ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова праця
на правах рукопису

МОСТОВИЙ СЕРГІЙ ВОЛОДИМИРОВИЧ

УДК 004.45

ДИСЕРТАЦІЯ

МЕТОД ТА ЗАСОБИ ПРОГНОЗУВАННЯ СТАНУ ВЗАЄМОБЛОКУВАННЯ ПРОЦЕСІВ В КОМП'ЮТЕРНИХ СИСТЕМАХ

05.13.05 – Комп'ютерні системи та компоненти
05 – Технічні науки

Подається на здобуття наукового ступеня кандидата технічних наук.
Дисертація містить результати власних досліджень. Подані до захисту наукові положення є власним напрацюванням. Всі використані ідеї, наукові результати, цитати супроводжуються належними посиланнями на їх авторів та джерела опублікування. Всі частини тексту дисертації, під час написання яких використовувались технології штучного інтелекту, перевірені та відредаговані особисто.

_____С.В. Мостовий
підпис

Науковий керівник:
Савенко Олег Станіславович, доктор технічних наук, професор

Хмельницький – 2026

Ідентичність примірників дисертації засвідчую

Вчений секретар СВР Андрій НІЧЕПОРУК

Мостовий С.В. Метод та засоби прогнозування стану взаємоблокування процесів в комп'ютерних системах. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук з галузі знань 05 Технічні науки за спеціальністю 05.13.05 Комп'ютерні системи та компоненти. – Хмельницький національний університет, Хмельницький, 2026.

У роботі подано результати дослідження, спрямованого на підвищення достовірності прогнозування стану взаємоблокування процесів у комп’ютерних системах. Запропоновано підхід до завчасного оцінювання ризику виникнення взаємоблокування на основі формалізованого подання ресурсної поведінки процесів у вигляді сигнатур, автоматизованого формування бази нечітких правил із використанням нечіткої кластеризації Fuzzy C-Means та нечіткого логічного висновку. Це дає змогу враховувати поточний стан процесів, утримувані й запитувані ресурси, часові та поведінкові характеристики, виявляти тенденцію до формування небезпечної ресурсної ситуації ще до фактичного виникнення взаємоблокування і створювати підстави для прийняття превентивних рішень щодо керування ресурсами комп’ютерної системи.

У дисертації проведено аналіз моделей, методів і алгоритмів вирішення задачі взаємоблокування процесів, а також можливостей застосування машинного навчання та інтелектуального аналізу для прогнозування небезпечних станів. Розроблено архітектуру системи прогнозування, моделі сигнатури процесу, стану комп’ютерної системи, формування бази нечітких правил і прогнозування стану взаємоблокування. Розроблено методи побудови сигнатури процесу, автоматизованого формування бази нечітких правил та прогнозування ризику переходу процесів у стан взаємоблокування, а також удосконалено архітектуру відповідних програмних засобів. Для практичної реалізації розроблено програмний засіб, що забезпечує отримання даних від операційної системи, опрацювання сигнатур процесів, нечітке оцінювання ризику та формування превентивних керуючих рішень.

Об'єктом дослідження є процеси функціонування комп'ютерних систем в умовах конкурентного використання системних ресурсів процесами, що можуть призводити до виникнення стану взаємоблокування.

Предметом дослідження є моделі, методи та засоби побудови сигнатур процесів, формування бази нечітких правил і прогнозування стану взаємоблокування процесів у комп'ютерних системах на основі аналізу ресурсної взаємодії процесів, нечіткої кластеризації та нечіткого логічного висновку.

Метою дисертаційного дослідження є підвищення достовірності та забезпечення своєчасності прогнозування стану взаємоблокування процесів у комп'ютерних системах шляхом розроблення методу та засобів аналізу сигнатур процесів, автоматизованого формування бази нечітких правил і завчасного оцінювання ризику переходу процесів у стан взаємоблокування.

Наукова новизна отриманих результатів полягає в наступному:

1) вперше розроблено метод прогнозування стану взаємоблокування процесів у комп'ютерних системах, який, на відміну від відомих методів виявлення взаємоблокувань після їх виникнення або запобігання їм на основі наперед заданих обмежень розподілу ресурсів, забезпечує завчасне оцінювання ризику переходу процесів у стан взаємоблокування шляхом обробки сигнатур процесів базою нечітких правил, сформованою за результатами кластеризації, що дає змогу враховувати поточний ресурсний стан системи, особливості взаємодії процесів із ресурсами та формувати підстави для прийняття превентивних рішень щодо керування ресурсами комп'ютерної системи;

2) удосконалено метод побудови сигнатури процесу в комп'ютерних системах, який, на відміну від підходів, що описують процес через окремі системні параметри або поточний стан виконання, передбачає формалізоване перетворення ідентифікаційних, ресурсних, часових та поведінкових характеристик процесу у впорядкований вектор ознак із урахуванням зайнятих ресурсів, запитуваного ресурсу та особливостей ресурсної взаємодії, що дає змогу сформувати інформативний

опис процесу для подальшої кластеризації, нечіткого аналізу та прогнозування стану взаємоблокування;

3) удосконалено метод побудови бази нечітких правил для прогнозування стану взаємоблокування процесів у комп'ютерних системах, який, на відміну від відомих підходів формування правил на основі повного експертного налаштування, використовує нечітку кластеризацію Fuzzy C-Means для автоматизованого виділення типових сценаріїв ресурсної поведінки процесів з подальшим формуванням правил нечіткого логічного висновку, що дає змогу зменшити залежність від експертних оцінок, врахувати приховані закономірності у вибірці станів процесів та підвищити адаптивність бази правил до зміни поведінки процесів у комп'ютерній системі;

4) удосконалено архітектуру засобів прогнозування стану взаємоблокування процесів у комп'ютерних системах, яка, на відміну від відомих підходів, орієнтованих переважно на виявлення або усунення взаємоблокування після його виникнення, передбачає поєднання модулів збору даних про процеси та ресурси, побудови сигнатур процесів, формування бази нечітких правил і прогнозування стану взаємоблокування, що забезпечує завчасне оцінювання ризику переходу процесів у стан взаємоблокування та створює основу для прийняття превентивних рішень щодо керування ресурсами комп'ютерної системи.

Практичне значення отриманих результатів. У результаті виконаного дисертаційного дослідження розроблено програмні засоби прогнозування стану взаємоблокування процесів у комп'ютерних системах, які забезпечують отримання системних подій, формування та збереження сигнатур процесів, використання автоматизовано сформованої бази нечітких правил, оцінювання ризику переходу процесів у стан взаємоблокування та формування підстав для превентивного керування ресурсами. Розділення процедур навчання та оперативного прогнозування дає змогу виконувати ресурсомісткі процедури кластеризації та формування бази правил поза контуром оперативного розподілу ресурсів, а під час функціонування системи використовувати сформовану базу правил для оцінювання поточних станів процесів.

