Известия СПбГЭТУ «ЛЭТИ». 2024. Т. 17, № 3. С. 44–55
LETI Transactions on Electrical Engineering & Computer Science. 2024. Vol. 17, no. 3. P. 44–55

УДК 004.94 Научная статья

https://doi.org/10.32603/2071-8985-2024-17-3-44-55

Использование динамических цифровых двойников при построении киберфизических систем

В. Я. Ананьева[1], **А. И. Водяхо**[1]✉, **Н. А. Жукова**[2], **М. А. Червонцев**[1]

[1] Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина), Санкт-Петербург, Россия

[2] Санкт-Петербургский Федеральный исследовательский центр Российской академии наук, СПИИРАН, Санкт-Петербург, Россия

✉ aivodyaho@mail.ru

Аннотация. Рассматривается подход к построению сложных распределенных киберфизических систем, реализованных на платформах туманных и граничных вычислений, с высоким уровнем архитектурной динамики. Подход основан на использовании динамических моделей наблюдаемой системы. Модели поддерживаются в актуальном состоянии посредством обработки потока событий, поступающих в форме логов. Предложена эталонная архитектура динамического цифрового двойника времени выполнения. Описана графовая модель, используемая в динамическом цифровом двойнике. Данный подход ориентирован на использование в крупномасштабных распределенных системах различного назначения, построенных на платформах IoT и IIoT, и может представлять интерес для специалистов, занимающихся исследованиями и разработками в области с высоким уровнем архитектурной вариабельности.

Ключевые слова: киберфизические системы, динамические цифровые двойники, синтез моделей

Для цитирования: Использование динамических цифровых двойников при построении киберфизических систем / В. Я. Ананьева, А. И. Водяхо, Н. А. Жукова, М. А. Червонцев // Изв. СПбГЭТУ «ЛЭТИ». 2024. Т. 17, № 3. С. 44–55. doi: 10.32603/2071-8985-2024-17-3-44-55.

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Original article

The Use of Dynamic Digital Twins in the Cyber-Physical Systems Design

V. Ya. Ananeva[1], **A. I. Vodyaho**[1]✉, **N. A. Zhukova**[2], **M. A. Chervoncev**[1]

[1] Saint Petersburg Electrotechnical University, Saint Petersburg, Russia

[2] SPIIRAS of St. Petersburg Federal Research Center of the Russian Academy of Sciences, Saint Petersburg, Russia

✉ aivodyaho@mail.ru

Abstract. The article considers an approach to the construction of complex distributed, implemented on the platforms of fog and edge computing, cyber-physical systems with a high level of architectural dynamics. The approach is based on the use of dynamic digital twins. The models are kept in actual state by processing the flow of events coming in the form of logs. A reference architecture of a dynamic runtime digital twin is proposed. The graph based model used by the dynamic digital twin is described. The described approach is focused on the use in large-scale distributed systems for various purposes, built on IoT and IIoT platforms, and may be of interest to specialists engaged in research and development in the field with a high level of architectural variability.

Keywords: cyber-physical systems, dynamic digital twins, model synthesis

For citation: The Use of Dynamic Digital Twins in the Cyber-Physical Systems Design / V. Ya. Ananeva, A. I. Vodyaho, N. A. Zhukova, M. A. Chervoncev // LETI Transactions on Electrical Engineering & Computer Science. 2024. Vol. 17, no. 3. P. 44–55. doi: 10.32603/2071-8985-2024-17-3-44-55.

Conflict of interest. The authors declare no conflicts of interest.

Актуальность. Цифровые технологии представляют собой стратегические движущие силы, во многом определяющие тенденции развития современного общества.

Данные, информация и знания становятся движущей силой цифровой трансформации [1] современного глобального общества.

В широком смысле под цифровизацией [2] обычно понимают не только активное использование цифровых технологий, но и зрелую фазу перехода от аналогового представления мира к цифровому. При этом современные цифровые технологии предоставляют три основные возможности для общества и бизнеса [3]: вездесущие данные, неограниченные средства связи и неограниченные вычислительные мощности. Концепции цифровизации и цифровой трансформации самым тесным образом связаны с использованием новых цифровых технологий и платформ, что, в свою очередь, связано с постоянными изменениями в используемых архитектурных решениях на всех уровнях, начиная от бизнес-архитектуры и заканчивая технической архитектурой.

Успехи в таких областях, как нанотехнологии, телекоммуникации, программная инженерия, позволили выйти на принципиально новый уровень сложности создаваемых антропогенных систем. При этом уровень сложности определяется не только числом элементов и числом уровней иерархии, но и степенью вариабельности структуры и поведения [4].

При построении современных информационно-ориентированных систем (ИОС) в большинстве случаев используются такие парадигмы, как интернет вещей (Internet of Things, IoT), промышленный интернет вещей (Industrial Internet of Things, IIoT), киберфизические системы (КФС), социокибернетические системы, системы окружающего интеллекта [5], [6].

КФС нового поколения часто состоят из элементов разной физической природы, создаются для решения самых разнообразных задач. При создании таких КФС программный и информационные компоненты играют ключевую роль, т. е. можно утверждать, что современные КФС являются ИОС.

Увеличение сложности создаваемых КФС выражается не только в увеличении числа элементов и уровней иерархии, усложнении структуры связей между элементами системы, но и в усложнении поведения систем.

На протяжении последних лет модельный подход находит самое широкое применение в качестве инструмента решения задач интеграции гетерогенных систем и сбора данных. Одним из вариантов модельного подхода можно считать подход, основанный на использовании цифровых двойников (ЦД).

