https://doi.org/10.31891/2219-9365-2025-84-32
УДК 004.451

МОСТОВИЙ Сергій
Хмельницький національний університет
https://orcid.org/0000-0002-9505-3206
e-mail: serhii.mostovyi@khmnu.edu.ua
ПИРЧ Олена
Хмельницький національний університет
https://orcid.org/0009-0002-6485-4462
e-mail: oleksukolena@gmail.com

ПРОГНОЗУВАННЯ СТАНУ ВЗАЄМОБЛОКУВАННЯ ПРОЦЕСІВ У КОМП'ЮТЕРНИХ СИСТЕМАХ

У статті розглянуто задачу прогнозування стану взаємоблокування процесів у комп'ютерних системах, що функціонують в умовах паралельного виконання процесів та конкурентного доступу до спільних ресурсів. Обґрунтовано обмеження класичних підходів до запобігання, уникнення, виявлення та відновлення після взаємоблокування, оскільки значна частина з них потребує попереднього знання майбутніх запитів процесів або реагує вже після формування небезпечного стану. Запропоновано метод прогнозування взаємоблокування, який поєднує формування сигнатури процесу, модель поточного стану комп'ютерної системи, автоматизовану побудову бази нечітких правил і подальше оцінювання ризику переходу процесів у граничний або взаємозаблокований стан. Для виділення типових ресурсних конфігурацій використано нечітку кластеризацію Fuzzy C-Means, що дає змогу враховувати поступовий характер переходу між нормальним, граничним і небезпечним режимами функціонування. Запропонований підхід дозволяє виконувати прогнозування у моменти ненадання ресурсу за запитом процесу, зменшувати обчислювальні витрати та формувати підстави для превентивного керування ресурсами комп'ютерної системи до моменту повної втрати працездатності процесів.

Ключові слова: комп'ютерна система, процес, взаємоблокування, прогнозування взаємоблокування, сигнатура процесу, ресурсний стан, нечіткі правила, Fuzzy C-Means, нечітка кластеризація.

MOSTOVYI Serhii, PYRCH Olena
Khmelnitskyi National University

PREDICTING THE DEADLOCK STATE OF PROCESSES IN COMPUTER SYSTEMS

The article considers the problem of predicting the deadlock state of processes in computer systems operating under conditions of parallel process execution, dynamic resource allocation, and competitive access to memory, files, input/output devices, synchronization objects, transactions, and database records. The relevance of the study is determined by the fact that deadlock is a critical state in which several processes hold part of the required resources and simultaneously wait for resources occupied by other processes. This leads to reduced performance, disruption of service availability, and the impossibility of further execution of blocked processes without external intervention. It is shown that classical approaches to deadlock prevention, avoidance, detection, and recovery have certain limitations, since they often require prior knowledge of future process requests or respond only after a dangerous state has already been formed. A method for predicting the deadlock state of processes is proposed. It combine the formation of a process signature, a model of the current state of the computer system, automated construction of a fuzzy rule base, and assessment of the risk of process transition into a boundary or deadlocked state. The process signature is represented as a structured description of identification, temporal, resource, and behavioral characteristics. This makes it possible to take into account not only the fact of waiting for a resource, but also the context of the process interaction with other processes and system resources. Fuzzy C-Means clustering is used to form the rule base, which ensures the identification of typical resource states and takes into account the gradual transition between normal execution, regular waiting, boundary state, and deadlock. The proposed method is activated at the moments when a requested resource is not granted to a process. This reduces computational costs and focuses the analysis on potentially dangerous situations. The obtained results create a basis for preventive resource management in a computer system, early detection of deadlock prerequisites, and improvement of the stability of the computing environment.

Keywords: computer system, process, deadlock, deadlock prediction, process signature, resource state, fuzzy rules, Fuzzy C-Means, fuzzy clustering.

Стаття надійшла до редакції / Received 01.11.2025
Прийнята до друку / Accepted 02.12.2025

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Інтенсивне паралельне виконання процесів у сучасних обчислювальних системах підвищує ризик конфліктів доступу до спільних ресурсів, найнебезпечнішим виявом яких є взаємоблокування. Воно спричиняє критичне зниження продуктивності й порушення доступності сервісів без зовнішнього втручання [1-3]. У суміжних ресурсно-орієнтованих системах ця проблема також розглядається як наслідок взаємозалежності процесів, ресурсів і станів керування, зокрема у моделях на основі мереж Петрі та систем із ненадійними ресурсами [4-5]. У реальних середовищах взаємоблокування формується через складну послідовність динамічних запитів, що ускладнює його виявлення під час тестування. Класичні підходи до запобігання, уникнення, виявлення та відновлення мають суттєві обмеження, оскільки вимагають жорсткого

попереднього регламентування, апріорного знання майбутніх запитів або реагують уже після виникнення конфлікту [6-8]. Для систем реального часу та вбудованих систем подібна проблема ускладнюється потребою формальної перевірки властивостей і зменшенням простору станів під час аналізу [9-10], а в задачах відмовостійкості додатково враховуються обмеженість ресурсів і необхідність раннього виявлення деградації [11-12]. Додаткову складність становить нечітка межа між штатним короткочасним очікуванням ресурсу та початком критичного блокування. Для розв'язання цієї проблеми доцільно формувати узагальнений опис динаміки процесів [13-14], а також використовувати нечітку кластеризацію як основу м'якого поділу станів за ступенями належності [15]. Відтак, сутність науково-практичної проблеми полягає у забезпеченні достовірності при ранньому виявленні взаємоблокувань шляхом переходу від реактивного виявлення до превентивного аналізу ресурсної поведінки зі збереженням можливості подальшого інтелектуального, нечіткого та статистичного аналізу результатів [15-17].

ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Класичні дослідження взаємоблокувань зосереджуються на аналізі умов їх виникнення та визначенні стратегій запобігання або виявлення. У попередніх роботах із цієї проблематики наголошується на необхідності врахування динаміки зміни станів процесу в часі, оскільки взаємоблокування формується поступово через накопичення взаємних залежностей [1; 6; 13]. Серед моделей ресурсної взаємодії активно застосовуються графові описи, моделі станів і мережі Петрі, які дають змогу формалізувати конкуренцію за ресурси та перевіряти відсутність тупикових конфігурацій [4; 5]. Водночас для складних динамічних середовищ повний формальний аналіз супроводжується значними обчислювальними витратами, тому в сучасних роботах використовуються композиційна верифікація та спеціальні прийоми редукції простору станів [9; 10].

Методи запобігання спрямовані на недопущення умов виникнення конфлікту, зокрема через попередню формалізацію допустимих станів, побудову обмежувальних правил або керування переходами у моделі ресурсного середовища [4; 5]. Такі рішення забезпечують строгість аналізу, але можуть бути занадто жорсткими для комп’ютерних систем загального призначення з динамічними запитами, оскільки потребують повної або майже повної інформації про допустимі стани системи [9; 10]. Методи уникнення взаємоблокувань аналізують наслідки виділення ресурсів, але вимагають інформації про майбутні потреби процесів, що складно реалізувати у веборієнтованих і багатозадачних середовищах [7; 2]. У свою чергу, механізми моніторингу, watchdog-таймери та процедури відновлення зберігають гнучкість, проте переважно реагують на вже проявлену затримку, відмову або симптом деградації, а не на ранню ресурсну конфігурацію, що передує взаємоблокуванню [11; 18].

Найбільш перспективним є прогнозний підхід, який оцінює ймовірність переходу процесів у небезпечний стан до фактичного формування взаємоблокування, створюючи часовий резерв для прийняття превентивних рішень [2; 3; 14]. Для його розвитку доцільно використовувати сигнатуру процесу, що об'єднує ідентифікаційні, часові, ресурсні та поведінкові ознаки [13; 7]. Це особливо актуально для операційних систем реального часу, кіберфізичних середовищ і систем з інтенсивними змінами станів, де потрібно виявляти деградацію ще до повної втрати працездатності, а не лише після спрацювання механізмів відмовостійкості або контролю затримок [12; 18].

У задачах прогнозування, коли поведінку об'єкта неможливо описати жорсткими правилами, ефективно застосовувати методи машинного навчання, інтелектуального аналізу відмов і нечіткої кластеризації [15-17]. Зокрема, нечітка кластеризація, до якої належить Fuzzy C-Means, дозволяє описувати проміжні ситуації через ступені належності об'єкта до кількох кластерів, що відповідає поступовому переходу процесу між штатним, граничним і небезпечним станами [15]. Таким чином, поєднання сигнатур процесів, нечіткої кластеризації та бази інтерпретованих правил забезпечує обґрунтований перехід від реактивного виявлення до превентивного керування ресурсами комп’ютерних систем [3; 14; 15].

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ ДОСЛІДЖЕННЯ

Архітектура системи прогнозування взаємоблокування процесів побудована як взаємодія ресурсів комп’ютерної системи, ядра операційної системи та запропонованої підсистеми аналізу. Ресурсами вважаються пам’ять, файли, пристрої введення-виведення, об’єкти синхронізації, семафори, м’ютекси, дескриптори та інші елементи, що можуть бути об’єктами конкурентного доступу. Ядро ОС передає дані про процеси через планувальник процесів і менеджер ресурсів. Планувальник фіксує створення, виконання, очікування і завершення процесів, а менеджер ресурсів відображає факти виділення, запиту, ненадання та звільнення ресурсів.

У запропонованій архітектурі (рис.1) виділено три функціональні модулі: модуль побудови сигнатур процесів, модуль навчання і модуль прогнозування. Модуль побудови сигнатур перетворює різнорідні дані ОС у формалізований опис процесу. Модуль навчання використовує накопичені сигнатури для формування бази нечітких правил. Модуль прогнозування застосовує сформовану базу правил до поточних сигнатур процесів, для яких зафіксовано ненадання запитуваного ресурсу.