Результати прогнозування можуть бути використані в операційних системах і засобах керування ресурсами для посиленого моніторингу, зміни черговості доступу до ресурсів, відкладення ресурсного запиту та формування інших превентивних керуючих впливів.

Результати експериментальних досліджень підтвердили практичну придатність розроблених методів і засобів для достовірного та своєчасного прогнозування стану взаємоблокування процесів. За результатами імітаційного моделювання для різної кількості ресурсів і частки процесів, що потрапляють у стан взаємоблокування, забезпечено завершення 99,85–99,95 % процесів. Значення Accuracy становило 99,65–99,88 %, Recall – 99,00 %, Precision – 98,61–98,67 %, F1-score – 98,80–98,84 %, при цьому частка пропущених небезпечних процесів не перевищувала 1,00 %. Експериментальна перевірка на реальній комп’ютерній системі підтвердила стабільність результатів: Accuracy становила 99,65–99,82 %, Precision – 98,39–98,49 %, F1-score – 98,19–98,25 %, а прогноз формувався в середньому за 3,1–3,6 мс до фактичного виникнення взаємоблокування. Порівняно з методом на основі графа розподілу ресурсів середній час обробки скоротився на 8,6–8,8 %, а пропускна здатність зросла на 9,4 %.

Теоретичні та практичні результати дослідження впроваджені в ПП "Авіві" (довідка від 29.06.2026р. м.Хмельницький), ТОВ "ІТТ" (акт від 25.06.2026р. м.Хмельницький), у освітній процес Хмельницького національного університету (акт від 26.06.2026р.) при викладанні дисциплін на кафедрі кібербезпеки для здобувачів спеціальності F5 Кібербезпека, зокрема в курсах "Основи інформаційної безпеки", "Операційні системи та технології їх захисту" та при викладанні дисциплін на кафедрі комп’ютерної інженерії та інформаційних систем для здобувачів спеціальності F7 Комп’ютерна інженерія, зокрема в курсах "Системне програмне забезпечення", "Технологія проєктування спеціалізованих операційних систем".

У вступі обґрунтовано актуальність науково-прикладної задачі підвищення достовірності прогнозування стану взаємоблокування процесів у комп’ютерних си-

стемах. Сформульовано мету та завдання дослідження, визначено його об’єкт, предмет і методи. Показано зв’язок роботи з науковими програмами, планами та темами, наведено наукову новизну і практичне значення отриманих результатів, особистий внесок здобувача, відомості про апробацію, публікації та впровадження результатів дисертаційної роботи.

У першому розділі проаналізовано причини й умови виникнення взаємоблокувань, моделі подання станів процесів і ресурсних залежностей, а також відомі методи запобігання, уникнення, виявлення та усунення взаємоблокувань. Встановлено обмежений або реактивний характер традиційних підходів і обґрунтовано доцільність прогнозування на основі сигнатур процесів, нечіткої кластеризації та нечіткого логічного висновку.

У другому розділі розроблено комплекс взаємопов’язаних моделей прогнозування стану взаємоблокування процесів. Запропоновано архітектуру системи з відокремленими контурами навчання та оперативного прогнозування, модель сигнатури процесу, що поєднує ідентифікаційні, поведінкові та ресурсні характеристики, і модель стану комп’ютерної системи як часово впорядкованої множини сигнатур активних процесів. Обґрунтовано бінарне кодування ресурсних ознак, застосування евклідової відстані та методу Fuzzy C-Means для виділення нормальних, граничних і небезпечних станів. На основі результатів кластеризації сформовано моделі побудови бази нечітких правил та оцінювання ризику переходу процесів у стан взаємоблокування.

У третьому розділі розроблено метод побудови сигнатури процесу, який перетворює різнорідні дані операційної системи на впорядкований вектор ознак сталої розмірності з урахуванням утримуваних і запитуваних ресурсів. Удосконалено метод побудови бази нечітких правил, що передбачає формування навчальної вибірки з накопичених сигнатур, нечітку кластеризацію Fuzzy C-Means, визначення ризику для сформованих кластерів та автоматизоване генерування правил типу "якщо–то". Розроблено метод прогнозування, у межах якого поточна сигнатура

процесу фазифікується, зіставляється з базою правил, після чого шляхом агрегування та дефазифікації визначається інтегральний показник ризику. Аналіз виконується для процесів, що очікують ресурс, а перевищення заданого порогу є підставою для формування множини процесів-кандидатів і застосування превентивних керуючих дій.

У четвертому розділі розроблено архітектуру програмного засобу прогнозування стану взаємоблокування процесів, яка забезпечує повний цикл опрацювання, тобто від отримання подій ядра операційної системи та побудови сигнатур процесів до нечіткого оцінювання ризику й формування керуючого рішення. Деталізовано алгоритми роботи модулів спостереження, побудови сигнатур, зберігання навчальних і поточних даних, формування бази нечітких правил, прогнозування та превентивних дій. Для перевірки результативності методу визначено підхід до оцінювання достовірності прогнозування за показниками Accuracy, Precision, Recall, Specificity та F1-score, а також до аналізу часової складності й порівняння з відомими підходами.

У висновках узагальнено наукові та практичні результати дослідження, що забезпечують перехід від реактивного виявлення взаємоблокувань до завчасного оцінювання ризику їх виникнення на основі сигнатур процесів, автоматично сформованої бази нечітких правил і нечіткого логічного висновку.

У додатках наведено матеріали, що підтверджують апробацію та практичне використання результатів дисертаційної роботи, наукові публікації здобувача, акти впровадження, допоміжні матеріали й лістинги розробленого програмного забезпечення.

Ключові слова: комп’ютерні системи; процес; системні ресурси; взаємоблокування процесів; прогнозування; сигнатура процесу; ресурсна взаємодія; розподіл ресурсів; адаптивні алгоритми; машинне навчання; нечітка кластеризація; Fuzzy C-Means; база нечітких правил; нечіткий логічний висновок; оцінювання ризику; алгоритми обробки даних; операційна система.

ANNOTATION

Mostovyi S.V. Method and Tools for Predicting the Process Deadlock State in Computer Systems. – Qualifying scientific work submitted as a manuscript.

Dissertation for the degree of Candidate of Technical Sciences in the field of knowledge 05 Technical Sciences, specialty 05.13.05 Computer Systems and Components. – Khmelnytskyi National University, Khmelnytskyi, 2026.