В данной статье понятие ЦД определяется следующим образом: «ЦД – это работающая в реальном времени модель физической или виртуальной системы, которая способна адаптироваться к изменениям в наблюдаемых и управляемых системах посредством анализа собираемых в реальном времени данных, а также предсказывать поведение и состояние наблюдаемой системы (НС)».

Работая со сложными КФС, которые часто строятся по принципу системы систем (System of Systems, SoS) [7], обычно не удается построить некоторую единую монолитную модель и приходится использовать системы (сети) ЦД. Сети ЦД можно определить как распределенную систему, элементами которой служат ЦД.

Современное состояние. В настоящее время технологии, основанные на использовании ЦД, уже находят применение в таких сферах, как производство, авиация, здравоохранение, телекоммуникации, умный дом, умный транспорт, умный город и т. п. Если в 2020 г. рынок ЦД оценивался в 3.3 млрд долларов, то в 2028 г. он составит не менее 38 млрд долларов [8].

Практически все современные КФС в той или иной степени реализуют механизмы адаптации и адаптивного поведения, т. е. их структура и поведение изменяются во времени. Все чаще создаваемые системы реализуют элементы когнитивного поведения, предполагающие, в частности, способность к обучению и самообучению [9].

Проблемы, стоящие на пути более широкого распространения ЦД. На пути более широкого внедрения в практику ЦД-технологий стоят две связанные между собой основные проблемы:

1. ЦД-технологии в части реализации достаточно сложны и дороги. Несмотря на существен-

ные потенциальные выигрыши, расходы на разработку часто оказываются непомерно высокими, фактически приходится выполнять двойную работу: разрабатывать и сами системы, и их модели для всех этапов жизненного цикла систем.

2. Использование ЦД требует более мощных вычислителей и более широких каналов связи. Во многих случаях, особенно когда речь идет о системах низкого цифрового диапазона, это оказывается неприемлемым.

Предлагаемый подход. Динамические ЦД. Предлагаемый подход ориентирован на использование применительно к сетям ЦД времени выполнения, входящих в состав сложных КФС с высоким уровнем структурной и функциональной динамики (вариабельности).

В основу развиваемого подхода к построению КФС на базе динамических ЦД (ДЦД) положены следующие принципы:

1. ДЦД рассматриваются как средство реализации архитектурной гибкости и средство построения гибких цифровых нитей (ЦН).

2. Реализация гибкости обеспечивается за счет использования ДЦД, содержащих знания о текущем состоянии КФС или ее элементов.

3. Уровень гибкости, который закладывается в архитектуру, требует дополнительных расходов. Выбор уровня гибкости – это задача оптимизации ТСО.

4. Используемые модели, в основе своей динамические, строятся и поддерживаются в актуальном состоянии в процессе функционирования автоматически.

5. Современные КФС имеют достаточно большое время жизни, и на протяжении их жизненного цикла (ЖЦ) постоянно меняется контекст, в котором они функционируют. Их можно отнести к классу развивающихся систем. Необходимость строить КФС с гибкой архитектурой обусловлена, в первую очередь, неопределенностями требований. Учет неопределенностей должен быть заложен на этапе архитектурного проектирования.

По сравнению с известными подходами можно выделить два основных отличия:

– все модели, входящие в состав ДЦД, строятся и поддерживаются в актуальном состоянии автоматически;

– разрабатываемые динамические модели строятся в контексте ЦН, что предполагает построение моделей на базе тех, которые относятся к более ранним этапам ЖЦ.

Требования, предъявляемые к ДЦД времени выполнения (ВВ). Предметом рассмотрения в данной статье служат ДЦД и сети ДЦД времени выполнения. В свою очередь, основной элемент ДЦД ВВ – это динамическая исполняемая модель, которая должна строиться и поддерживаться в актуальном состоянии в процессе функционирования НС. Следует заметить, что в качестве НС применительно к КФС могут выступать сущности, имеющие разную физическую природу: разного рода виртуальные сущности, в частности ЦД, физические сущности, природные сущности, включая природные экосистемы, люди и коллективы, а также системы, состоящие из сущностей разных типов.

Общие требования к ДЦД ВВ могут быть описаны следующим образом:

– быть способными описывать НС, которая может быть построена по принципу SoS, иметь динамическую структуру и реализовывать адаптивное, в частности, когнитивное поведение;

– быть функционально полными, т. е. содержать данные, информацию и знания (ДИЗ) о НС, которые позволяют отвечать на запросы всех заинтересованных сторон о ее состоянии, т. е. быть Multi-stakeholder;

– быть исполняемыми, т. е. обладать способностью работать в режиме run time и, желательно, в режиме по крайней мере нежесткого реального времени;

– иметь реализации приемлемой сложности, даже для НС, состоящих из большого числа элементов.

Эталонная архитектура ДЦД ВВ. ДЦД ВВ представляют собой подкласс класса ДЦД, ориентированного на использование на этапе функционирования.

На этом этапе решаются две основные задачи: обслуживание запросов пользователей в условиях вариабельной архитектуры и управление архитектурными состояниями (АС), описываемыми А-моделью, которая должна поддерживать механизмы реализации архитектурной гибкости, т. е. перехода между АС.

Под АС понимается совокупность описания структуры и поведения НС. АС описываются с помощью А-модели. Структура НС описывается с помощью С-модели, а поведение – с помощью П-модели.

