УДК 004.4’23
DOI: 10.17586/0021-3454-2024-67-9-741-750

МЕТОД ДИНАМИЧЕСКОЙ АКТУАЛИЗАЦИИ МОДЕЛИ ВЗАИМОДЕЙСТВИЯ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ ВО ВСТРОЕННЫХ СИСТЕМАХ

А. А. Гончаров*, С. В. Быковский

Университет ИТМО, Санкт-Петербург, Россия
** aagoncharov@itmo.ru*

Аннотация. Рассматривается метод динамической актуализации формальной модели параллельных процессов, который предназначен для отладки и верификации программного обеспечения микроконтроллеров в процессе натурных испытаний. Предложенный метод основан на применении методов интеллектуального анализа процессов и изначально отличается от предыдущих подходов тем, что позволяет записывать наблюдаемое поведение системы в формальную модель и обновить эту модель в режиме реального времени в процессе функционирования системы. Такой подход позволяет существенно сократить затраты ресурсов памяти на журналирование событий, поддерживать причинно-следственную связь между ними, производить наблюдение за системой в случаях, когда доступ к ней ограничен в течение продолжительного времени, а также строить модели процессов для распределенных систем в режиме реального времени. Метод, воплощенный в виде библиотеки на языке С, был реализован в виде совокупности предварительно подготовленных таблиц, представляющих собой динамически актуализируемую модель процессов системы в форме событийного графа с частотными характеристиками, обновляемыми за счет поступления информации о событиях в системе. Приведена формула для оценки необходимых ресурсов для целевых платформ, а также даны указания по применению разработанного инструментария.

Ключевые слова: *верификация, формальная модель процесса, встраиваемые системы, микроконтроллеры, интеллектуальный анализ процессов*

Ссылка для цитирования: *Гончаров А. А., Быковский С. В.* Метод динамической актуализации модели взаимодействия параллельных процессов во встроенных системах // Изв. вузов. Приборостроение. 2024. Т. 67, № 9. С. 741–750. DOI: 10.17586/0021-3454-2024-67-9-741-750.

METHOD OF DYNAMIC UPDATING OF THE INTERACTION MODEL OF PARALLEL PROCESSES IN EMBEDDED SYSTEMS

A. A. Goncharov*, S. V. Bykovsky

ITMO University, St. Petersburg, Russia,
** aagoncharov@itmo.ru*

Abstract. A method for dynamic updating of a formal model of parallel processes, intended for debugging and verification of microcontroller software during field testing, is considered. The proposed method is based on the application of methods of process mining and initially differs from previous approaches in that it allows recording the observed behavior of the system in a formal model and updating this model in real time during the operation of the system. This approach allows to significantly reduce the memory resource costs for event logging, maintain the cause-and-effect relationship between them, monitor the system in cases where access to it is limited for a long time, and build process models for distributed systems in real time. The method, embodied in the form of a library in the C language, is implemented as a set of pre-prepared tables representing a dynamically updated model of the system processes in the form of an event graph with frequency characteristics updated due to the receipt of information about events in the system. A formula for assessing the necessary resources for target platforms is given, and instructions for using the developed toolkit are given.

Key words: verification, formal process model, embedded systems, microcontrollers, process mining

For citation: Goncharov A. A., Bykovsky S. V. Method of dynamic updating models of the interaction model of parallel processes in embedded systems. 2024. Vol. 67, N 9. P. 741–750 (in Russian). DOI: 10.17586/0021-3454-2024-67-9-741-750.

Введение. Промышленные системы управления, бытовая техника, медицинское оборудование, автомобили, а также различные современные устройства — область применения встроенных систем на базе микроконтроллеров [1, 2]. Программное обеспечение таких систем должно соответствовать спецификациям, так как даже малейшие ошибки в коде могут привести к авариям.

Одним из эффективных способов проверки корректности программного обеспечения встроенных систем является формальная верификация [3]. Особый интерес представляет проверка на соответствие спецификации формальной модели в процессе работы системы.

Интеллектуальный анализ процессов (process mining) — это технология, основанная на сборе и обработке данных журналов событий системы, позволяющая анализировать процессы, восстанавливать их формальные модели, обнаруживать несоответствия ожидаемому поведению системы [4–6]. Использование такого подхода на целевых платформах во встроенных системах ограничено требованиями по относительно высокой производительности алгоритмов, а также необходимостью накопления объемных журналов событий для последующей обработки. Несмотря на эти сложности, некоторые методы интеллектуального анализа процессов можно применить для анализа встроенных систем.

Обзор публикаций. Возможность применения методов интеллектуального анализа процессов в области встроенных систем рассматривается во множестве исследований. Так, в работе [7] предлагается использовать эту технологию для получения актуальной информации о текущем состоянии наблюдаемых сложных киберфизических систем. Результат в виде моделей формируется на отдельном вычислительном сервере после сбора журналов событий со всех компонентов. Данный подход апробирован на реальных системах и рекомендован для киберфизических систем средней и высокой сложности. В [8] результат, полученный посредством интеллектуального анализа процессов, дополняется проверкой с помощью формул темпоральной логики устойчивости киберфизических систем к сбоям ресурсов. Подход к разработке крупномасштабных когнитивных киберфизических систем, характеризующихся высоким уровнем структурной, функциональной и архитектурной динамики, рассматривается в работе [9]. Основная идея заключается в использовании нескольких современных парадигм — непрерывной архитектуры, гибкой архитектуры, цифровых двойников и цифровых потоков. Важным моментом является предварительное заложение на этапе разработки системы возможности динамической актуализации системы во время работы.