The dissertation presents the results of a study aimed at improving the reliability of predicting the deadlock state of processes in computer systems. An approach is proposed for the early assessment of the risk of deadlock occurrence based on a formalized representation of the resource behavior of processes in the form of signatures, the automated formation of a fuzzy rule base using Fuzzy C-Means clustering, and fuzzy logical inference. This makes it possible to take into account the current state of processes, the resources held and requested, as well as temporal and behavioral characteristics, to identify a tendency toward the formation of a dangerous resource situation before the actual occurrence of a deadlock, and to create a basis for making preventive decisions on resource management in a computer system.

The dissertation analyzes models, methods, and algorithms for solving the problem of process deadlocks, as well as the possibilities of applying machine learning and intelligent data analysis to the prediction of dangerous states. An architecture of the prediction system has been developed, together with models of the process signature, the state of the computer system, the formation of the fuzzy rule base, and the prediction of the deadlock state. Methods have been developed for constructing a process signature, for the automated formation of a fuzzy rule base, and for predicting the risk of transition of processes into a deadlock state; the architecture of the corresponding software tools has also been improved. For practical implementation, a software tool has been developed that ensures the acquisition of data from the operating system, the processing of process signatures, fuzzy risk assessment, and the formation of preventive control decisions.

The object of the research is the processes of functioning of computer systems under conditions of competitive use of system resources by processes, which may lead to the occurrence of a deadlock state.

The subject of the research comprises models, methods, and tools for constructing process signatures, forming a fuzzy rule base, and predicting the deadlock state of processes in computer systems based on the analysis of resource interaction between processes, fuzzy clustering, and fuzzy logical inference.

The aim of the dissertation research is to improve the reliability of predicting the deadlock state of processes in computer systems by developing a method and tools for analyzing process signatures, automatically forming a fuzzy rule base, and performing early assessment of the risk of transition of processes into a deadlock state.

The scientific novelty of the obtained results is as follows:

1) for the first time, a method for predicting the deadlock state of processes in computer systems has been developed, which, unlike known methods for detecting deadlocks after their occurrence or preventing them on the basis of predefined restrictions on resource allocation, provides early assessment of the risk of transition of processes into a deadlock state by processing process signatures using a fuzzy rule base formed as a result of clustering. This makes it possible to take into account the current resource state of the system, the specific features of process interaction with resources, and to form a basis for making preventive decisions on resource management in a computer system;

2) the method for constructing a process signature in computer systems has been improved. Unlike approaches that describe a process through separate system parameters or the current execution state, the proposed method provides for the formalized transformation of the identification, resource, temporal, and behavioral characteristics of a process into an ordered feature vector, taking into account the occupied resources, the requested resource, and the specific features of resource interaction. This enables the formation of an informative description of the process for subsequent clustering, fuzzy analysis, and prediction of the deadlock state;

3) the method for constructing a fuzzy rule base for predicting the deadlock state of processes in computer systems has been improved. Unlike known approaches to rule formation based on complete expert configuration, the proposed method uses Fuzzy C-Means clustering for the automated identification of typical scenarios of resource behavior of processes, followed by the formation of fuzzy inference rules. This reduces dependence on expert assessments, allows hidden regularities in the sample of process states to be taken into account, and increases the adaptability of the rule base to changes in process behavior in a computer system;

4) the architecture of tools for predicting the deadlock state of processes in computer systems has been improved. Unlike known approaches focused mainly on detecting or eliminating a deadlock after its occurrence, the proposed architecture provides for the integration of modules for collecting data on processes and resources, constructing process signatures, forming a fuzzy rule base, and predicting the deadlock state. This ensures early assessment of the risk of transition of processes into a deadlock state and creates a basis for making preventive decisions on resource management in a computer system.

Practical significance of the obtained results. As a result of the dissertation research, software tools for predicting the deadlock state of processes in computer systems have been developed. These tools ensure the acquisition of system events, the formation and storage of process signatures, the use of an automatically generated fuzzy rule base, the assessment of the risk of transition of processes into a deadlock state, and the formation of grounds for preventive resource management. The separation of training procedures and operational prediction makes it possible to perform resource-intensive clustering procedures and rule-base formation outside the operational resource allocation loop, while during system operation the formed rule base is used to assess the current states of processes. The prediction results can be used in operating systems and resource management tools for enhanced monitoring, changing the order of access to resources, deferring resource requests, and forming other preventive control actions.

The results of experimental studies confirmed the practical applicability of the developed methods and tools for reliable and timely prediction of the deadlock state of processes. According to the results of simulation modeling for different numbers of resources and different proportions of processes entering a deadlock state, the completion of 99.85–99.95% of processes was ensured. The Accuracy value was 99.65–99.88%, Recall was 99.00%, Precision was 98.61–98.67%, and F1-score was 98.80–98.84%, while the proportion of missed dangerous processes did not exceed 1.00%. Experimental verification on a real computer system confirmed the stability of the results: Accuracy was 99.65–99.82%, Precision was 98.39–98.49%, F1-score was 98.19–98.25%, and the prediction was formed on average 3.1–3.6 ms before the actual occurrence of a deadlock. Compared with the method based on the resource allocation graph, the average processing time was reduced by 8.6–8.8%, while throughput increased by 9.4%.

The theoretical and practical results of the study have been implemented at PE “Avivi” (certificate dated 29 June 2026, Khmelnytskyi), LLC “ITT” (implementation act dated 25 June 2026, Khmelnytskyi), and in the educational process of Khmelnytskyi National University (implementation act dated 26 June 2026) in teaching disciplines at the Department of Cybersecurity for students majoring in F5 Cybersecurity, in particular in the courses “Fundamentals of Information Security” and “Operating Systems and Technologies for Their Protection”, as well as in teaching disciplines at the Department of Computer Engineering and Information Systems for students majoring in F7 Computer Engineering, in particular in the courses “System Software” and “Technology for Designing Specialized Operating Systems”.

The introduction substantiates the relevance of the scientific and applied problem of improving the reliability of predicting the deadlock state of processes in computer systems. The aim and objectives of the research are formulated, and its object, subject, and methods are defined. The connection of the work with scientific programs, plans, and topics is shown; the scientific novelty and practical significance of the obtained results, the personal contribution of the applicant, and information on approbation, publications, and implementation of the dissertation results are presented.

The first chapter analyzes the causes and conditions of deadlock occurrence, models for representing process states and resource dependencies, as well as known methods of deadlock prevention, avoidance, detection, and recovery. The limited or reactive nature of traditional approaches is established, and the feasibility of prediction based on process signatures, fuzzy clustering, and fuzzy logical inference is substantiated.

The second chapter develops a set of interrelated models for predicting the deadlock state of processes. An architecture of the system with separate training and operational prediction loops is proposed, along with a process signature model that combines identification, behavioral, and resource characteristics, and a model of the computer system state as a time-ordered set of signatures of active processes. The binary coding of resource features, the use of Euclidean distance, and the Fuzzy C-Means method for identifying normal, boundary, and dangerous states are substantiated. Based on the clustering results, models for constructing the fuzzy rule base and assessing the risk of transition of processes into a deadlock state are formed.