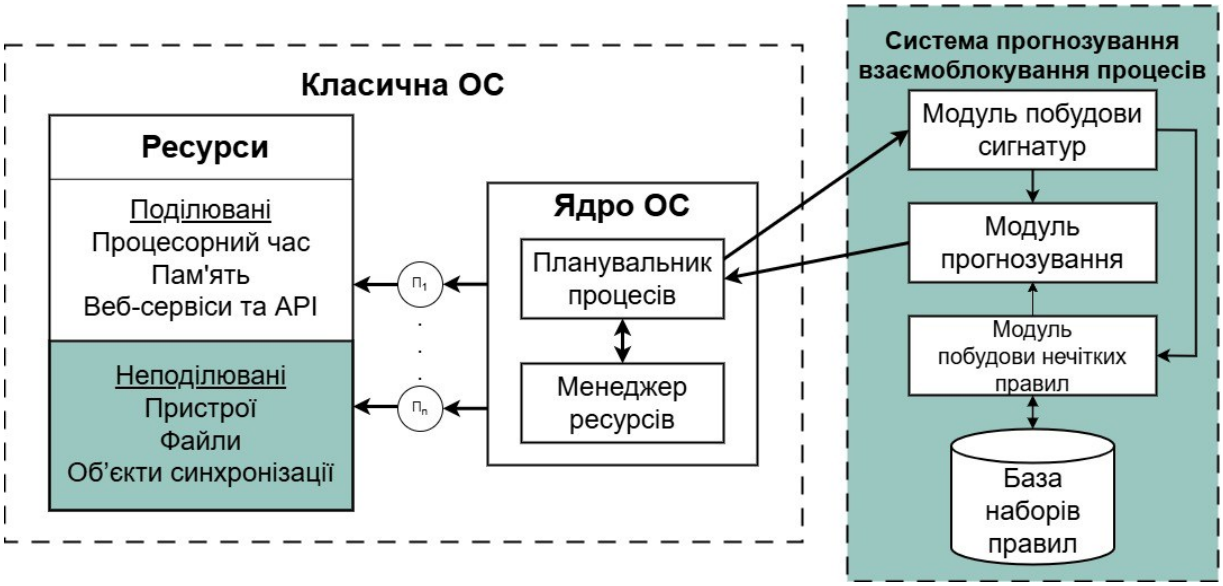

Рис. 1. Архітектура системи прогнозування стану взаємоблокування процесів у комп'ютерній системі

Подальша формалізація починається з моделі сигнатури процесу в комп'ютерній системі, яка забезпечує структурований опис його ідентифікаційних, часових, ресурсних та поведінкових характеристик. Важливість такого комплексного опису зумовлена тим, що окремо взятий параметр не дозволяє коректно оцінити ризик деструктивного стану. Зокрема, сам факт очікування ресурсу часто є штатним елементом виконання процесу, проте у поєднанні з утриманням іншого критичного ресурсу та наявністю паралельного процесу, який очікує на зворотне звільнення, він сигналізує про формування небезпечної конфігурації. Для формалізації зазначених властивостей сигнатура процесу в момент часу t подається у вигляді вектору:

$$a(t) = (a_1^t, \dots, a_k^t, a_{k+1}^t, \dots, a_{z-1}^t, a_z^t, a_{z+1}^t)$$

де z – кількість параметрів сигнатури; $a_1^t, \dots, a_k^t$ – параметри, що ідентифікують процес в системі; $a_{k+1}^t, \dots, a_{z-1}^t$ – ідентифікатори ресурсів, які процес уже займає; компонент a_z^t відображає запитуваний ресурс і визначається як:

$$a_z^t = \begin{cases} id \text{ ресурса}, & \text{якщо процес потребує додаткових ресурсів} \\ 0, & \text{якщо процес не потребує додаткових ресурсів} \end{cases}$$

причому $\forall i \in \{k+1, z\}: a_i^t \in RE$, де - множина наявних у системі ресурсів. Останній компонент a_{z+1}^t виступає індикатором поточного стану взаємоблокування:

$$a_{z+1}^t = \begin{cases} 1, \text{якщо процес знаходиться в стані взаємоблокування} \\ 0, \text{якщо процес не знаходиться в стані взаємоблокування} \end{cases}$$

Перехід від аналізу поодинокого об'єкта до оцінювання загальної ресурсної конфігурації середовища здійснюється за допомогою моделі стану комп'ютерної системи, що формується на основі множини сигнатур усіх активних процесів. Такий крок є принциповим, оскільки взаємоблокування не є ізольованою властивістю одного процесу, а виникає виключно внаслідок взаємодії кількох сутностей, які одночасно утримують одні ресурси та очікують на інші. Множину сигнатур процесів у момент часу t запропоновано подавати як:

$$A_t = \{a(t)_i\}, i = \overline{1, np}$$

де np – кількість процесів у КС. Відповідно, стан комп'ютерної системи в момент часу t визначається як інформаційний зріз взаємодії об'єктів з ресурсами і моделюється вектором $w_t = (A_t, \mathrm{t}, RE)$, де t – поточний час. Для комплексного опису функціонування обчислювального середовища протягом тривалого періоду спостереження вводиться множина станів $W = \{\mathrm{w_t} | t_s \le \mathrm{t} \le t_e\}$, де t_s – час початку спостереження; t_e – час завершення спостереження. Наявність такої часової послідовності дозволяє ретроспективно встановити, які саме ресурсні конфігурації безпосередньо передували деградації системи.