В [10–12] интеллектуальный анализ процессов используется для проверки соответствия применительно к различным сферам: роботизированным механизмам, промышленным системам управления и интернету вещей. Возможность автоматизация анализа моделей процессов, полученных с использованием интеллектуального анализа процессов, показана в [13]. Работы [14, 15] посвящены использованию систем анализа моделей для сложных эксплуатируемых систем: солнечных электростанций и европейской системы управления железнодорожным движением. Возможность применения методов интеллектуального анализа процессов для выявления аномального поведения индустриальных систем и обнаружения кибератак на эти системы продемонстрирована в [16].

В работе [17] авторами произведен сравнительный обзор алгоритмов интеллектуального анализа процессов по различным параметрам, например, таким как возможности обработки циклов различной длины, обнаружение повторяющихся задач, учет частоты событий, работа с „шумными“ журналами, скорость обработки журналов событий с относительно большим количеством строк. Был выбран индуктивный алгоритм для поиска общих зависимостей между действиями, который для обнаружения частотных характеристик дополнен алгоритмом выравнивания. Такой подход позволил авторам получить корректный событийный граф процессов.

На основе воспроизведения объектов достигнута полная автоматизация проверки корректности поведения системы.

Представленный обзор показывает возможность и актуальность использования методов интеллектуального анализа процессов для современных встроенных систем. Важно отметить, что в рассмотренных работах анализ данных, получаемых от различных компонентов системы, осуществляется вне целевой платформы, обычно на отдельном вычислительном устройстве оператора. В настоящей статье предложен метод динамической актуализации модели параллельных процессов встроенных систем, реализация которого полностью возможна на целевой платформе.

Предлагаемый метод. Анализ существующих работ показал отсутствие инструментов, способных обеспечить динамическую актуализацию модели процессов без использования внешнего относительно встраиваемой системы вычислительного устройства. Встраиваемые системы могут использоваться длительное время автономно и, соответственно, не иметь возможности передавать накопленный журнал событий для расчета модели. Поэтому в рамках данного исследования предлагается метод, позволяющий динамически обновлять модель процессов в режиме реального времени за счет данных, получаемых от встраиваемой системы. Хранение модели планируется также на целевой платформе. Таким образом, можно обеспечить хранение и обновление модели в рамках целевой платформы с периодической передачей полученной модели для дальнейшего анализа на другом вычислительном устройстве.

В первую очередь был разработан метод анализа модели процессов одиночного устройства, позволяющий отказаться от накопления журнала событий и последующей длительной обработки. Вместо классического подхода с постобработкой журналов событий предлагается создать на этапе инициализации устройства модель процессов в виде таблиц, учитывающих все возможные переходы между событиями, и по мере накопления событий, поступающих с компонентов системы, обновлять данные в таблицах.

Постепенно заполняемые таблицы представляют собой динамически актуализируемую модель процессов системы в виде событийного графа с частотными характеристиками, обновляемыми за счет поступления информации о событиях в системе. Матрица смежности графа, иллюстрирующая систему переходов между событиями (events), имеет следующий вид:

$$\text{Events} = \begin{bmatrix} e_1 \to e_1 & e_1 \to e_2 & e_1 \to e_3 & \dots & e_1 \to e_n \\ e_2 \to e_1 & e_2 \to e_2 & e_2 \to e_3 & \dots & e_2 \to e_n \\ e_3 \to e_1 & e_3 \to e_2 & e_3 \to e_3 & \dots & e_3 \to e_n \\ \dots & \dots & \dots & \dots & \dots \\ e_n \to e_1 & e_n \to e_2 & e_n \to e_3 & \dots & e_n \to e_n \end{bmatrix},$$

где e_i — некое событие в системе.

Предлагается создать несколько таблиц: таблицу, данные для которой должен ввести разработчик на этапе инициализации устройства, с максимально допустимым временем между переходами для учета недопустимых по времени переходов, например долгого ответа устройств, датчиков и т. д.; таблицу с переходами, время которых не превышает допустимое; таблицу с переходами, время которых превышает допустимое. Совокупность таких таблиц позволит не только получать актуализированную модель процессов и вести подсчет частотных характеристик между событиями, но и устанавливать нежелательное поведение системы в части скорости работы между событиями.

Таким образом, разработчик, использующий предлагаемый метод, может точно оценить необходимые затраты ресурсов системы по объему памяти, а также по необходимой вычислительной мощности.

Для разработанного метода основными временными затратами являются:

— затраты на добавление в таблицы нового события в системе — постоянная величина вне зависимости от количества ранее добавленных событий, состоящая из следующих операций:

извлечение из таблицы временных ограничений (чтение из массива и перемещение в переменную — две инструкции); сравнение времени между переходами с ранее извлеченным временем (сравнение и переход в зависимости от результата сравнения — две инструкции); инкрементирование ячейки таблицы с корректными или некорректными по времени переходами в зависимости от результата сравнения (чтение из массива, инкрементирование прочитанного массива и перемещение нового значения обратно в массив — три инструкции);

— выгрузка актуальной модели процессов на операторский персональный компьютер через проводной или беспроводный интерфейс (время изменяется пропорционально росту количества переходов между событиями и заполняемости ячеек таблиц).

На рис. 1 представлена схема, отображающая разработанный авторами метод динамической актуализации модели поведения системы для анализа процессов одиночного устройства встраиваемой системы.