Для решения первой задачи следует знать текущее АС НС, а для решения второй задачи требуется некоторая статистика об эффективности

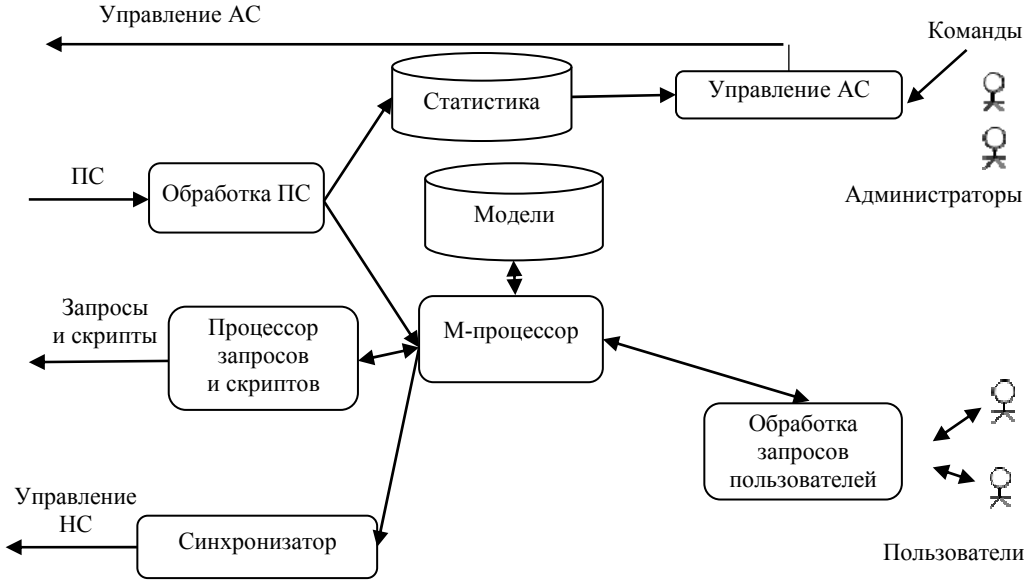

Рис. 1. Эталонная архитектурная модель ДЦД ВВ
Fig. 1. Reference architectural model of runtime dynamic DT

функционирования системы, находящейся в данном конкретном АС.

На рис. 1 показана *эталонная архитектурная модель* (*reference architectural model*) *ДЦД ВВ*.

Данная модель включает в себя следующие элементы: подсистему обработки потока событий (логов), репозитарий событий (статистика), подсистему управления архитектурными состояниями, репозитарий моделей (модели), процессор моделей (М-процессор), подсистему обработки запросов пользователей и синхронизатор. М-процессор отвечает за выполнение операций с моделями (построение, модернизация, обработка запросов к моделям).

Подсистема «*Обработка потока событий*» (*ПС*) отвечает за прием и обработку потоков входных событий, поступающих от НС. События, в зависимости от специфики наблюдаемой системы, могут поступать в разных форматах:

<событие> ::= <простое событие>| <значения параметров>| <информация об изменении значений параметров>| < информация об изменении состояния НС или ее элементов>| <текстовые сообщения>| <документы>| <сериализованные объекты>.

Подсистема обработки ПС реализует процедуру чистки событий (удаление дублирований, ложных событий, обработку сложных событий). Можно считать, что эта подсистема отвечает за придание событиям определенного смысла.

Обработанные события поступают в репозитарий событий (*Статистика*) и в *М-процессор*. События, поступившие в модуль *Статистика*, обрабатываются *подсистемой* «*Управление архитектурным состоянием*» (*АС*), которая отвечает за переход между АС (его функционирование подробно будет рассмотрено далее).

Подсистема «*Синхронизатор*» отвечает за приведение структуры и бизнес-процессов (БП), протекающих в НС, в соответствие с моделью.

Подсистема «*Обработка запросов пользователей*» обеспечивает обработку запросов пользователей и формирование требуемых представлений.

В процессе функционирования все обращения пользователей обрабатываются с использованием *Модели*, т. е. информация о текущем состоянии НС берется из модели. За корректность и актуальность модели отвечает отдельный процесс. Функционирование ДЦД ВВ может быть описано в терминах трех отдельных асинхронно функционирующих процессов:

1) обработки пользовательских запросов;

2) построения и поддержания в актуальном состоянии модели НС;

3) поддержки архитектурной гибкости.

Обобщенный алгоритм функционирования ДЦД выглядит следующим образом.

Процесс обработки запросов от пользователей:

Шаг 1. Запуск фоновых процессов мониторинга состояния НС (поллинг).

Шаг 2. Ожидание запросов.

Шаг 3. Если есть запрос от наблюдателя, то формирование представления и переход к п. 2.

Шаг 4. Если есть запрос от лица, принимающего решение (ЛПР), то формирование представ-

ления и ожидание запроса на получение указаний на выдачу управляющих воздействий.

Шаг 5. Коррекция модели и переход к п. 2.

Шаг 6. Если есть событие, интересное для Х-процессора, то – коррекция модели и переход к п. 2.

Процесс обработки потока событий:

Шаг 1. Запуск фоновых процессов мониторинга потока событий.

Шаг 2. Сканирование потока событий в фоновом режиме.

Ожидание информации о событии в НС или внешнем мире в форме лога.

Шаг 3. Если поступил лог, то чистка лога, обработка, коррекция модели и переход к п. 2.

Шаг 4. Если поступил лог с информацией об изменении модели со стороны пользователей, то формирование и выдача управляющих воздействий в НС, переход к п. 2.

Процесс поддержки архитектурной гибкости может быть представлен следующим образом:

Шаг 1. Запуск фонового процесса мониторинга архитектурного состояния.

Шаг 2. Сканирование потока событий в фоновом режиме.

Шаг 3. Оценка важности события с точки зрения целесообразности перехода в новое АС.

Шаг 4. Если требуется переход в новое АС, то вычисляется значение функции соответствия и принимается решение о выборе нового АС.