На основі сформованих станів КС задається модель процесу побудови бази нечітких правил, яка забезпечує перехід від історичних спостережень до правил нечіткого логічного висновку. Для формування

навчальної вибірки для кожної сигнатури процесу виділяється пара $(a^*(t), y^t)$,де $a^*(t)$ є підмножиною зайнятих та додатково запитуваних ідентифікаторів ресурсів процесу в момент часу t

$$a^*(t) = (a_{k+1}^t, \dots, a_{z-1}^t, a_z^t)$$

Цільовий прогнозний маркер $y(a)^t$ відображає факт переходу процесу a у стан взаємоблокування протягом наступного часового інтервалу і обчислюється як

$$y(a)^t = \begin{cases} 1, & \exists \tau \in [t, t+\Delta t]: a_{z+1}^\tau = 1, \\ 0, & \text{інакше}. \end{cases}$$

Поточна сигнатура маркується як гранична, якщо для цього самого процесу у найближчому майбутньому вікні Δt зафіксовано колапс працездатності. Тоді навчальна вибірка для одного фіксованого моменту часу системи набуває вигляду:

$$A^*(t) = \{(x, y(x)^t) \mid \exists x \in w_{t,A_t}\},$$

а загальна навчальна вибірка по всіх станах системи W за весь період спостереження репрезентується об'єднанням:

$$A^* = \bigcup_{t \in [t_h, t_e]} A^*(t)$$

Оскільки алгоритм FCM оперує виключно числовими матрицями фіксованої розмірності, вихідні сигнатури підлягають бінарному кодуванню, що приводить різнорідні характеристики до уніфікованого простору ознак без втрати смислового навантаження про зайняті та очікувані ресурси. Для формування бінарних ознак зайнятих ресурсів використовується повна множина наявних у системі ресурсів RE:

$$\forall r_j \in RE: h_j^{(s),t} = \begin{cases} 1, & r_j \in \{a_{k+1}^{(s),t}, \dots, a_{z-1}^{(s),t}\}, \\ 0, & r_j \notin \{a_{k+1}^{(s),t}, \dots, a_{z-1}^{(s),t}\}, \end{cases} \qquad j = 1, \dots, |RE|$$

де $h_j^{(s),t} = 1$ означає, що процес а у момент часу займає ресурс .

Окремо й незалежно кодується ресурс, який процес очікує в поточний момент:

$$q_j^{(s),t} = \begin{cases} 1, & a_z^{(s),t} = r_j, \\ 0, & a_z^{(s),t} \neq r_j, \end{cases} \qquad j = 1, \dots, m$$

У випадку, коли процес виконується у штатному режимі й не очікує на виділення ресурсів ($a_z^{(s),t} = 0$), всі компоненти вектора запитів прирівнюються до нуля ($q_j^{(s),t} = 0$). В результаті кожна сигнатура процесу перетворюється на фіксований вектор:

$$\tilde{a}^{(s)}(t) = \left(h_1^{(s),t}, \dots, h_m^{(s),t}, q_1^{(s),t}, \dots, q_m^{(s),t}\right)$$

З усіх перетворених сигнатур навчальної вибірки A* формується матриця даних X для FCM, кожний рядок якої відповідає одному вектору ознак процесу,

$$X = \left[a_{s_1}^{t_1}; a_{s_2}^{t_2}; \dots; a_{s_{|A^*|}}^{t_{|A^*|}}\right].$$

На етапі кластеризації фіксується кількість кластерів c = 4, що відповідає чотирьом базовим станам: нормальна робота, очікування ресурсу без критичного ризику, граничний стан та активоване взаємоблокування. Результат застосування процедури нечіткої кластеризації відображається як:

$$FCM(X, 4) \to \{v_j, u_{ij}\}, j = \overline{1,4}, i = \overline{1, |A^*|}$$

де v_j — центр j-го кластера; u_{ij} — ступінь належності і-го рядка матриці X до j-го кластера, значення u_{ij} належить інтервалу [0; 1] і показує, наскільки відповідна перетворена сигнатура близька до центра v_j.

На основі збережених цільових маркерів y_i обчислюється апостеріорний нечіткий ризик P_j для кожного сформованого кластера:

$$P_j = \frac{\sum_{i=1}^{|A^*|} u_{ij}^{mf} \cdot y_i}{\sum_{i=1}^{|A^*|} u_{ij}^{mf}}, j = \overline{1,4}$$

де mf — параметр нечіткості. Значення P_j показує, яка частка сигнатур, близьких до кластера j, у найближчому майбутньому переходила у взаємоблокування. Якщо P_j наближається до 0, кластер відповідає безпечному стану. Якщо P_j наближається до 1, кластер відповідає станам із високим ризиком взаємоблокування. Кожен кластер безпосередньо задає структуру одного нечіткого правила продукційної бази:

$$R_j: IF\ a_s^t \sim v_j\ THEN\ P_d = P_j, j = \overline{1,4}$$

де $a_s^t \sim v_j$ описує міру близькості поточної сигнатури до еталонного центру, формуючи автоматично згенеровану базу правил $B = \{R_1, R_2, R_3, R_4\}$.