The third chapter develops a method for constructing a process signature, which transforms heterogeneous operating system data into an ordered feature vector of fixed dimensionality, taking into account the resources held and requested. The method for constructing a fuzzy rule base is improved; it provides for the formation of a training sample from accumulated signatures, Fuzzy C-Means clustering, risk determination for the formed clusters, and the automated generation of “if–then” rules. A prediction method is developed, within which the current process signature is fuzzified and compared with the rule base, after which an integral risk indicator is determined through aggregation and defuzzification. The analysis is performed for processes waiting for a resource, and exceeding a specified threshold serves as a basis for forming a set of candidate processes and applying preventive control actions.

The fourth chapter develops the architecture of a software tool for predicting the deadlock state of processes, which ensures the full processing cycle, from obtaining operating system kernel events and constructing process signatures to fuzzy risk assessment and the formation of a control decision. The algorithms of the observation

module, signature construction module, storage of training and current data, fuzzy rule-base formation, prediction, and preventive action modules are detailed. To verify the effectiveness of the method, an approach is defined for assessing prediction reliability using Accuracy, Precision, Recall, Specificity, and F1-score, as well as for analyzing time complexity and comparing the proposed method with known approaches.

The conclusions summarize the scientific and practical results of the study, which ensure the transition from reactive detection of deadlocks to early assessment of the risk of their occurrence based on process signatures, an automatically generated fuzzy rule base, and fuzzy logical inference.

The appendices provide materials confirming the approbation and practical use of the dissertation results, the applicant's scientific publications, implementation acts, auxiliary materials, and listings of the developed software.

Keywords: computer systems; process; system resources; process deadlock; prediction; process signature; resource interaction; resource allocation; adaptive algorithms; machine learning; fuzzy clustering; Fuzzy C-Means; fuzzy rule base; fuzzy logical inference; risk assessment; data processing algorithms; operating system.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці, в яких опубліковані основні наукові результати дисертації

1. Ю.П. Кльоц, С.В. Мостовий. Прогнозування взаємоблокувань процесів в комп'ютерних системах. *Електротехнічні та комп'ютерні системи.* – 2015, № 19 (95) – С.205-209. URL: https://eltecs.op.edu.ua/index.php/journal/article/view/1897/946

2. 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, 14 (3), pp. 1484–1492. URL: http://doi.org/10.11591/ijeecs.v14.i3.pp1484-1492 (індексована в наукометричній базі Scopus)

3. Мостовий С. Аналіз методів та моделей вирішення задачі взаємоблокування процесів у комп'ютерних системах. *Вісник Хмельницького національного університету. Технічні науки*. 2021, №5. С.27-31. URL: http://doi.org/10.31891/2307-5732-2021-301-5-27-31

4. Мостовий С., Пирч О. Прогнозування стану взаємоблокування процесів у комп'ютерних системах. *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2025, №4. С.284-292. URL: https://doi.org/10.31891/2219-9365-2025-84-32

5. Мостовий С. Система прогнозування стану взаємоблокування процесів у комп'ютерних системах. *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2026, №2. С.418-424. URL: https://doi.org/10.31891/2219-9365-2026-86-49

Праці, які засвідчують апробацію матеріалів дисертації

6. Мостовий С.В. Архітектура системи прогнозування взаємоблокування процесів у комп'ютерній системі. *Збірник наукових праць за матеріалами XV Всеукраїнської науково-практичної конференції "Актуальні проблеми комп'ютерних наук АПКН-2023"*, м.Хмельницький, 17-18 листопада 2023р. / ХНУ Хмельницький, 2023. С. 343-345. URL: https://kn.khmnu.edu.ua/wp-content/uploads/sites/18/apkn-2023_corpuspaper.pdf

7. Мостовий С.В. Модель та метод побудови сигнатури процесу для прогнозування взаємоблокувань у комп'ютерній системі. *Збірник наукових праць за матеріалами XVI Всеукраїнської науково-практичної конференції "Актуальні проблеми комп'ютерних наук АПКН-2024"*, м.Хмельницький, 15-16 листопада 2024р. / ХНУ Хмельницький, 2024. С. 580-583. URL: https://kn.khmnu.edu.ua/wp-content/uploads/sites/18/apkn-2024-corpuspaper.pdf

8. Мостовий С.В. Метод прогнозування стану взаємоблокування процесів у комп'ютерній системі. ITSec: *Безпека інформаційних технологій: матеріали XV Міжнар.наук.-техн. конф.*, м. Тернопіль, 27-29 трав. 2026 р. / Тернопіль-Київ:ТНТУ-ДУІКТ, 2026. С. 378-380 URL: https://itsec.tntu.edu.ua/#conference-abstracts

9. Oleg Savenko, Mykola Fedula, Jan Rabchan, Kyrylo Voznyi, Serhii Mostovyi, Anatolii Barabash. The method of parallelization of dynamic sequential system programs using networks of multifaceted processes. *Proceedings of the 7th International Workshop on Intelligent Information Technologies & Systems of Information Security (IntelITSIS-2026)*. / Khmelnytskyi, 2026. P. 20 URL: https://ceur-ws.org/Vol-4242/paper10.pdf (індексована в наукометричній базі Scopus)

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 18
ВСТУП 19
РОЗДІЛ 1 АНАЛІЗ ВЗАЄМОБЛОКУВАНЬ ПРОЦЕСІВ У КОМП’ЮТЕРНИХ СИСТЕМАХ 31
1.1 Проблема взаємоблокування процесів у комп’ютерних системах 31
1.2 Аналіз методів вирішення задачі взаємоблокування 41
1.3 Методи машинного навчання та інтелектуального аналізу для прогнозування взаємоблокування процесів 53
1.4 Постановка задачі дослідження 57
1.5 Висновки до першого розділу 61
РОЗДІЛ 2 МОДЕЛІ ПРОГНОЗУВАННЯ СТАНУ ВЗАЄМОБЛОКУВАННЯ 63
2.1 Архітектура системи прогнозування взаємоблокування процесів 63
2.2 Модель сигнатури процесу в комп'ютерних системах 75
2.3 Модель стану комп’ютерних систем 82
2.4 Модель процесу побудови бази нечітких правил 85
2.5 Модель прогнозування взаємоблокування процесів в комп'ютерних системах 103
2.6 Висновки до другого розділу 111
РОЗДІЛ 3 МЕТОДИ ПОБУДОВИ СИГНАТУРИ ПРОЦЕСУ І БАЗИ НЕЧІТКИХ ПРАВИЛ ТА ПРОГНОЗУВАННЯ СТАНУ ВЗАЄМОБЛОКУВАННЯ ПРОЦЕСІВ У КОМП’ЮТЕРНИХ СИСТЕМАХ 112
3.1 Метод побудови сигнатури процесу 112
3.2 Метод побудови бази нечітких правил 127
3.3 Метод прогнозування стану взаємоблокування процесів в КС 139