Шаг 5. Формируется и выполняется скрипт перевода НС в новое АС.

Шаг 6. Переход к п. 2.

Система команд М-процессора. М-процессор управляет текущей моделью посредством набора команд по реструктуризации текущей модели, а также отвечает за построение моделей. Его система команд включает три группы команд (табл. 1):

1) низкоуровневые операции с моделями;

2) низкоуровневые запросы к моделям;

3) высокоуровневые операции с моделями.

ДЦД как наблюдатель. В упрощенном общем виде ДЦД можно представить как четверку:

$$\text{ДЦД} = \langle \text{НО}, \text{М}, \text{П}, \text{И} \rangle,$$

где НО – наблюдаемый объект; М – модель НО; П – процессор модели; И – интерфейс взаимодействия между НО и М (рис. 2).

НО может быть как физической, так и виртуальной сущностью. Если НО представляет собой виртуальную сущность, то в частном случае он может выполнять функции М, т. е. быть моделью.

Табл. 1. Система команд М-процессора
Tab. 1. M-processor command system

Команда	Название
Низкоуровневые операции с моделями	
AddV	Добавить вершину
DelV	Удалить вершину
AddL	Добавить связь
DelL	Удалить связь
AddAt	Добавить атрибут
DelAt	Удалить атрибут
Низкоуровневые запросы к моделям	
GetAt	Получить значение атрибута
SetAt	Установить значение атрибута
Высокоуровневые запросы к моделям	
ChkCon	Проверить актуальность модели
CreM	Построить модель по логам
ProcLog	Обработка лога
SincM	Синхронизировать наблюдаемый объект (НО) и модель
CmpM	Сравнить модели
QM	Найти подмодель

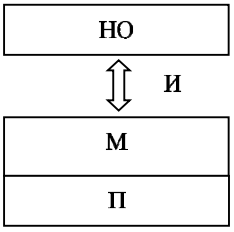

Рис. 2. Упрощенная структура ДЦД
Fig. 2. Simplified structure of dynamic DT

Понятие наблюдателя достаточно хорошо известно из теории управления.

В общем виде наблюдателя можно определить как

$$\text{OBS} = \langle \text{Conc}, \text{DSL} \rangle,$$

где Conc – интересы наблюдателя, а DSL – язык, с помощью которого наблюдатель может реализовывать свои интересы.

Основные типы ДЦД. В зависимости от специфики конкретной решаемой с помощью ЦД задачи и роли конкретной заинтересованной стороны можно выделить следующие основные точки зрения на ЦД:

1) модель (система моделей), сопровождающая НС на определенных этапах ЖЦ;

2) порт доступа к другим системам и мирам, в частности физическим системам;

3) интеллектуальный контроллер некоторой сущности;

4) сама НС;

5) архитектурный стиль.

В первом случае речь идет о реализации концепции цифровой нити (ЦН) [10].

Во втором случае ЦД рассматривается как средство построения НС, построенных по принципу системы систем (System of Systems, SoS). Каждая система, входящая в состав SoS, может иметь произвольное число ЦД, которые описывают систему на языке, понятном другим системам, входящим в состав SoS. В этом случае ЦД можно рассматривать как сервис, доступный другим системам, входящим в состав SoS.

В третьем случае ЦД рассматривается как средство реализации механизма адаптивности, в этом случае ЦД управляет реконфигурацией наблюдаемой и управляемой системы.

В четвертом случае ЦД рассматривается с точки зрения программной архитектуры – как одну из возможных реализаций архитектурного стиля – *виртуальные машины* [11].

В пятом случае система, состоящая из реконфигурации НС и ЦД, рассматривается как единая система, доступ к которой осуществляется через ЦД. Это пользовательская точка зрения.

Эти точки зрения можно рассматривать как архитектурные точки зрения [12].

Возможны разные подходы к использованию ЦД в реальных системах. Чаще всего встречаются два типа ЦД.

ЦД первого типа – это полный или неполный функциональный аналог НС, реализующий сервисы, которые необходимы в конкретном контексте. Такой подход позволяет заменять физические системы на их модели, например на этапах проектирования, отладки, испытаний.

ЦД второго типа – это некоторая частная модель НС, которая строится для решения конкретной задачи в интересах конкретного пользователя. Например, такой подход можно использовать для сбора данных и представления результатов в виде модели.

С понятием ЦД тесным образом связано понятие *цифровой тени* [13]. Применительно к НС с вариабельной структурой и поведением понятие цифровой тени можно определить как моментальный снимок, отображающий состояние НС на конкретный момент времени. Множество цифровых теней, относящихся к отдельным упорядоченным во времени моментам времени, образуют *теневой след*. В отличие от ЦД, цифровая тень пассивна, и она только отображает текущее состояние НС в конкретный момент времени.

ЦД может быть реализован разными способами – встроен в саму управляемую систему или реализован как отдельный компонент.

Метамодель ДЦД ВВ. ЦД могут сопровождать НС на протяжении всего ЖЦ, однако модели, относящиеся к разным фазам, существенно различаются. Наименее разработаны модели, относящиеся ко времени выполнения (ВВ).

Предлагаемая модель ВВ ориентирована на использовании в ДЦД ВВ и представляет одну из возможных реализаций и адаптацией акторной модели применительно к ДЦД ВВ. Предлагаемая модель имеет следующие отличительные особенности:

– НС рассматривается как гетерогенная распределенная КФС, физические компоненты которой представлены ЦД;

– модель – дискретная;

– модель описывает НС в терминах структуры, поведения и ресурсов;

– структура и поведение могут изменяться во времени;

– ресурсы могут быть физическими, виртуальными или смешанными.