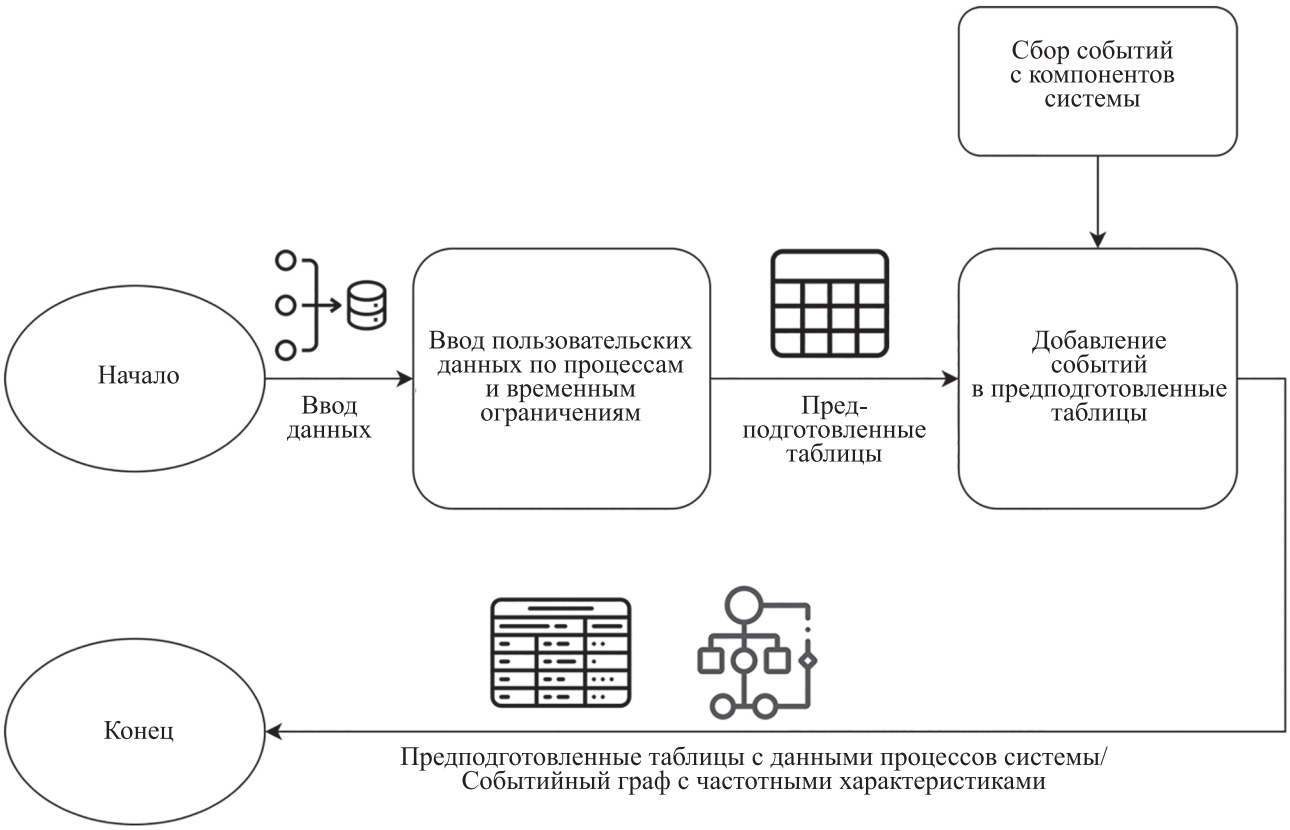

Рис. 1

Основные преимущества разработанного метода — прогнозируемость в части использования ресурсов памяти и требуемой производительности для обновления данных в таблицах, а также возможность контролировать изменения модели во время работы системы, т. е. возможность наблюдать динамику процессов. Недостатками подхода являются: отсутствие журнала событий системы и, соответственно, невозможность построения модели поведения системы с помощью других алгоритмов, а также невозможность получения всей цепочки событий от запуска системы до возникновения сбоя.

На рис. 2 приведена схема предлагаемого метода актуализации модели процессов во встроенных системах для параллельных процессов. Данный метод является расширением для множества устройств представленного выше метода динамической актуализации модели (см. рис. 1). В качестве точек объединения моделей предлагается использовать события взаимодействия между устройствами.