3.4 Висновки до третього розділу 148
РОЗДІЛ 4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ОЦІНЮВАННЯ МЕТОДІВ 150
4.1 Архітектура програмного засобу прогнозування стану взаємоблокування процесів 150
4.2 Оцінка достовірності методу прогнозування взаємоблокувань процесів 160
4.3 Експериментальна оцінка методу прогнозування взаємоблокувань процесів 181
4.4 Висновки до четвертого розділу 192
ВИСНОВКИ 194
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 197
ДОДАТКИ 218
ДОДАТОК А Список публікацій здобувача за темою дисертації 218
ДОДАТОК Б Акти впровадження 220
ДОДАТОК В Інструкція користувача програмного засобу Deadlock Prediction System v1.0 224
ДОДАТОК Г Результати експериментів 233

ПЕРЕЛІК СКОРОЧЕНЬ

АБ – алгоритм банкіра

БД – база даних

ГО – граф очікування

ГПР – граф розподілу ресурсів

ДІ – довірчий інтервал

ЗП – заборона на переривання

ІГН – ігнорування взаємоблокувань

КС – комп’ютерна система

МП – метод прогнозування взаємоблокування

ОС – операційна система

ПВР – примусове вилучення ресурсу або відкат

СКБД – система керування базами даних

API – Application Programming Interface – інтерфейс програмування застосунків

CV – Coefficient of Variation – коефіцієнт варіації

FCM – Fuzzy C-Means – метод нечітких С-середніх

FN – False Negative – хибнонегативний результат

FP – False Positive – хибнопозитивний результат

GNN – Graph Neural Network – графова нейронна мережа

GRU – Gated Recurrent Unit – вентильний рекурентний блок

LSTM – Long Short-Term Memory – довга короткочасна пам’ять

PID – Process Identifier – ідентифікатор процесу

PPID – Parent Process Identifier – ідентифікатор батьківського процесу

RNN – Recurrent Neural Network – рекурентна нейронна мережа

RTOS – Real-Time Operating System – операційна система реального часу

TN – True Negative – істиннонегативний результат

TP – True Positive – істиннопозитивний результат

UID – User Identifier – ідентифікатор користувача

ВСТУП

Обгрунтування вибору теми дослідження. Сучасні комп’ютерні системи (КС) характеризуються високим рівнем паралелізму, що зумовлює одночасне опрацювання спільних даних багатьма процесами, внаслідок чого множина одночасно виконуваних процесів формує складні групи взаємопов’язаних задач, які конкурують за спільний пул ресурсів. За фізичною природою такі ресурси поділяють на апаратні та програмні, за можливістю спільного використання – на поділювані й неподілювані, за можливістю примусового вилучення – на відновлювані та невідновлювані, а за кількістю доступних екземплярів – на одноекземплярні та багатоекземплярні [21].

Особливе значення для КС мають неподілювані ресурси, зокрема апаратні або програмні об’єкти, які в певний момент часу можуть використовуватися лише одним процесом і тому потребують монопольного доступу. До них належать окремі пристрої введення-виведення, файли та записи баз даних у режимі модифікації, критичні секції, м’ютекси, семафори й інші об’єкти синхронізації. Якщо такий ресурс уже зайнятий, то інші процеси переходять у стан очікування його звільнення[23].

Критичною для досліджуваного класу комп’ютерних систем є ситуація, за якої група взаємопов’язаних процесів одночасно претендує на підмножину неподілюваних ресурсів: кожний процес утримує принаймні один виділений йому ресурс і очікує звільнення ресурсу, зайнятого іншим процесом цієї групи. За одночасного виконання умов взаємного виключення, утримання й очікування, невідчужуваності ресурсів та циклічного очікування в системі формується замкнений цикл ресурсних залежностей, унаслідок чого процеси не можуть продовжити виконання і переходять у стан взаємоблокування, що спричиняє збільшення часу виконання або відмову в обслуговуванні відповідних задач.

Неузгодженість процедур захоплення та вивільнення неподілюваних ресурсів у межах групи задач є першопричиною виникнення низки деструктивних станів

КС. До них належать ресурсні блокування, невизначено тривале очікування ресурсів окремими процесами, та найбільш критичний стан, яким є взаємоблокування. Збільшення таких конфліктів спричиняє непродуктивне утримання ресурсної множини, експоненційне зростання часу виконання задач, деградацію загальної пропускної здатності системи та порушення регламентованих термінів обслуговування запитів. У граничних випадках формування циклічних залежностей між задачами групи призводить до часткової або повної відмови програмних комплексів. Це вимагає застосування ресурсомістких процедур відновлення: примусового переривання процесів, відкату транзакцій, перезапуску всієї системи [12; 61; 68].

Масштабування кількості паралельних обчислень, висока динаміка навантаження критично ускладнюють підтримання цілісної та актуальної інформації про глобальний стан системи [7; 19; 20; 24; 98]. За таких умов традиційні механізми недопущення або вирішення взаємоблокувань демонструють недостатню ефективність. Відповідно, об’єктивною необхідністю є розроблення нових методів, які забезпечують комплексний аналіз ресурсної поведінки групи задач в умовах монопольного доступу до ресурсів та дозволяють проактивно оцінювати ризик переходу комп’ютерної системи у стан взаємоблокування [25].

Значний внесок у дослідження проблеми взаємоблокування процесів зробили як зарубіжні, так і українські науковці. Теоретичні основи дослідження взаємоблокувань були закладені в роботах Е. Коффмана (Edward G. Coffman), М. Елфіка (Michael J. Elphick) та А. Шошані (Arie Shoshani), які сформулювали базові умови виникнення взаємоблокувань та запропонували підходи до їх аналізу в багатопрограмних системах [14].

Подальший розвиток теорії управління ресурсами та уникнення взаємоблокувань пов'язаний із роботами Е. Дейкстри (Edsger W. Dijkstra), який запропонував алгоритм банкіра для забезпечення безпечного розподілу ресурсів між процесами [22]. Значний вплив на розвиток методів аналізу взаємоблокувань також мали дослідження Дж. Холта (James W. Holt), присвячені використанню графів розподілу

ресурсів для виявлення циклів очікування та аналізу потенційних взаємоблокувань [15].

