой архитектуры, является сервер супервизор, обеспечивающий стандартизированный способ обмена данными и поддерживающий адаптацию к возможно возрастающим нагрузкам путем контейнеризации и балансировки потоков данных. Основной и базовый, но не единственно доступный протокол передачи данных — это HTTPS. При этом данные в таком случае передаются в формате JSON, а весь процесс обмена данными построен на принципах REST API.

Дальнейшая работа по развитию предложенной архитектуры должна быть связана в первую очередь с встраиванием в нее дополнительных источников данных в виде реальных объектов, которые должны иметь возможность быть интегрированы в систему для использования их как в качестве обычного источника, аналогичного цифровому двойнику, с целью применения потребителей данных для работы с ними, так и в качестве источника первичных реальных данных для программ моделирования, основанных на технологиях машинного обучения.

Список литературы

1. *Glaessgen E. H., and D. Stargel.* The Digital Twin Paradigm for Future NASA and US Air Force Vehicles. // 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AI, 23 - 26 April 2012, Honolulu, Hawaii - 2012 - p. 1-14.
2. *Boschert S., & Rosen R.* Digital Twin—The Simulation Aspect. // Mechatronic Futures, Switzerland - 2016 - 59-74.
3. *Qi Q., Tao F., Zuo Y., & Zhao D.* Digital Twin Service towards Smart Manufacturing. // 51st CIRP Conference on Manufacturing Systems - 2018 - Vol. 72 - p. 237-242.
4. *Tao F., Sui F., Liu A., Qi Q., Zhang M., Song B., Guo Z., Lu S.C.-Y.., Nee A. Y. C.* Digital twin-driven product design framework. // International Journal of Production Research - 2019 - Vol. 57 - № 12 - p. 1-19.
5. *Tao F., Cheng J., Qi Q., Zhang M., Zhang H., Sui F. (2017).* Digital twin-driven product design, manufacturing and service with big data. // The International Journal of Advanced Manufacturing Technology, - 2017 - Vol. 94 - p. 3563-3576.
6. *Schmidt M.T., Hutchison B., Lambros P., & Phippen R. (2005).* The Enterprise Service Bus: Making service-oriented architecture real. // IBM Systems Journal – 205 - Vol. 44 - № 4 - p. 781-797.
7. *Ford N., Parsons R., Kua Patrick.* Building Evolutionary Architectures: Support Constant Change - 1st Edition - O'Reilly Media, 2017 – 190 p.
8. *Microservices / J. Lewis and M.* Fowler. https://martinfowler.com/articles/microservices.html (10 November 2020).

Пысин Максим Дмитриевич *– аспирант,* ***Зубов Дмитрий Владимирович*** *– канд. техн. наук, доцент,*
Филиппова Елена Борисовна *– канд. техн. наук, доцент,* ***Шушпанов Виктор*** *– аспирант,*
Кольцова Элеонора Моисеевна *– д-р техн. наук, проф., заведующий кафедрой,*
Крашенинников Роман Сергеевич *– магистрант,* ***Лобанов Алексей Владимирович*** *– магистрант кафедры информационный компьютерных технологий*
Российского химико-технологического университета им. Д.И. Менделеева.
E-mail: aprogrom@gmail.com feb_ikt@muctr.ru

Первый в России диагностический стенд на базе ЧПУ Fanuc

При выполнении ремонтов металлорежущих станков необходимо иметь возможность выполнять точную диагностику всех основных узлов оборудования. В связи с этой потребностью в компании Инжис было решено разработать и изготовить диагностический стенд на базе ЧПУ Fanuc.

Работы по созданию нового продукта были разбиты на несколько этапов. Среди них: моделирование, конструирование и 3D-проектирование, а также подбор и поставка комплектующих, монтаж и отладка.

Необходимость в точном подборе составляющих стенда, в том числе и из-за их довольно высокой стоимости, не позволяла ошибиться и начать сначала.

Второй и самой большой сложностью стала необходимость найти компанию, которая бы изготовила конструкцию стенда, – потенциальные исполнители этих работ зачастую отказывались вникать в специфические особенности инновационного продукта. Тем не менее, такой исполнитель нашелся – компания «ЭнергияЛаб» (Воронеж), сотрудничество с которой позволило реализовать проект в железе.

Другие сложности были связаны с необходимостью создания ПО для управления стендом, который специалистам Инжис пришлось написать с нуля. В этом им была оказана консультационная и техническая поддержка специалистов российского представительства Fanuc.

На сегодня инновационный диагностический стенд изготовлен, испытан и позволяет на высоком профессиональном уровне осуществлять диагностику, ремонт и настройку станков с ЧПУ.

Можно утверждать, что в России подобный проект реализован впервые.

https://rci36.ru