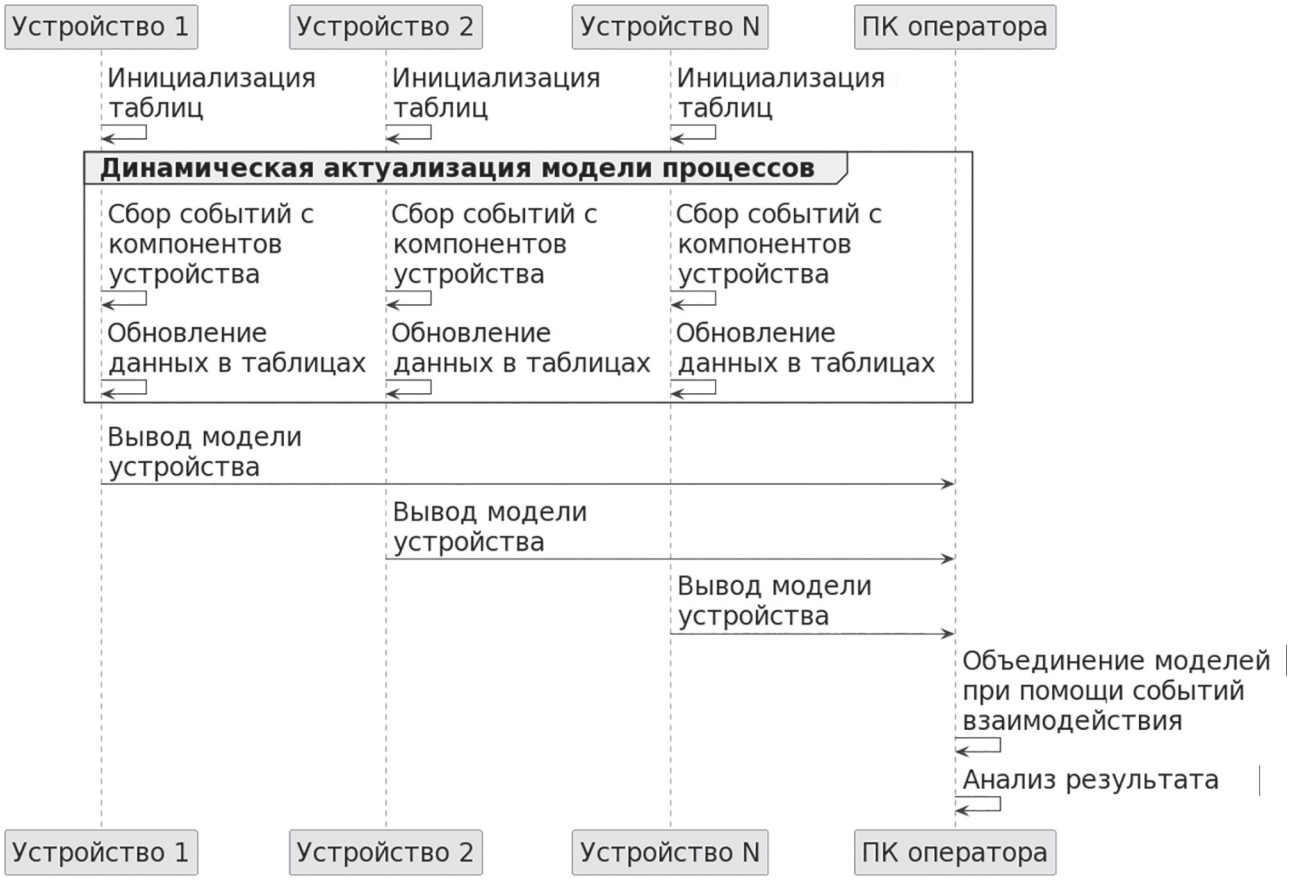

Рис. 2

Ниже для примера представлены матрицы смежности графов и матрица взаимодействия двух устройств:

$$\text{Events 1} = \begin{bmatrix} e_{1\_1} \to e_{1\_1} & e_{1\_1} \to e_{1\_2} & \dots & e_{1\_1} \to e_{1\_n} \\ e_{1\_2} \to e_{1\_1} & e_{1\_2} \to e_{1\_2} & \dots & e_{1\_2} \to e_{1\_n} \\ e_{1\_3} \to e_{1\_1} & e_{1\_3} \to e_{1\_2} & \dots & e_{1\_3} \to e_{1\_n} \\ \dots & \dots & \dots & \dots \\ e_{1\_n} \to e_{1\_1} & e_{1\_n} \to e_{1\_2} & \dots & e_{1\_n} \to e_{1\_n} \end{bmatrix}, \quad (1)$$

$$\text{Events 2} = \begin{bmatrix} e_{2\_1} \to e_{2\_1} & e_{2\_1} \to e_{2\_2} & \dots & e_{2\_1} \to e_{2\_n} \\ e_{2\_2} \to e_{2\_1} & e_{2\_2} \to e_{2\_2} & \dots & e_{2\_2} \to e_{2\_n} \\ e_{2\_3} \to e_{2\_1} & e_{2\_3} \to e_{2\_2} & \dots & e_{2\_3} \to e_{2\_n} \\ \dots & \dots & \dots & \dots \\ e_{2\_n} \to e_{2\_1} & e_{2\_n} \to e_{2\_2} & \dots & e_{2\_n} \to e_{2\_n} \end{bmatrix}, \quad (2)$$

$$\text{Events Cooperation 1\&2} = \begin{bmatrix} e_{1\_1} \leftrightarrow e_{2\_1} & e_{1\_1} \leftrightarrow e_{2\_2} & \dots & e_{1\_1} \leftrightarrow e_{2\_n} \\ e_{1\_2} \leftrightarrow e_{2\_1} & e_{1\_2} \leftrightarrow e_{2\_2} & \dots & e_{1\_2} \leftrightarrow e_{2\_n} \\ e_{1\_3} \leftrightarrow e_{2\_1} & e_{1\_3} \leftrightarrow e_{2\_2} & \dots & e_{1\_3} \leftrightarrow e_{2\_n} \\ \dots & \dots & \dots & \dots \\ e_{1\_n} \leftrightarrow e_{2\_1} & e_{1\_n} \leftrightarrow e_{2\_2} & \dots & e_{1\_n} \leftrightarrow e_{2\_n} \end{bmatrix}. \quad (3)$$

Размерность матриц (1) и (2) определяется количеством фиксируемых событий для устройства, а размерность матрицы (3) — количеством событий взаимодействия между устройствами:

например, если устройства только периодически обмениваются сообщениями в формате запрос-ответ, такую матрицу можно представить в виде $[e_{1\_1} \rightarrow e_{2\_1}]$. Вид (3) матрицы Events Cooperation 1&2 представлен для множества событий взаимодействия между устройствами. В случае когда система представляет собой множество устройств, необходимо построить матрицы взаимодействия для каждой пары из множества. События взаимодействия определяются разработчиком, внедряющим предлагаемый метод, а частотные характеристики для событий взаимодействия рассчитываются в соответствии с данными, полученными из моделей устройств.

Оценим необходимые затраты ресурсов памяти для использования во встраиваемых системах. Для начала представим формулы, позволяющие разработчику на этапе проектирования архитектуры системы рассчитать объем памяти, требуемой для хранения таблиц для каждого устройства. Так, минимальное количество потенциально фиксируемых переходов в событийном графе при заданных таблицах, их размерности и количестве фиксируемых событий определяется как

$$K_{\min} = a \cdot 2^{8 \cdot S}, \tag{4}$$

где a — количество активных переходов, т. е. переходов с максимальными частотами актуализации, а именно количество ячеек в таблице переходов, значения которых выше заданного порога, определяющего высокочастотные переходы (порог определяется в соответствии с особенностями проекта); S — размерность одной ячейки в таблице, байт.