Предлагаемую метамодель можно определить как

$$MM = <Act, Afr, Lnk, Tk, Res, BP>,$$

где Act – множество акторов; Afr – множество правил запуска акторов; Lnk – множество типов связей между акторами; Tk – множество типов сообщений (токены, фишки); Res – множество типов ресурсов; BP – множество бизнес-процессов (БП), описывающих функциональность НС.

Данные компоненты, в свою очередь, можно определить следующим образом:

<Act> ::= <Преобразование>|<Управление>|<Порт>,

<Afr>::= <Правила запуска><Правила формирования результата>,

<Lnk>::= <Связи по ресурсам>|<Связи по данным>|<Связи по управлению >,

<Tk> ::=<Токен данных>|<Токен управления>|<Ресурсный токен>,

<Res>::= <Физический ресурс>| <Виртуальный ресурс>,

<BP>::= <Статический>|<Динамический>.

В простейшем случае используется статический БП. В этом случае в процессе функционирования все операторы и связи между ними известны заранее.

В этом случае модель может быть представлена в виде *размеченного* (*labbeled*) *графа*, содержащего вершины двух типов: акторы и ресурсы. Акторы соединены между собой дугами, определяющими информационные зависимости. Ресурсные вершины соединены дугами, которые соответствуют физическим или логическим связям между отдельными ресурсами. Акторы и ресурсные вершины соединены дугами, имеющими смысл «выполняется на». Эти графы можно рассматривать как отдельные подграфы:

$$MM = \{Gd, Gr, Ldr\},$$

где Gd – размеченный граф данных; Gr – ресурсный граф; Ldr – связи между вершинами Gd и Gr.

Принципиально Gr можно определить через атрибуты вершин Gd, но этот граф представляет интерес как предмет синтеза.

Каждый актор имеет n входящих и m исходящих дуг (рис. 3).

Входящие дуги:

– произвольное число дуг данных D,

– одна дуга C для директивного запуска актора,

– одна дуга Z для получения запросной фишки,

– одна дуга R для получения фишки с информацией о готовности ресурса.

Исходящие дуги:

– произвольное число дуг данных D,

– одна R-дуга, по которой выдается фишка с информацией об освобождении ресурса.

Таким образом, в системе акторов циркулируют следующие типы фишек:

– фишки данных (D-фишки);

– фишки принудительного запуска актора (C-фишки);

– фишки запросов (Z-фишки);

– ресурсные фишки (R-фишки).

Каждая фишка может принадлежать либо классу стационарных фишек, либо классу подвижных фишек. Стационарные фишки не меняют своей позиции в процессе функционирования, а подвижные фишки могут появляться и исчезать. По своей сути подвижные фишки идентичны фишкам в сетях Петри.

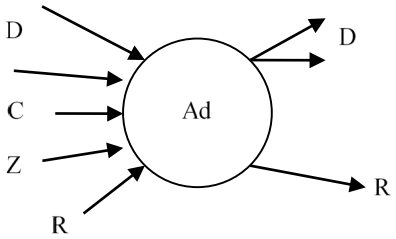

Рис. 3. Структура элемента Ad графа данных
Fig. 3. Structure of the Ad data graphelement

В предлагаемой модели реализуется следующая логика работы.

Если на всех входных дугах появились фишки, то актор срабатывает. По одной подвижной фишке с каждого входа исчезает. На выходах появляются новые фишки в соответствии с логикой работы актора. Все стационарные фишки сохраняют свои позиции.

Ресурсный граф Gr отражает текущую структуру, которая в основе своей виртуальная, поскольку элементы отличной природы представлены ЦД. Ресурсный граф Gr можно определить как

$$Gr = <R, Lr, Ldr>,$$

где R – ресурсы; Lr – связи между ресурсами; Ldr – связи с Ad.

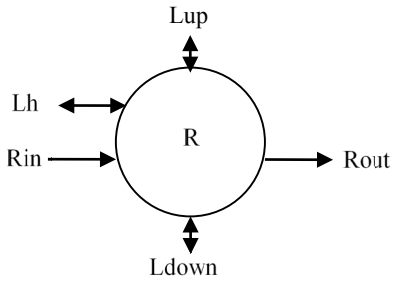

Рис. 4. Структура ресурсного элемента R
Fig. 4. The structure of the resource element R

Структура ресурсного элемента R показана на рис. 4, где Rin – запрос на ресурс, Rout – разрешение, Lh – горизонтальные связи ресурса, Lup и Ldown – связи с элементами вышележащего и нижележащего уровней (для порта). Сам ресурсный элемент R может находиться в состояниях «свободен-занят», «исправен-неисправен».

В основе статической модели лежит граф потока данных.

Предлагаемую модель можно рассматривать как полимодель, которая позволяет описывать структуру и поведение НС.

Типовые варианты постановки задачи по формированию и поддержанию в актуальном состоянии структурно-поведенческих моделей НС. В зависимости от того, что известно и что требуется узнать о НС, для каждого уровня иерархии можно выделить четыре базовых варианта постановки задачи синтеза модели НС (табл. 2).

Табл. 2. Типы задач мониторинга
Tab. 2. Types of monitoring tasks

Задача	Структурные модели	Модели поведения
A	Известны	Известны
B	Известны	Неизвестны
C	Неизвестны	Известны
D	Неизвестны	Неизвестны

Задача A «*Проверка актуальности модели*». Соответствует случаю, когда известны как статическая (структура НС), так и динамическая модели, т. е. известны модели, относящиеся ко всем уровням, представляющим интерес. В общем виде задача А может быть сформулирована следующим образом: используя известные статическую и динамическую модели НС и логи, при минимальных затратах проверить актуальность моделей и, в случае, если модели неактуальны, построить актуальную статическую и динамическую модели с заданным уровнем качества.