Модель прогнозування взаємоблокувань орієнтована на оперативний аналіз поточних передумов виникнення небезпечних станів і активується асинхронно — у моменти виникнення подій ненадання ресурсу за запитом. Це дозволяє суттєво оптимізувати обчислювальні витрати, фокусуючи увагу лише на об'єктах із потенційною ресурсною залежністю. Після фіксації події незадоволеного запиту формується оновлений вектор ознак процесу відповідно до правил кодування навчальної вибірки $a_i^{t+\Delta t} \to a_i^{*\,t+\Delta t}$.

Цей вектор подається на вхід бази правил B, де кожне правило спрацьовує пропорційно близькості вектора до центрів кластерів. Виходом моделі є вектор ступенів належності, який відображає поступовий, розмитий характер розвитку конфлікту:

$$F_B(a_i^{*\,t+\Delta t}) = (\gamma_{i1}, \gamma_{i2}, \gamma_{i3}, \gamma_{i4})$$

де $\gamma_{il} = \mu_{R_l}(a_i^{*\,t+\Delta t}), l = 1, \ldots, 4$ визначає інтенсивність активації відповідного правила.

Для прийняття остаточного рішення щодо превентивного керування основна увага концентрується на аналізі третього (граничного) та четвертого (критичного) станів за допомогою операції нечіткого об'єднання:

$$\gamma_i^{3,4} = \gamma_{i3} \vee \gamma_{i4} = \max(\gamma_{i3}, \gamma_{i4})$$

Умова віднесення аналізованого процесу до множини граничних об'єктів , що потребують негайного втручання, визначається перевищенням заданого порогового значення θ:

$$\forall a_i^{t+\Delta t} \in A^{t+\Delta t}: \left[a_{i,z}^{t+\Delta t} \neq 0 \wedge \max\left(\mu_{R_3}(a_i^{*\,t+\Delta t}), \mu_{R_4}(a_i^{*\,t+\Delta t})\right) > \theta\right] \Rightarrow a_i^{t+\Delta t} \in D_\theta$$

Якщо розрахована оцінка не перевищує поріг, процес залишається в системі без зовнішнього втручання. У разі підтвердження небезпеки модель переходить до етапу вибору оптимальної керуючої дії. З метою мінімізації системних втрат алгоритм здійснює пошук прикладної (несистемної) сутності з мінімальним часом перебування в комп'ютерній системі за критерієм:

$$a_{i,1}^{t+\Delta t}: \left[a_i^{t+\Delta t} \in D_\theta \wedge a_{i,4}^{t+\Delta t} \neq 1 \wedge a_{i,5}^{t+\Delta t} = \min\left\{a_{k,5}^{t+\Delta t} \mid a_k^{t+\Delta t} \in D_\theta \wedge a_{k,4}^{t+\Delta t} \neq 1\right\}\right]$$

Практична реалізація наведених моделей починається з методу побудови сигнатури процесу, який перетворює системні дані ОС у впорядкований вектор ознак без ручного опису кожної можливої комбінації ресурсів.

Метод побудови сигнатури процесу призначений для формування або оновлення сигнатури $a(t)_i$ на основі поточних даних ОС. Вхідними даними є PID процесу, параметри процесу, множина ресурсів RE, поточна множина сигнатур A_t, дані про зайняті ресурси та інформація про запитуваний ресурс. Вихідними даними є актуальна сигнатура процесу і оновлена множина A_t. Метод не визначає ризик взаємоблокування, а тільки формує коректний вхід для наступних етапів.

Крок 1. На вхід подається PID процесу, для якого потрібно побудувати або оновити сигнатуру.

Крок 2. У множині A_t перевіряється наявність сигнатури з таким PID.

Крок 3. Якщо сигнатура відсутня, формується ідентифікаційна частина нового запису.

Крок 4. Якщо сигнатура вже існує, оновлюється її змінна частина без дублювання запису в A_t.

Крок 5. Визначається множина ресурсів із RE, які процес уже займає в момент спостереження.

Крок 6. Перевіряється наявність невиконаного запиту на ресурс. Якщо запиту немає, $a_z^t = 0$; якщо запит є, то a_z^t набуває значення ідентифікатора запитуваного ресурсу.

Крок 7. Сформована або оновлена сигнатура записується до A_t. Для нового процесу виконується додавання, а для відомого процесу - заміна попереднього запису актуальним.

Крок 8. Повертається поточна сигнатура a_i^t і оновлена множина A_t.