Тогда требуемый объем памяти можно вычислить следующим образом:

$$V = TE^2S, \tag{5}$$

где T — количество предварительно подготовленных таблиц, например таблицы временных ограничений, таблицы корректных и некорректных переходов, таблицы взаимодействия между устройствами; E — количество фиксируемых событий в системе.

На основе формулы (4) можно вывести усредненное время работы системы до полного исчерпания памяти в предварительно созданных таблицах, зная усредненную частоту f возникновения событий в системе:

$$\bar{t} = \frac{K_{\min}}{f} = \frac{a \cdot 2^{8 \cdot S}}{f}.$$

Оценка эффективности метода. Для апробации предлагаемый метод был промоделирован на персональном компьютере. Модели независимых устройств были представлены в виде процессов, запущенных в отдельных потоках с периодической синхронизацией из отдельного потока. Для наглядности были созданы пять потоков с двумя видами типовых устройств, заданы временные ограничения, добавлена периодическая задержка для фиксации невыполнения заданного временного ограничения. Каждое независимое устройство представляет собой два блока: генератор событий, имитирующий некоторое устройство, и наблюдатель, используемый для реализации предлагаемого метода (монитор). Упрощенная схема взаимодействия имитатора устройства и персонального компьютера оператора представлена на рис. 3.

Рис. 3

На рис. 4 представлен результат работы метода — объединенный событийный граф, состоящий из событийных графов потоков, объединенных событиями взаимодействия потоков. События взаимодействия показаны синей линией и для упрощения восприятия объединенного графа соединены попарно между потоками, невыполнение требований по времени между событиями отмечено пунктирной линией.

На основе спецификации проекта разработчик может выделить ряд допустимых и недопустимых переходов между событиями для каждого из устройств. В случае выполнения некорректных переходов легко обнаружить отклонения в поведении системы. Пример событийного графа с некорректными переходами (отмечены красной линией) продемонстрирован на рис. 5; граф получен путем добавления случайных переходов для генератора событий во время работы системы.

Таким образом, предлагаемый метод позволяет проверить требования к системе, регламентируемые спецификацией [18].