На практике могут появиться несколько отличающиеся варианты постановки задачи А, которые можно рассматривать как частные случаи. Можно выделить по крайней мере шесть таких частных задач:

A1 – подтвердить сам факт актуальности моделей;

A2 – откорректировать модель;

A3 – определить точку, время и причину расхождения эталонной и реальной моделей;

A4 – построить картину развития аварийной ситуации;

A5 – определить угрозы появления аварийных ситуаций и представить варианты развития;

A6 – обнаружить недокументированные активности с НС.

Задача B «*Построение модели поведения по структурной модели и логам*». В данном случае структура НС известна, но отсутствуют знания о БП, протекающих в НС. К этому классу, в частности, относится классическая задача Process Mining, когда требуется восстановить структуру БП либо с нуля, либо по известной динамической модели БП более высокого уровня. В качестве частных случаев можно выделить следующие задачи:

B1 – известны статические модели, относящиеся ко всем уровням, и динамические модели, относящиеся к верхним уровням, требуется построить недостающие модели БП;

B2 – требуется по логам отследить БП, относящиеся к определенным группам, например БП отдельных пользователей или определенного типа;

B3 – требуется быстро определить подозрительную активность БП.

Задача C «*Построение структурной модели по модели поведения и логам*». БП, протекающие в системе, известны или могут быть восстановлены по логам. Требуется определить реальную структуру НС. Здесь можно выделить следующие частные задачи:

C1 – получать актуальную информацию о постоянно изменяющейся структуре системы;

C2 – по известному поведению определить структуру «черного ящика» за минимальное число шагов;

C3 – определить неисправный элемент системы за минимальное время;

C4 – определить, какой элемент системы является «узким местом».

Например, применительно к биологическим системам требуется по внешним проявлениям определить больной орган, в технических системах это может быть вариант, когда требуется по аномалии в поведении определить причину неисправности.

Задача D «*Построение структурной и поведенческой моделей методом спуска*». Отсутствует полная информация как о структуре системы, так и о БП. Имеется только набор логов. Предполагается, что структура и семантика логов известны.

В формально общем виде задачу поддержания в актуальном состоянии структурно-поведенческих моделей НС можно сформулировать следующим образом – требуется построить искомую модель по известным моделям и контекстному знанию:

$$Mt \leftarrow \{Ms, LOG, Kc\},$$

где Mt – целевая модель; Ms – известные модели; LOG – снимаемые с НС логи; Kc – контекстное знание.

В качестве моделей выступают структурная С-модель и П-модель поведения. В рассмотрении участвуют эталонная (de jure) и реальная (de facto) модели.

Потенциальные выигрыши от использования предлагаемого подхода. Потенциальные выигрыши и дополнительные издержки, связанные с применением гибких ЦН на базе ДЦД.

Можно выделить следующие основные источники экономии и дополнительных расходов, связанных с применением гибких ЦН на базе ДЦД.

К основным свойствам гибких ЦН на базе ДЦД, позволяющим получить *экономию* по сравнению с КФС, не использующих механизмы ДЦД, можно отнести следующие:

1) возможность создавать КФС нового уровня сложности;

2) повышение эффективности функционирования за счет контекстной и контентной адаптации;

3) продление срока службы системы;

4) возможность эффективно строить линейки продуктов;

5) уменьшение затрат на поддержание НС;

6) возможность повысить эффективность производства за счет использования ЦД.

К основным *дополнительным издержкам*, можно отнести следующие моменты:

1) концептуальная сложность;

2) необходимость разрабатывать и поддерживать модели как входящие в состав ДЦД, так и другие модели;

3) вероятностный характер получения выигрыша;

4) необходимость использовать более мощные вычислительные ресурсы для реализации более сложных механизмов управления;

5) проблемы при работе в режиме реального времени.

Предлагаемый подход был использован для решения задач различного рода при построении гетерогенных распределенных КФС. Следует заметить, что не всегда целесообразно использовать описанный подход в полном объеме, и достаточно сложно найти задачу, когда это необходимо, по крайней мере, с экономической точки зрения.

В качестве примеров использования предлагаемого подхода рассмотрены примеры реальных проектов: системы технической поддержки сетей кабельного телевидения, производственной системы, умный арктический аэродром. При решении каждой из этих задач был получен положительный эффект.

Построение и поддержание моделей в актуальном состоянии. В качестве основного источника данных, используемых для построения ДЦД, выступают лог-файлы (логи), которые могут быть как получены от БП, в частности тестов, так и сгенерированы аппаратными средствами НС.

Задача построения и поддержания в актуальном состоянии рассмотренной модели может иметь достаточно много вариантов постановки (рис. 5):

Задачи построения и поддержания модели в актуальном состоянии = Тип задачи ×

× Тип модели × Режим построения модели.

Можно выделить четыре уровня изменения А-модели в процессе функционирования:

– население модели данными (population);

– проверка актуальности (conformation);

– коррекция (correction);

– конструирование (construction).

Границы между этими понятиями достаточно условны. Во всяком случае, вряд ли можно говорить о построении модели с нуля.

Чаще всего на практике приходится встречаться с такими интересами сторон:

– текущее состояние НС;

– текущая структура;

– текущее состояние БП;

– статистика по использованию аппаратной и программной платформ;

– статистика по БП.

Можно выделить два альтернативных подхода к автоматическому построению (синтезу) моделей: индуктивный и дедуктивный подходы.

При использовании индуктивного подхода модель строится «снизу», т. е. с нижних уровней, постепенно перемещаясь на вышележащие уровни.