Вагоме концептуальне значення для систематизації методів боротьби із взаємоблокуваннями мають праці Е. Таненбаума (Andrew S. Tanenbaum) та В. Столлінгса (William Stallings). У своїх фундаментальних публікаціях, зокрема в праці [21], Е. Таненбаум детально дослідив графові моделі управління ресурсами, класифікував алгоритми виявлення тупиків та обґрунтував умови доцільності застосування "алгоритму страуса". В. Столлінгс у науковій роботі [23] розробив підходи до інтегрованого управління взаємоблокуваннями, глибоко систематизувавши механізми їх запобігання, уникнення та виявлення в умовах багатопотокового виконання. Праці цих учених дозволили уніфікувати стратегії обробки тупикових ситуацій та заклали стандарти для проектування підсистем управління процесами в сучасних операційних системах.

Вагомий внесок у розвиток методів виявлення взаємоблокувань у розподілених системах зробили Е. Кнапп (Edgar Knapp) [16], Дж. Холлідей (JoAnne L. Holliday), А. Ель-Аббаді (Amr El Abbadi) [13], а також К. Чанді (K. Mani Chandy), Дж. Місра (Jayadev Misra) та Л. Хаас (Laura Haas) [17], які розробили алгоритми розподіленого виявлення взаємоблокувань на основі графів очікування та механізмів поширення повідомлень між вузлами мережі.

Серед сучасних зарубіжних досліджень особливої уваги заслуговують роботи У. Матура (Umang Mathur), А. Павлоджіанніса (Andreas Pavlogiannis), М. Вісванатана (Mahesh Viswanathan), Ю. Чжана (Yu Zhang), К. Чжана (Kaiwen Zhang) та інших науковців, присвячені питанням прогнозування, статичного та динамічного виявлення взаємоблокувань у багатопотокових програмних системах і середовищах програмування нового покоління [60; 101; 103-107].

Серед українських учених вагомий внесок у розвиток теоретичних і прикладних засад функціонування комп'ютерних систем, розподілених обчислень, управління ресурсами та забезпечення надійності програмно-апаратних комплексів зробили А. О. Мельник, В. Є. Мухін, І. А. Дичка, А. М. Волокита, А. О. Дорошенко, І.

В. Сергієнко, О. А. Пастух, М. З. Згуровський, О. В. Палагін та інші дослідники. Зокрема, А. О. Мельник розвинув наукові засади побудови високопродуктивних комп'ютерних систем, багатопроцесорних архітектур та паралельних обчислень [10-11], В. Є. Мухін та його наукова школа досліджують розподілені комп'ютерні системи, хмарні обчислення, адаптивне управління ресурсами та забезпечення надійності функціонування складних інформаційних систем [3-5], у працях І. А. Дички досліджено теоретичні та прикладні засади побудови паралельних і розподілених комп’ютерних систем, організації обчислювальних процесів та підвищення ефективності використання системних ресурсів [1-2], О. А. Пастух досліджував функціонування програмно-технічних і комп’ютерних систем, організації оброблення даних, керування обчислювальними ресурсами та підвищення надійності їх роботи [158-160], А. М. Волокита досліджував питання аналізу ресурсної залежності, функціонування розподілених обчислювальних середовищ та проблеми взаємоблокувань процесів [6]. Праці зазначених учених створили наукове підґрунтя для розвитку сучасних методів моніторингу станів комп'ютерних систем, аналізу критичних ситуацій та управління ресурсами.

Перспективним напрямом підвищення ефективності управління ресурсами комп'ютерних систем є перехід від реактивних механізмів виявлення взаємоблокувань до проактивних засобів їх прогнозування. На відміну від традиційних підходів, прогнозування дозволяє виявляти потенційно небезпечні стани ще до фактичного виникнення взаємоблокування та своєчасно виконувати коригувальні дії щодо перерозподілу ресурсів або зміни параметрів планування процесів [28].

Аналіз наукових праць показує, що існуючі дослідження переважно зосереджені на розробленні методів запобігання та виявлення взаємоблокувань, тоді як питання прогнозування таких станів досліджено недостатньо. Зокрема, потребують подальшого розвитку методи побудови моделей станів процесів і ресурсів, алгоритми оцінювання ризику виникнення взаємоблокувань та засоби інтеграції прогнозних механізмів у сучасні операційні системи [102].

Отже, наявне науково-технічне протиріччя між необхідністю забезпечення достовірного та своєчасного прогнозування переходу процесів у стан взаємоблокування в умовах динамічної зміни ресурсного стану комп'ютерної системи, з однієї сторони, та з іншої обмеженими можливостями відомих методів, що переважно орієнтовані на виявлення та усунення взаємоблокування після його виникнення або на запобігання йому шляхом застосування обмежень щодо розподілу ресурсів. Наявні підходи недостатньо враховують сукупність ідентифікаційних, ресурсних, часових і поведінкових характеристик процесів, а формування правил оцінювання небезпечних станів значною мірою залежить від експертного втручання. Вирішення зазначеного протиріччя потребує розроблення моделей, методів і засобів, які забезпечують формування інформативних сигнатур процесів, автоматизований аналіз їх ресурсної поведінки та завчасне оцінювання ризику переходу комп'ютерної системи у стан взаємоблокування.

Таким чином, науково-прикладна задача полягає у розробленні методів та засобів достовірного і своєчасного прогнозування стану взаємоблокування процесів у комп'ютерних системах на основі формування та аналізу сигнатур процесів, автоматизованої побудови бази нечітких правил і завчасного оцінювання ризику переходу процесів у стан взаємоблокування.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційне дослідження виконувалось у рамках науково-дослідних тематик Хмельницького національного університету: держбюджетної науково-дослідної теми № 2Б-2008 "Теорія інтелектуального діагностування відмовостійких комп'ютерних систем з програмованою логікою" та міжнародного проєкту "Кібербезпека критично важливої інфраструктури України", що фінансувався USAID (2021-2024рр., договір № 72012120C00002), в яких автор дисертації був виконавцем.

Мета і завдання дослідження.

Об'єкт дослідження – процеси функціонування комп'ютерних систем в умовах конкурентного використання системних ресурсів процесами, що можуть призводити до виникнення стану взаємоблокування.

Предмет дослідження – моделі, методи та засоби побудови сигнатур процесів, формування бази нечітких правил і прогнозування стану взаємоблокування процесів у комп'ютерних системах на основі аналізу ресурсної взаємодії процесів, нечіткої кластеризації та нечіткого логічного висновку.

Метою дисертаційної роботи є підвищення достовірності та забезпечення своєчасності прогнозування стану взаємоблокування процесів у комп'ютерних системах шляхом розроблення методу та засобів аналізу сигнатур процесів, автоматизованого формування бази нечітких правил і завчасного оцінювання ризику переходу процесів у стан взаємоблокування.