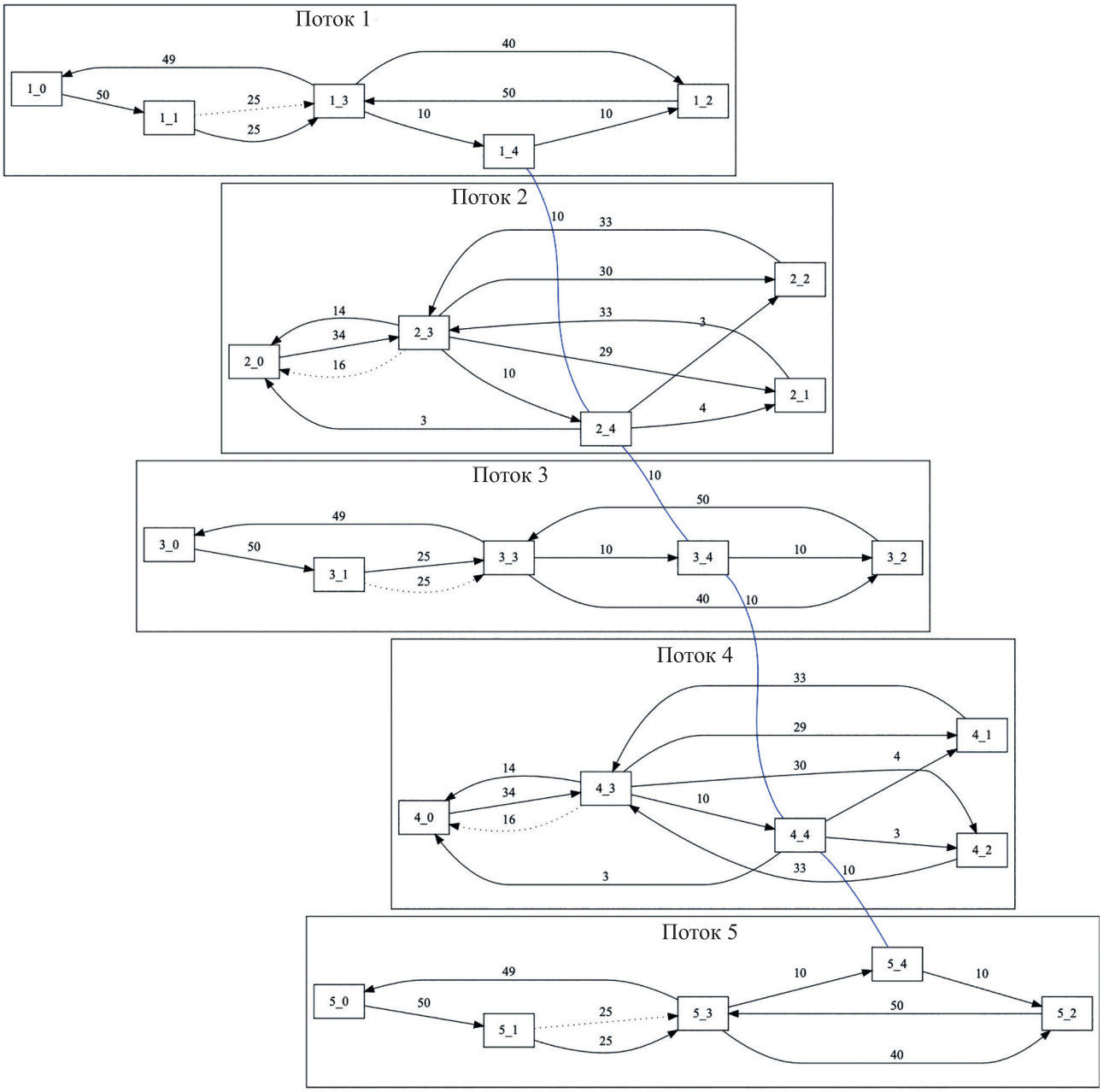

Рис. 4

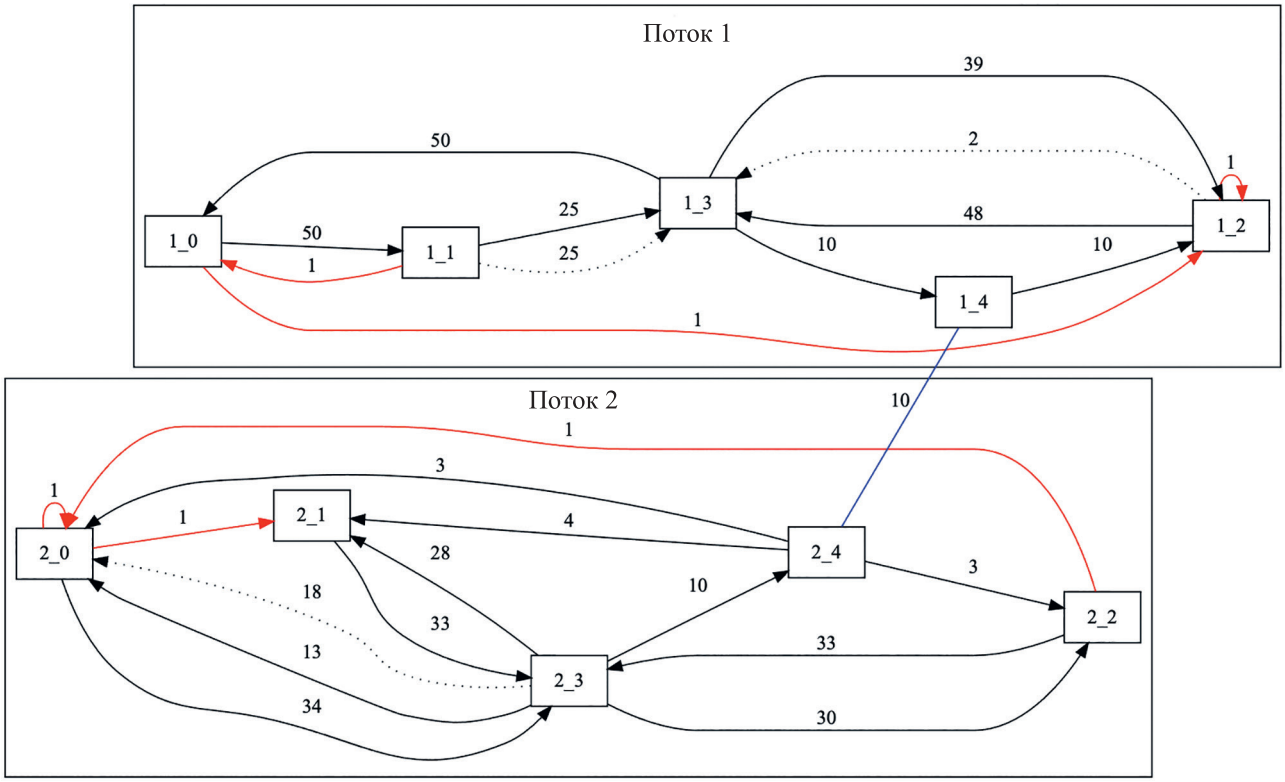
Поток 1
1_0
1_1
1_2
1_3
1_4
50
50
1
25
25
39
2
1
48
10
10
1
Поток 2
2_0
2_1
2_2
2_3
2_4
1
1
3
1
4
28
18
33
13
34
10
3
33
30
10