Крок 9. Метод завершується; отримана сигнатура може бути використана для навчання, кластеризації або прогнозування.

Даний метод завершує підготовчий етап: різнорідні події ОС і параметри ресурсної взаємодії вже подані у вигляді однотипних сигнатур. Оскільки самі сигнатури ще не містять правил інтерпретації ризику, наступним етапом застосовується метод побудови бази нечітких правил, який використовує Fuzzy C-Means для виділення типових груп ресурсної поведінки та перетворення результатів кластеризації у правила висновку.

Метод побудови бази нечітких правил призначений для автоматизованого формування правил підсистеми нечіткого логічного висновку на основі множини станів W. Вхідними даними є W, сигнатури процесів, множина ресурсів RE, інтервал Δt і параметри FCM-кластеризації. Вихідними даними є база правил B.

Крок 1. На вхід методу подається множина станів W за період від t_s до t_e.

Крок 2. Із W формується модифікована множина навчальних сигнатур із прив'язкою до PID, часу t і вихідного стану КС.

Крок 3. Перевіряється наявність неопрацьованих сигнатур. Якщо всі сигнатури опрацьовані, виконується перехід до формування матриці X.

Крок 4. Для поточної сигнатури формується навчальна пара.

Крок 5. Визначається прогнозний маркер $y(a)^t$.

Крок 6. Виділяється ресурсна частина сигнатури.

Крок 7. Зайняті ресурси кодуються.

Крок 8. Формується фіксований ресурсний вектор.

Крок 9. Підготовлений ресурсний вектор і відповідний маркер додаються до навчальної вибірки A*; після цього метод повертається до перевірки неопрацьованих сигнатур.

Крок 10. Після опрацювання всіх сигнатур формується матриця X.

Крок 11. Виконується FCM-кластеризація матриці X.

Крок 12. Для кожного кластера обчислюється ризик P_j, після чого генеруються нечіткі правила R_j і формується база B.

Сутність методу полягає в тому, що Fuzzy C-Means не замінює прогнозування, а використовується для побудови інтерпретованих ресурсних станів. Маркер $y(a)^t$ не нав'язує кластери, а застосовується після кластеризації для визначення ризику кожного кластера. Це дозволяє отримати базу правил, у якій передумови формуються на основі структури даних, а висновки - на основі фактичних переходів до взаємоблокування.

Після другого методу отримується база B, яка описує типові ресурсні стани, але ще не визначає рішення для конкретного поточного PID. Тому завершальним етапом є метод прогнозування стану процесів у КС. Він застосовує сформовану базу правил до поточного стану системи та визначає множину процесів з підвищеним ризиком взаємоблокування.

Метод прогнозування стану процесів у КС використовує поточні сигнатури, базу нечітких правил B і поріг θ для виявлення процесів, що мають належність до граничного або взаємозаблокованого стану. Вхідними даними є A_t, RE, B, системна подія про зміну стану процесу або ресурсної взаємодії, а також поріг θ. Вихідними даними методу є результат оцінювання поточного стану процесів: відсутність ризикових

процесів або сформована множина D_θ процесів, що мають належність до небезпечних кластерних станів.

Крок 1. Перевіряється наявність бази нечітких правил B.

Крок 2. Якщо база B відсутня, виконується метод побудови бази нечітких правил; після формування B прогнозування продовжується.

Крок 3. Зчитується база B і перевіряється відповідність її структури чотирьом кластерним станам.

Крок 4. Виявляється зміна в і-му процесі або його ресурсній взаємодії.

Крок 5. Методом побудови сигнатури процесу оновлюється $a(t)_i$ і множина A_t.

Крок 6. Перевіряється, чи запитав процес ресурс і чи не отримав його. Якщо запиту немає або він задоволений, прогнозне оцінювання не виконується.

Крок 7. Якщо ресурс не надано, сигнатура перетворюється у вектор і подається на базу правил B.

Крок 8. Обчислюється вектор спрацювання правил.

Крок 9. Обчислюється належність до небезпечної області.

Крок 10. Якщо $\gamma_i^{3,4} > \theta$ і $a_{i,z}^{t+\Delta t} \neq 0$, сигнатура включається до D_θ.

Крок 11. Якщо D_θ не порожня, то визначається PID процесу для керуючої дії за критерієм несистемності та найменшого часу перебування в системі.

Крок 12. Виконується керуюча дія або формується повідомлення про неможливість автоматичного втручання; після цього оновлюються A_t і RE.