Для вирішення вказаної проблематики необхідне розв'язання таких завдань:

1) провести аналіз відомих моделей, методів та алгоритмів вирішення задачі взаємоблокування процесів у комп'ютерних системах і визначити їхні обмеження щодо завчасного прогнозування стану взаємоблокування, урахування ресурсної поведінки процесів та придатності до реалізації в сучасних операційних системах;

2) удосконалити архітектуру засобів прогнозування стану взаємоблокування процесів у комп'ютерних системах, яка забезпечує збирання даних про процеси та ресурси, формування сигнатур процесів, побудову бази нечітких правил і виконання прогнозування ризику переходу процесів у стан взаємоблокування;

3) удосконалити метод побудови сигнатури процесу в комп'ютерних системах шляхом формалізованого перетворення ідентифікаційних, ресурсних, часових та поведінкових характеристик процесу у впорядкований вектор ознак, придатний для подальшої кластеризації, нечіткого аналізу та прогнозування стану взаємоблокування;

4) удосконалити метод побудови бази нечітких правил для прогнозування стану взаємоблокування процесів на основі нечіткої кластеризації Fuzzy C-Means, що забезпечуватиме автоматизоване виділення типових сценаріїв ресурсної поведінки процесів і формування правил нечіткого логічного висновку з меншим рівнем залежності від експертного налаштування;

5) розробити метод прогнозування стану взаємоблокування процесів у комп’ютерній системі, який на основі сигнатур процесів, поточного ресурсного стану системи та сформованої бази нечітких правил забезпечуватиме завчасне оцінювання ризику переходу процесів у стан взаємоблокування і формування підстав для прийняття превентивних рішень щодо керування ресурсами;

6) розробити програмні засоби реалізації запропонованих методів і провести експериментальну перевірку достовірності та своєчасності прогнозування стану взаємоблокування процесів з оцінюванням якості за показниками Accuracy, Precision, Recall, Specificity, F1-score, а також виконати оцінювання часової складності та порівняння розробленого методу з відомими підходами.

Методи досліджень. Для розв'язання поставлених у дисертаційній роботі завдань використано комплекс взаємопов'язаних теоретичних та експериментальних методів дослідження:

1) методи системного аналізу та теорії операційних систем для дослідження процесів функціонування комп'ютерних систем, аналізу механізмів розподілу ресурсів, причин виникнення взаємоблокувань та формалізації задачі прогнозування стану взаємоблокування процесів;

2) методи математичного моделювання для розроблення моделі сигнатури процесу, моделі стану комп'ютерної системи, моделі формування бази нечітких правил та моделі прогнозування стану взаємоблокування процесів;

3) методи теорії нечітких множин і нечіткої логіки для формалізації невизначеності параметрів функціонування процесів, побудови функцій належності, формування бази нечітких правил та реалізації нечіткого логічного висновку під час оцінювання ризику виникнення взаємоблокування;

4) методи теорії ймовірностей та математичної статистики для оцінювання достовірності результатів прогнозування, аналізу експериментальних даних та статистичної обробки результатів досліджень;

5) методи обчислювального експерименту та імітаційного моделювання для перевірки працездатності розроблених моделей, методів і алгоритмів прогнозування взаємоблокувань процесів у різних сценаріях функціонування комп'ютерної системи;

6) методи оцінювання якості моделей класифікації для визначення ефективності запропонованого методу прогнозування за показниками Accuracy, Precision, Recall, Specificity та F1-score.

Наукова новизна отриманих результатів. На підставі теоретичних і практичних досліджень, виконаних у дисертаційній роботі, отримано такі нові наукові результати:

1) вперше розроблено метод прогнозування стану взаємоблокування процесів у комп’ютерних системах, який, на відміну від відомих методів виявлення взаємоблокувань після їх виникнення або запобігання їм на основі наперед заданих обмежень розподілу ресурсів, забезпечує завчасне оцінювання ризику переходу процесів у стан взаємоблокування шляхом обробки сигнатур процесів базою нечітких правил, сформованою за результатами кластеризації, що дає змогу враховувати поточний ресурсний стан системи, особливості взаємодії процесів із ресурсами та формувати підстави для прийняття превентивних рішень щодо керування ресурсами комп’ютерної системи;

2) удосконалено метод побудови сигнатури процесу в комп’ютерних системах, який, на відміну від підходів, що описують процес через окремі системні параметри або поточний стан виконання, передбачає формалізоване перетворення ідентифікаційних, ресурсних, часових та поведінкових характеристик процесу у впорядкований вектор ознак із урахуванням зайнятих ресурсів, запитуваного ресурсу та особливостей ресурсної взаємодії, що дає змогу сформувати інформативний опис процесу для подальшої кластеризації, нечіткого аналізу та прогнозування стану взаємоблокування;

3) удосконалено метод побудови бази нечітких правил для прогнозування стану взаємоблокування процесів у комп’ютерних системах, який, на відміну від

відомих підходів формування правил на основі повного експертного налаштування, використовує нечітку кластеризацію Fuzzy C-Means для автоматизованого виділення типових сценаріїв ресурсної поведінки процесів з подальшим формуванням правил нечіткого логічного висновку, що дає змогу зменшити залежність від експертних оцінок, врахувати приховані закономірності у вибірці станів процесів та підвищити адаптивність бази правил до зміни поведінки процесів у комп'ютерній системі;

4) удосконалено архітектуру засобів прогнозування стану взаємоблокування процесів у комп'ютерних системах, яка, на відміну від відомих підходів, орієнтованих переважно на виявлення або усунення взаємоблокування після його виникнення, передбачає поєднання модулів збору даних про процеси та ресурси, побудови сигнатур процесів, формування бази нечітких правил і прогнозування стану взаємоблокування, що забезпечує завчасне оцінювання ризику переходу процесів у стан взаємоблокування та створює основу для прийняття превентивних рішень щодо керування ресурсами комп'ютерної системи.

Практичне значення отриманих результатів.

У результаті виконаного дисертаційного дослідження розроблено програмні засоби прогнозування стану взаємоблокування процесів у комп'ютерних системах, які забезпечують отримання системних подій, формування та збереження сигнатур процесів, використання автоматизовано сформованої бази нечітких правил, оцінювання ризику переходу процесів у стан взаємоблокування та формування підстав для превентивного керування ресурсами. Розділення процедур навчання та оперативного прогнозування дає змогу виконувати ресурсомісткі процедури кластеризації та формування бази правил поза контуром оперативного розподілу ресурсів, а під час функціонування системи використовувати сформовану базу правил для оцінювання поточних станів процесів. Результати прогнозування можуть бути використані в операційних системах і засобах керування ресурсами для посиленого моніторингу, зміни черговості доступу до ресурсів, відкладення ресурсного запиту та формування інших превентивних керуючих впливів.