Рис. 5

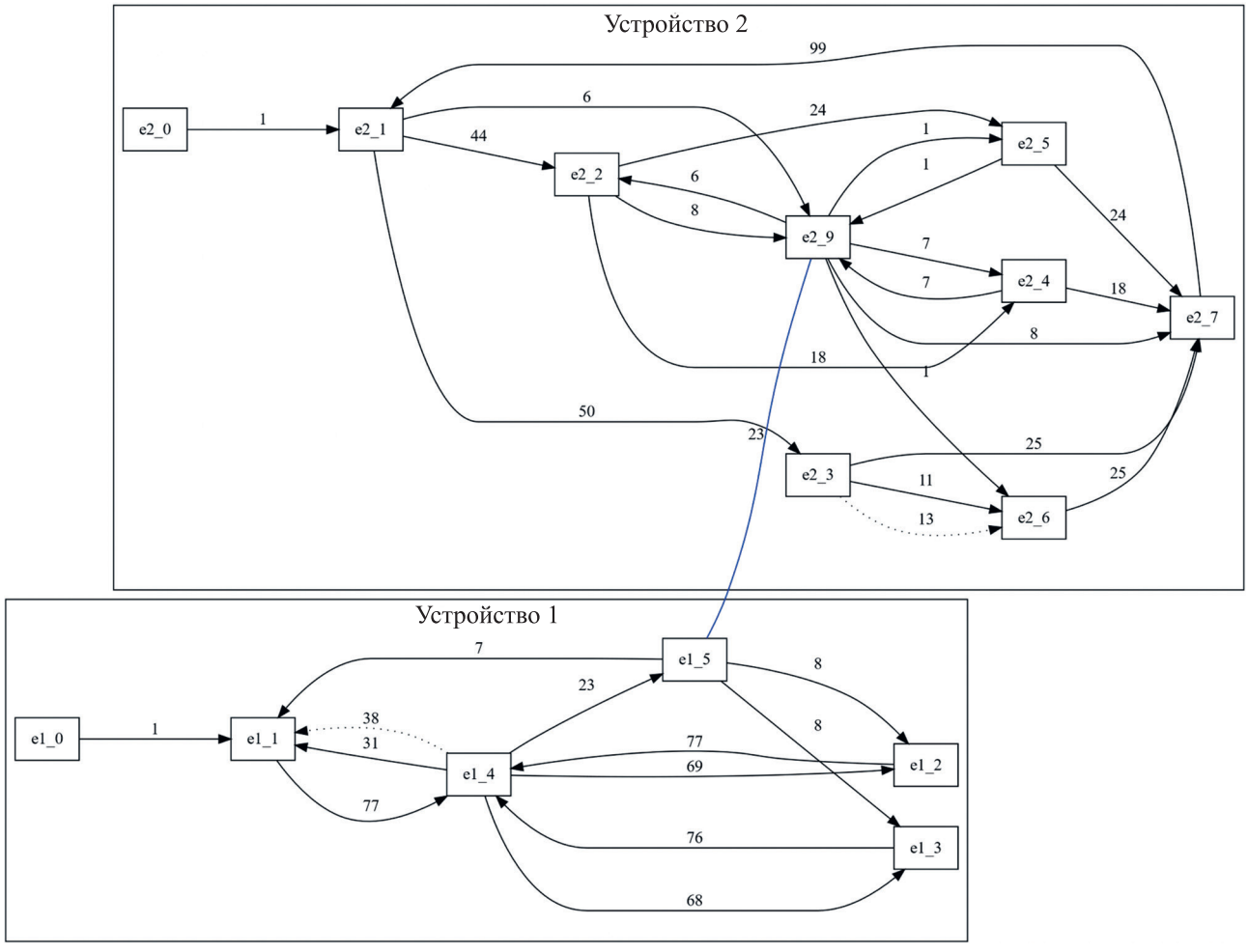
Устройство 2
e2_0
e2_1
e2_2
e2_3
e2_4
e2_5
e2_6
e2_7
e2_9
1
99
6
44
24
1
1
6
8
7
7
18
8
18
1
50
23
25
11
13
25
Устройство 1
e1_0
e1_1
e1_2
e1_3
e1_4
e1_5
1
7
38
31
77
23
77
69
8
8
76
68

Рис. 6

Апробация метода была проведена также на целевых платформах с использованием библиотеки на языке С, реализующей представленный метод [19]. На рис. 6 продемонстрирован результат апробации для двух взаимодействующих устройств на базе микроконтроллеров STM32F103RB. С некоторой периодичностью устройство 1 отправляет данные на устройство 2, это взаимодействие между устройствами соответствует паре событий e_{1_5}–e_{2_9}. Каждое из устройств по отдельности передает модели процессов на ПК оператора, где впоследствии происходит объединение моделей через события взаимодействия. Необходимый объем памяти для устройств 1 и 2 можно вычислить следующим образом: $V_1 = TE^2S = 3 \cdot 6^2 \cdot 2 = 216$ байт, $V_2 = TE^2S = 3 \cdot 9^2 \cdot 2 = 486$ байт. Каждая табличная ячейка размерностью 2 байта позволяет сохранить до переполнения количество переходов между событиями 2^{16}. При необходимости более длительного времени наблюдения можно увеличить размерность ячейки до 4 байт.

Представленный метод можно использовать для всех типов задач, решаемых встраиваемыми системами, — управления, вычислительных, коммуникационных, комбинированных.

Заключение. Разработан и апробирован метод динамической актуализации модели процессов во встроенных системах для параллельных процессов в виде реализации библиотеки на языке С для программного обеспечения микроконтроллеров.

Преимуществом предлагаемого метода является фиксированное использование памяти микроконтроллера, объем которой разработчик может вычислить на этапе внедрения метода в архитектуру системы за счет отказа от журналирования событий.

Представленный метод позволяет масштабировать область исследования встраиваемой системы, например можно строить модели для отдельных устройств, выделяя в качестве событий системы атомарные операции микроконтроллера или более высокоуровневые операции, а также строить модели для нескольких устройств с необходимой степенью погружения относительно операций и команд.

СПИСОК ЛИТЕРАТУРЫ

1. *Pivoto D. G. S.* et al. Cyber-physical systems architectures for industrial internet of things applications in Industry 4.0: A literature review //Journal of manufacturing systems. 2021. Vol. 58. P. 176–192.
2. *Hamzah M.* et al. Distributed Control of Cyber Physical System on Various Domains: A Critical Review // Systems. 2023. Vol. 11, N 4. P. 208.
3. *Pola G., Di Benedetto M. D.* Control of cyber-physical-systems with logic specifications: A formal methods approach // Annual Reviews in Control. 2019. Vol. 47. P. 178–192.
4. *Souza J. T. de* et al. Data mining and machine learning in the context of sustainable evaluation: A literature review // IEEE Latin America Trans. 2019. N 3. P. 372–382.
5. *Taranto V., G.* et al. Algorithms and software for data mining and machine learning: a critical comparative view from a systematic review of the literature // The Journal of Supercomputing. 2021. N 7. P. 11481–11513.
6. *Cheng T.* et al. Spatio-temporal data mining // Handbook of Regional Science. 2021. P. 1691–1709.
7. *Vodyaho A.* et al. Model Based Approach to Cyber–Physical systems status Monitoring // Computers. 2020. Vol. 9, N 2. P. 47.
8. *Hsieh F. S.* Temporal Analysis of Influence of Resource Failures on Cyber-Physical Systems Based on Discrete Timed Petri Nets // Applied Sciences. 2021. Vol. 11. N 14. P. 6469.
9. *Chervontsev M.* et al. Use of Dynamic Models in Cognitive Cyber-Physical Systems // Engineering Proceedings. 2023. Vol. 33, N 1. P. 14.
10. *Nicoleta T. C.* Process Mining on a Robotic Mechanism // Intern. Conf. on Software Testing, Verification and Validation Workshops (ICSTW). IEEE, 2021. P. 205–212.
11. *Xavier M.* et al. Process mining in industrial control systems // 20th Intern. Conf. on Industrial Informatics (INDIN). IEEE, 2022. P. 1–6.
12. *Premchaiswadi W.* et al. Using IoT and Mobile Robots to Model and Analyze Work Processes with Process Mining Techniques // Progress in Applied Science and Technology. 2023.
13. *Zakarija I., Škopljanac-Mačina F., Blašković B.* Automated simulation and verification of process models discovered by process mining // Automatika: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije. 2020. Vol. 61, N 2. P. 312–324.