Зв'язок трьох методів є послідовним: метод побудови сигнатури формує вхідні дані, метод побудови бази нечітких правил навчає систему, а метод прогнозування застосовує сформовані правила в оперативному режимі. FCM-кластеризація не запускається для кожної нової події ОС; нова сигнатура тільки приводиться до узгодженого формату, порівнюється з центрами кластерів через правила B і отримує ступені належності до чотирьох ресурсних станів. Це зменшує обчислювальні витрати й дозволяє застосовувати метод у режимі поточного моніторингу.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

У статті запропоновано метод прогнозування стану взаємоблокування процесів у комп'ютерних системах, що ґрунтується на формуванні сигнатур процесів, моделі поточного стану системи, автоматизованій побудові бази нечітких правил та оцінюванні ризику переходу процесів у небезпечний ресурсний стан. Показано, що використання сигнатури процесу дає змогу враховувати не лише факт очікування ресурсу, а й сукупність ідентифікаційних, часових, ресурсних і поведінкових характеристик, які визначають контекст взаємодії процесів у системі. Застосування Fuzzy C-Means для побудови бази нечітких правил забезпечує виділення типових ресурсних станів і врахування поступового переходу від нормального виконання до граничного або взаємозаблокованого стану. Запропонований метод прогнозування активується у моменти ненадання ресурсу за запитом процесу, що дозволяє зменшити обчислювальні витрати та зосередити аналіз на потенційно небезпечних ситуаціях. Отримані результати створюють основу для превентивного керування ресурсами комп'ютерної системи шляхом раннього виявлення передумов взаємоблокування та вибору доцільної керуючої дії до моменту повної втрати працездатності процесів.

Література

1. Савенко О.С., Кльоц Ю.П., Мостовий С.В. Дослідження та аналіз блокування процесів в комп'ютерній системі. Вісник ХНУ. 2007. №3. С. 248–251.

2. Локазюк В.М., Савенко О.С., Мостовий С.В. Модель прогнозування стану взаємоблокування процесів комп'ютерної системи. Вісник ВПІ. 2011. №4. С. 130–133.

3. Савенко О.С., Мостовий С.В. Прогнозування потрапляння процесів у стан взаємоблокування. Вісник ВПІ. 2013. №2. С. 81–86.

4. Čapkovič F. Sustainability of Automated Manufacturing Systems with Resources by Means of Their Deadlock Prevention. Electronics. 2024. Vol. 13, No. 17. Article 3517. DOI: 10.3390/electronics13173517.

5. Luo J.C., Liu Z.Q., Zhou M.C., Xing K.Y., Wang X.N., Li X.L., Liu H.X. Robust deadlock control of automated manufacturing systems with multiple unreliable resources. Information Sciences. 2019. Vol. 479. P. 401–415. DOI: 10.1016/j.ins.2018.11.051.

6. Савенко О.С., Мостовий С.В. Метод прогнозування стану процесів в персональних комп'ютерах. Вісник ВПІ. 2008. №6. С. 120–125.

7. Мостовий С.В., Ковтун О.Л., Прокопишен І.В. Метод та засоби прогнозування стану взаємоблокування процесів в персональному комп'ютері. Вісник ХНУ. 2010. №1. С. 123–127.

8. Кльоц Ю.П., Мостовий С.В. Оцінка часової складності методу прогнозування взаємоблокувань процесів в комп'ютерних системах. Вісник ХНУ. 2014. №1. С. 87–93.

9. Foughali M., Hladik P.E., Zuepke A. Compositional verification of embedded real-time systems. Journal of Systems Architecture. 2023. Vol. 142. Article 102928. DOI: 10.1016/j.sysarc.2023.102928.

10. Yamane S., Kiriyama T., Wu Y. An Efficient Reduction of Timer Interrupts for Model Checking of Embedded Assembly Programs. Electronics. 2024. Vol. 13, No. 2. Article 463. DOI: 10.3390/electronics13020463.

11. Solouki M.A., Angizi S., Violante M. Dependability in embedded systems: a survey of fault-tolerance methods and software-based mitigation techniques. IEEE Access. 2024. Vol. 12. P. 180939–180965. DOI: 10.1109/ACCESS.2024.3509633.

12. Saraeian S., Shirazi B. Digital twin-based fault tolerance approach for Cyber-Physical Production System. ISA Transactions. 2022. Vol. 130. P. 35–50. DOI: 10.1016/j.isatra.2022.03.007.

13. Савенко О.С., Мостовий С.В. Життєвий цикл процесів комп'ютерної системи. Радіоелектронні і комп'ютерні системи. 2010. №7(48). С. 35–38.

14. Nicheporuk A., Klots Y., Yashyna O., Mostovyi S., Nicheporuk Y. Prediction of entering processes into the deadlock state. Indonesian Journal of Electrical Engineering and Computer Science. 2019. Vol. 14, No. 3. P. 1484–1492. DOI: 10.11591/ijeecs.v14.i3.pp1484-1492.

15. Shenify M., Mazarbhuiya F.A., Wungreiphi A.S. Detecting IoT Anomalies Using Fuzzy Subspace Clustering Algorithms. Applied Sciences. 2024. Vol. 14, No. 3. Article 1264. DOI: 10.3390/app14031264.

16. Mercorelli P. Recent advances in intelligent algorithms for fault detection and diagnosis. Sensors. 2024. Vol. 24, No. 8. Article 2656. DOI: 10.3390/s24082656.

17. Rahman M.M., Gupta D., Bhatt S., Shokouhmand S., Faezipour M. A Comprehensive Review of Machine Learning Approaches for Anomaly Detection in Smart Homes: Experimental Analysis and Future Directions. Future Internet. 2024. Vol. 16, No. 4. Article 139. DOI: 10.3390/fi16040139.