При использовании дедуктивного подхода модель строится начиная с верхних уровней и основывается на доказательстве существования с помощью правил. Например, некоторая КФС считается исправной, если все ее элементы исправны.

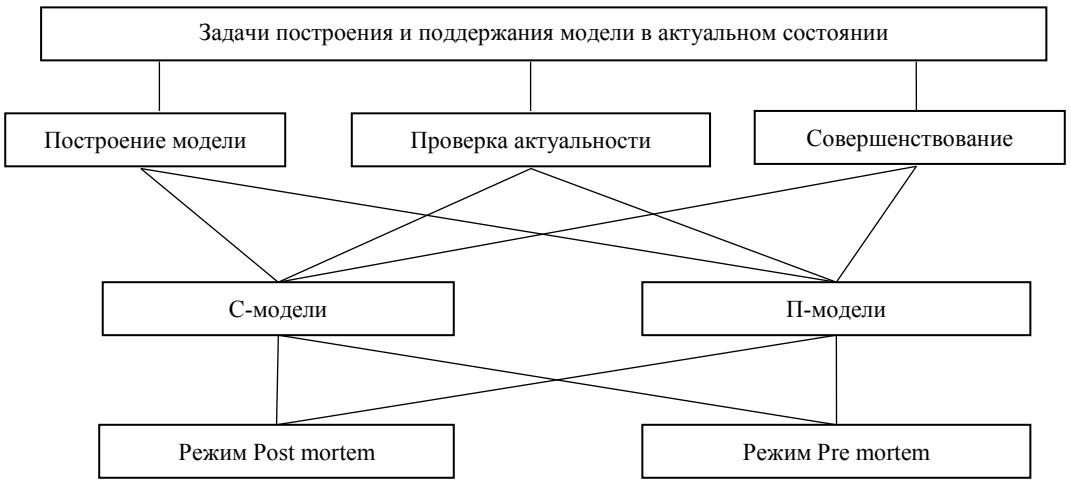

Рис. 5. Задачи построения и поддержания модели в актуальном состоянии
Fig. 5. Designing and maintaining the model in actual state issue

Примеры использования. Предлагаемый подход был использован при реализации ряда проектов в различных предметных доменах. Следует заметить, что не всегда целесообразно использовать описанный подход в полном объеме, и достаточно сложно найти задачу, когда это необходимо, по крайней мере, с экономической точки зрения.

Первый проект, связанный с построением системы мониторинга сетей кабельного телевидения, интересен своей масштабностью. В этом проекте решается задача технического обслуживания сложной распределенной системы.

Предметом второго проекта была модернизация производственной системы с мобильными сущностями, построенной на платформе IIOT. Здесь ключевой проблемой стал сбор данных на сенсорном уровне в условиях очень высоких электромагнитных помех, что потребовало введения дополнительного, туманного уровня. Использование ДЦД позволило сократить объем трафика и уменьшить время отклика как С-, так и П-модели. Для представления моделей и построения корпоративного графа знаний использовались графы знаний [14].

Заключение. В результате проведенных исследований разработан подход к построению сложных распределенных КФС, построенных на платформах туманных и граничных вычислений, с высоким уровнем архитектурной динамики. Предлагаемый подход основан на использовании динамических цифровых двойников и цифровых нитей, которые рассматриваются как архитектурные модели. Основное внимание уделяется вопросам автоматического построения (синтеза) ДЦД ВВ в распределенных КФС. Подробно проанализированы различные варианты постановки задач синтеза и алгоритмы синтеза моделей – основных элементов ДЦД ВВ.

Ключевая идея развиваемого подхода состоит в использовании динамических моделей НС, которая поддерживается в актуальном состоянии посредством обработки потока событий, поступающих в форме логов. Главным требованием, предъявляемым к модели, служит ее синтезируемость в процессе функционирования и возможность ее использования для выполнения запросов разных категорий пользователей.

Предложенный подход помогает в значительной мере решить проблему, связанную с высокой стоимостью разработки ЦД. Использование ДЦД может найти широкое применение при построении когнитивных систем.

Дальнейшие исследования предполагается проводить в двух основных направлениях: при завершении работ по созданию фреймворков и расширении сферы применения развиваемого подхода, в частности его использования применительно к социокибернетическим системам, например социальным сетям.

Список литературы

1. Zimmerman A., Schmidt R., Jain Lakhmi C. Architecting the Digital Transformation: Digital Business, Technology, Decision Support, Management. Cham, Switzerland: Springer Nature, 2021. 395 p. doi: 10.1007/978-3-030-49640-1.

2. Erder M., Pureur P., Woods E. Continuous architecture in practice: Software architecture in the age of agility and DevOps. Pearson education, Inc., 2021. 322 p.

3. Digital twin driven smart design / Fei Tao, Ang Liu, Tianliang Hu, A.Y.C. Nee. Elsevier Inc., 2020. 340 p.

4. Tarvainen P. Adaptability evaluation at software architecture level // The Open Software Engin. J. 2008. Vol. 2. P. 1–30.

5. Sanfelice R. G. Analysis and design of cyber-physical systems: A hybrid control systems approach. Boca raton, FL, USA: CRC Press, 2016. P. 3–31.

6. Perry L. Internet of Things for architects. Birmingham, UK: Packt Publishing, 2018. 515 p.

7. Jamshidi Mo. Systems of systems engineering: innovations for the 21st century. NJ, USA: John Wiley & Sons, 2009. 591 p.

8. Digital twin market size, share, trends 2027 // Markets and Markets. URL: https://www.marketsandmarkets.com/Market-Reports/digital-twin-market-225269522.html (дата обращения 25.11.2023).