14. *Shakya S.* A self monitoring and analyzing system for solar power station using IoT and data mining algorithms // Journal of Soft Computing Paradigm. 2021. Vol. 3, N 2. P. 96–109.
15. *Vitale F*. Run-Time Anomaly Detection with Process Mining: Methodology and Railway System Compliance Case-Study: Extended Abstracts of Master's Thesis. Linnaeus Univ., Sweden, 2021.
16. *Myers D*. et al. Anomaly detection for industrial control systems using process mining // Computers & Security. 2018. Vol. 78. P. 103–125.
17. *Гончаров А. А., Быковский С. В*. Метод восстановления модели процессов во встроенных системах по журналу событий // Изв. вузов. Поволжский регион. Технические науки. 2023. № 3. С. 5–17. DOI: 10.21685/2072-3059-2023-3-1.
18. *Карпов Ю. Г*. Model checking. Верификация параллельных и распределенных программных систем. СПб: БХВ-Петербург, 2010.
19. Table_miner [Электронный ресурс]: https://github.com/GoncharovAleshka/micropm, 09.03.2024.

СВЕДЕНИЯ ОБ АВТОРАХ

Алексей Андреевич Гончаров — аспирант; Университет ИТМО, факультет программной инженерии и компьютерной техники; E-mail: aagoncharov@itmo.ru

Сергей Вячеславович Быковский — канд. техн. наук, доцент; Университет ИТМО, факультет программной инженерии и компьютерной техники; E-mail: bsv.serg@gmail.com, sergei_bykovskii@itmo.ru

Поступила в редакцию 21.03.2021; одобрена после рецензирования 31.05.2024; принята к публикации 23.07.2024.

REFERENCES

1. Pivoto D.G.S. et al. *Journal of manufacturing systems*, 2021, vol. 58, pp. 176–192.
2. Hamzah M. et al. *Systems*, 2023, no. 4(11), pp. 208.
3. Pola G., Di Benedetto M.D. *Annual Reviews in Control*, 2019, vol. 47, pp. 178–192.
4. de Souza J.T. et al. *IEEE Latin America Transactions*, 2019, no. 03(17), pp. 372–382.
5. Taranto-Vera G. et al. *The Journal of Supercomputing*, 2021, vol. 77, pp. 11481–11513.
6. Cheng T. et al. *Handbook of regional science*, 2021, pp. 1691–1709.
7. Vodyaho A. et al. *Computers*, 2020, no. 9, pp. 47.
8. Hsieh F.S. *Applied Sciences*, 2021, no. 14(11), pp. 6469.
9. Chervontsev M. et al. *Engineering Proceedings*, 2023, no. 1(33), pp. 14.
10. Nicoleta T.C. *International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, IEEE, 2021, pp. 205–212.
11. Xavier M. et al. *20th International Conference on Industrial Informatics (INDIN)*, IEEE, 2022, pp. 1–6.
12. Premchaiswadi W. et al. *Progress in Applied Science and Technology*, 2023.
13. Zakarija I., Škopljanac-Mačina F., Blašković B. *Automatika: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije*, 2020, no. 2(61), pp. 312–324.
14. Shakya S. *Journal of Soft Computing Paradigm*, 2021, no. 2(3), pp. 96–109.
15. Vitale F. Run-Time Anomaly Detection with Process Mining: Methodology and Railway System Compliance Case-Study, Extended Abstracts of Master's Thesis. Linnaeus Univ., Sweden, 2021.
16. Myers D. et al. *Computers & Security*, 2018, vol. 78, pp. 103–125.
17. Goncharov A.A., Bykovskii S.V. *University Proceedings. Volga Region. Technical Sciences*, 2023, no. 3, pp. 5–17, DOI: 10.21685/2072-3059-2023-3-1. (in Russ.)
18. Karpov Yu.G. *Model shecking. Verifikatsiya parallel'nykh i raspredelennykh programmnykh sistem* (MODEL CHECKING. Verification of Parallel and Distributed Software Systems), St. Petersburg, 2010. (in Russ.)
19. Table_miner, https://github.com/GoncharovAleshka/micropm

DATA ON AUTHORS

Alexey A. Goncharov — Post-Graduate Student; ITMO University, Faculty of Software Engineering and Computer Systems; E-mail: aagoncharov@itmo.ru

Sergey V. Bykovsky — PhD, Associate Professor; ITMO University, Faculty of Software Engineering and Computer Systems;
E-mail: bsv.serg@gmail.com, sergei_bykovskii@itmo.ru

Received 21.03.2021; approved after reviewing 31.05.2024; accepted for publication 23.07.2024.