18. Asch B., Jellum E., Lohstroh M., Lee E.A. Software-Defined Watchdog Timers for Cyber-Physical Systems. IEEE Embedded Systems Letters. 2024. Vol. 18, No. 9. P. 115–118. DOI: 10.1109/LES.2024.3467332.

References

1. Savenko O.S., Klots Yu.P., Mostovyi S.V. Research and analysis of process blocking in a computer system. Herald of Khmelnytskyi National University. 2007. No. 3. P. 248–251. (in Ukrainian).

2. Lokaziuk V.M., Savenko O.S., Mostovyi S.V. Model for predicting the deadlock state of computer system processes. Bulletin of Vinnytsia Polytechnic Institute. 2011. No. 4. P. 130–133. (in Ukrainian).

3. Savenko O.S., Mostovyi S.V. Prediction of processes entering the deadlock state. Bulletin of Vinnytsia Polytechnic Institute. 2013. No. 2. P. 81–86. (in Ukrainian).

4. Čapkovič F. Sustainability of Automated Manufacturing Systems with Resources by Means of Their Deadlock Prevention. Electronics. 2024. Vol. 13, No. 17. Article 3517. DOI: 10.3390/electronics13173517.

5. Luo J.C., Liu Z.Q., Zhou M.C., Xing K.Y., Wang X.N., Li X.L., Liu H.X. Robust deadlock control of automated manufacturing systems with multiple unreliable resources. Information Sciences. 2019. Vol. 479. P. 401–415. DOI: 10.1016/j.ins.2018.11.051.

6. Savenko O.S., Mostovyi S.V. Method for predicting the state of processes in personal computers. Bulletin of Vinnytsia Polytechnic Institute. 2008. No. 6. P. 120–125. (in Ukrainian).

7. Mostovyi S.V., Kovtun O.L., Prokopyshen I.V. Method and tools for predicting the deadlock state of processes in a personal computer. Herald of Khmelnytskyi National University. 2010. No. 1. P. 123–127. (in Ukrainian).

8. Klots Yu.P., Mostovyi S.V. Estimation of the time complexity of the method for predicting process deadlocks in computer systems. Herald of Khmelnytskyi National University. 2014. No. 1. P. 87–93. (in Ukrainian).

9. Foughali M., Hladik P.E., Zuepke A. Compositional verification of embedded real-time systems. Journal of Systems Architecture. 2023. Vol. 142. Article 102928. DOI: 10.1016/j.sysarc.2023.102928.

10. Yamane S., Kiriyama T., Wu Y. An Efficient Reduction of Timer Interrupts for Model Checking of Embedded Assembly Programs. Electronics. 2024. Vol. 13, No. 2. Article 463. DOI: 10.3390/electronics13020463.

11. Solouki M.A., Angizi S., Violante M. Dependability in embedded systems: a survey of fault-tolerance methods and software-based mitigation techniques. IEEE Access. 2024. Vol. 12. P. 180939–180965. DOI: 10.1109/ACCESS.2024.3509633.

12. Saraeian S., Shirazi B. Digital twin-based fault tolerance approach for Cyber-Physical Production System. ISA Transactions. 2022. Vol. 130. P. 35–50. DOI: 10.1016/j.isatra.2022.03.007.

13. Savenko O.S., Mostovyi S.V. Life cycle of computer system processes. Radioelectronic and Computer Systems. 2010. No. 7(48). P. 35–38. (in Ukrainian).

14. Nicheporuk A., Klots Y., Yashyna O., Mostovyi S., Nicheporuk Y. Prediction of entering processes into the deadlock state. Indonesian Journal of Electrical Engineering and Computer Science. 2019. Vol. 14, No. 3. P. 1484–1492. DOI:

10.11591/ijeecs.v14.i3.pp1484-1492.

15. Shenify M., Mazarbhuiya F.A., Wungreiphi A.S. Detecting IoT Anomalies Using Fuzzy Subspace Clustering Algorithms. Applied Sciences. 2024. Vol. 14, No. 3. Article 1264. DOI: 10.3390/app14031264.

16. Mercorelli P. Recent advances in intelligent algorithms for fault detection and diagnosis. Sensors. 2024. Vol. 24, No. 8. Article 2656. DOI: 10.3390/s24082656.

17. Rahman M.M., Gupta D., Bhatt S., Shokouhmand S., Faezipour M. A Comprehensive Review of Machine Learning Approaches for Anomaly Detection in Smart Homes: Experimental Analysis and Future Directions. Future Internet. 2024. Vol. 16, No. 4. Article 139. DOI: 10.3390/fi16040139.

18. Asch B., Jellum E., Lohstroh M., Lee E.A. Software-Defined Watchdog Timers for Cyber-Physical Systems. IEEE Embedded Systems Letters. 2024. Vol. 18, No. 9. P. 115–118. DOI: 10.1109/LES.2024.3467332.