9. Станкевич Л. А. Когнитивные системы и роботы. СПб.: Политех-Пресс, 2019. 630 с.

10. Leiva C. Demystifying the digital thread and digital twin concepts // Industry Week. 2016. URL: https://www.industryweek.com/technology-and-iiot/systems-integration/article/22007865/demystifying-the-digital-thread-and-digital-twin-concepts (дата обращения 25.11.2023).

11. Xiao-Feng Li. Advanced design and implementation of virtual machines. Taylor & Francis Group, LLC, 2017. 464 p. doi: 10.1201/9781315386706.

12. Rozanski N., Woods E. Software systems architecture: working with stakeholders using viewpoints and perspectives // Viewpoints. 2005. Т. 8, № 2. 576 p.

13. Mahmood Z. Fog computing concepts, frameworks and technologies. Cham, Switzerland: Springer Intern. Publishing AG, 2018. 291 p.

14. Blumauer A., Nagy H. The knowledge graphs cookbook. Recipes that work. Vienna, Austria: SemanticWebCompany, 2020. 346 p.

Информация об авторах

Ананьева Варвара Яновна – аспирант, ассистент кафедры информационные системы СПбГЭТУ «ЛЭТИ».
E-mail: varvara.spb99@mail.ru

Водяхо Александр Иванович – д-р техн. наук, профессор кафедры вычислительной техники СПбГЭТУ «ЛЭТИ».
E-mail: aivodyaho@mail.ru
https://orcid.org/0000-0002-0933-0933

Жукова Наталия Александровна – д-р техн. наук, доцент, вед. научный сотрудник СПИИРАН СПб ФИЦ РАН, 14-я линия ВО, 39, Санкт-Петербург, 199178, Россия.
E-mail: nazhukova@mail.ru
https://orcid.org/0000-0001-5877-4461

Червонцев Михаил Александрович – аспирант кафедры вычислительной техники СПбГЭТУ «ЛЭТИ».
E-mail: chervontsev.mikhail@nicetu.spb.ru

References

1. Zimmerman A., Schmidt R., Jain Lakhmi C. Architecting the Digital Transformation: Digital Business, Technology, Decision Support, Management. Cham, Switzerland: Springer Nature, 2021. 395 p. doi: 10.1007/978-3-030-49640-1.

2. Erder M., Pureur P., Woods E. Continuous architecture in practice: Software architecture in the age of agility and DevOps. Pearson education, Inc., 2021. 322 p.

3. Digital twin driven smart design / Fei Tao, Ang Liu, Tianliang Hu, A.Y.C. Nee. Elsevier Inc., 2020. 340 p.

4. Tarvainen P. Adaptability evaluation at software architecture level // The Open Software Engin. J. 2008. Vol. 2. P. 1–30.

5. Sanfelice R. G. Analysis and design of cyber-physical systems: A hybrid control systems approach. Boca raton, FL, USA: CRC Press, 2016. P. 3–31.

6. Perry L. Internet of Things for architects. Birmingham, UK: Packt Publishing, 2018. 515 p.

7. Jamshidi Mo. Systems of systems engineering: innovations for the 21st century. NJ, USA: John Wiley & Sons, 2009. 591 p.

8. Digital twin market size, share, trends 2027 // Markets and Markets. URL: https://www.marketsandmarkets.com/Market-Reports/digital-twin-market-225269522.html (data obrashhenija 25.11.2023).

9. Stankevich L. A. Kognitivnye sistemy i roboty. SPb.: Politeh-Press, 2019. 630 s. (In Russ.).

10. Leiva C. Demystifying the digital thread and digital twin concepts // Industry Week. 2016. URL: https://www.industryweek.com/technology-and-iiot/systems-integration/article/22007865/demystifying-the-digital-thread-and-digital-twin-concepts (data obrashhenija 25.11.2023).

11. Xiao-Feng Li. Advanced design and implementation of virtual machines. Taylor & Francis Group, LLC, 2017. 464 p. doi: 10.1201/9781315386706.

12. Rozanski N., Woods E. Software systems architecture: working with stakeholders using viewpoints and perspectives // Viewpoints. 2005. T. 8, № 2. 576 p.

13. Mahmood Z. Fog computing concepts, frameworks and technologies. Cham, Switzerland: Springer Intern. Publishing AG, 2018. 291 p.

14. Blumauer A., Nagy H. The knowledge graphs cookbook. Recipes that work. Vienna, Austria: SemanticWebCompany, 2020. 346 p.

Information about the authors

Varvara Ya. Ananeva – postgraduate student, assistant of the Department of Information Systems of Saint Petersburg Electrotechnical University.
E-mail: varvara.spb99@mail.ru

Aleksandr I. Vodyaho – Dr Sci. (Eng.), Professor of the Department of Computer Technology of Saint Petersburg Electrotechnical University.
E-mail: aivodyaho@mail.ru
https://orcid.org/0000-0002-0933-0933

Natalia A. Zhukova – Dr Sci. (Eng.), Associate Professor, leading researcher, SPIIRAS of St. Petersburg Federal Research Center of the Russian Academy of Sciences, 14st line, 39, Vasilievsky Island, Saint Petersburg, 199178, Russia.
E-mail: nazhukova@mail.ru
https://orcid.org/0000-0001-5877-4461

Mikhail A. Chervoncev – postgraduate student, Department of Computer Technology of Saint Petersburg Electrotechnical University.
E-mail: chervontsev.mikhail@nicetu.spb.ru

Статья поступила в редакцию 28.12.2023; принята к публикации после рецензирования 13.01.2024; опубликована онлайн 25.03.2024.

Submitted 28.12.2023; accepted 13.01.2024; published online 25.03.2024.