Результати експериментальних досліджень підтвердили практичну придатність розроблених методів і засобів для достовірного та своєчасного прогнозування стану взаємоблокування процесів. За результатами імітаційного моделювання для різної кількості ресурсів і частки процесів, що потрапляють у стан взаємоблокування, забезпечено завершення 99,85–99,95 % процесів; значення Accuracy становило 99,65–99,88 %, Recall – 99,00 %, Precision – 98,61–98,67 %, F1-score – 98,80–98,84 %, при цьому частка пропущених небезпечних процесів не перевищувала 1,00 %. Експериментальна перевірка на реальній комп'ютерній системі підтвердила стабільність результатів: Accuracy становила 99,65–99,82 %, Precision – 98,39–98,49 %, F1-score – 98,19–98,25 %, а прогноз формувався в середньому за 3,1–3,6 мс до фактичного виникнення взаємоблокування. Порівняно з методом на основі графа розподілу ресурсів середній час обробки скоротився на 8,6–8,8 %, а пропускна здатність зросла на 9,4 %.

Результати дисертаційної роботи впроваджено в ПП "Авіві" (довідка від 29.06.2026р. м.Хмельницький), ТОВ "ІТТ" (акт від 25.06.2026р. м.Хмельницький), у освітній процес Хмельницького національного університету (акт від 26.06.2026р.) при викладанні дисциплін на кафедрі кібербезпеки для здобувачів спеціальності F5 Кібербезпека, зокрема в курсах "Основи інформаційної безпеки", "Операційні системи та технології їх захисту" та на кафедрі комп'ютерної інженерії та інформаційних систем для здобувачів спеціальності F7 Комп'ютерна інженерія, зокрема в курсах "Системне програмне забезпечення", "Технологія проєктування спеціалізованих операційних систем".

У дисертаційній роботі розроблено методи, моделі, алгоритми та програмні засоби прогнозування стану взаємоблокування процесів в комп'ютерних системах.

Особистий внесок здобувача. Усі наукові результати дисертаційної роботи, що виносяться на захист, отримані автором особисто. У роботах, опублікованих автором одноосібно, отримано наступні результати: [119] – розроблено засоби прогнозування стану взаємоблокування процесів в КС; [120] – розроблено архітектуру

системи прогнозування взаємоблокування процесів у КС; [121] – розроблено модель та метод побудови сигнатури процесу; [122] – розроблено метод прогнозування стану взаємоблокування процесів в КС; [123] - запропонована концепція вирішення науково-прикладної проблеми.

У роботах, опублікованих у співавторстві, автору належать основні ідеї, теоретична та практична розробка положень, відображених у характеристиці наукової новизни отриманих результатів, а саме: [116] – розроблена система прогнозування стану процесів та алгоритми її роботи; [117] – розроблено метод прогнозування входження процесів у взаємоблокування; [118] – розроблено метод побудови сигнатури процесу, метод побудови бази нечітких правил, метод прогнозування взаємоблокування; [124] – розроблено модель сигнатури процесу.

У роботах, які опубліковані у співавторстві, співавторам належать такі результати: [116] – проведено оцінку часової складності алгоритму прогнозування взаємоблокування (Кльоц Ю.П.); [117] – реалізовано постановку експерименту (Кльоц Ю.П., Нічепорук А.О.), опрацьовано результати експерименту (Яшина О., Нічепорук Ю.О.); [118] – проведено аналіз відомих рішень взаємоблокування (Пирч О.В.); [124] – запропоновано підхід до подання послідовних системних програм мережею багатогранних процесів та визначення ділянок їх паралельного виконання (O.Savenko, M.Fedula), формалізовано умови паралельного виконання програмних операцій з урахуванням залежностей між даними та обчислювальними ресурсами (J.Rabchan), реалізовано програмну модель мережі взаємодіючих процесів і постановку експериментального дослідження (K.Voznyi, A.Barabash).

Апробація результатів дисертації. Апробацію основних положень, ідей, висновків дисертаційної роботи проведено на розширеному науковому семінарі кафедри кібербезпеки у Хмельницькому національному університеті. Наукові результати роботи доповідалися також на: XV Всеукраїнській науково-практичній конференції "Актуальні проблеми комп’ютерних наук АПКН-2023" (Хмельницький, 17-18 листопада 2023р.); XVI Всеукраїнській науково-практичній конференції "Актуальні проблеми комп’ютерних наук АПКН-2024" (Хмельницький, 15-16 листопада

2024р.); XV міжнародній науково-технічній конференції "Безпека інформаційних технологій" (Тернопіль, 27-29 травня 2026р.); 7th International Workshop on Intelligent Information Technologies & Systems of Information Security (IntelITSIS-2026) (Khmelnytskyi, Ukraine, July 3, 2026).

Публікації. За результатами проведених досліджень основні наукові результати опубліковано у п’яти наукових статтях у трьох фахових наукових журналах України [116; 118; 119; 123] та одному міжнародному науковому журналі [117], що індексується в наукометричній базі Scopus. Апробація засвідчена публікаціями чотирьох праць в матеріалах міжнародних конференцій [120-122; 124], одна з яких індексована в наукометричній базі Scopus [124].

Структура та обсяг дисертації. Дисертаційна робота складається з анотації, змісту, переліку умовних скорочень, вступу, чотирьох розділів, висновку, списку використаних джерел та шести додатків. Повний обсяг роботи містить 233 сторінки друкованого тексту, з них анотація – на 14 с., зміст – на 2 с., перелік скорочень – на 1 с., основний текст – на 178 с., список із 160 використаних джерел – на 21 с., додатки – на 16 с. Дисертація містить 28 рисунків та 18 таблиць.

РОЗДІЛ 1
АНАЛІЗ ВЗАЄМОБЛОКУВАНЬ ПРОЦЕСІВ У КОМП'ЮТЕРНИХ СИСТЕМАХ

1.1 Проблема взаємоблокування процесів у комп'ютерних системах

Функціонування будь-якої комп'ютерної системи ґрунтується на використанні певної множини апаратних і програмних ресурсів, необхідних для виконання процесів та забезпечення взаємодії між компонентами системи. В умовах багатозадачного режиму роботи одночасно можуть виконуватися десятки або сотні процесів, які конкурують за доступ до обмежених ресурсів. Саме тому ефективне керування ресурсами є одним із ключових завдань операційної системи. Від правильності організації розподілу ресурсів залежить не лише продуктивність системи, а й можливість уникнення критичних ситуацій, пов'язаних із взаємним блокуванням процесів. Дослідження проблеми взаємоблокувань показують, що основною причиною їх виникнення є конкуренція процесів за обмежені ресурси та неузгодженість механізмів їх виділення [8; 128].

У теорії операційних систем під ресурсом розуміють будь-який об'єкт апаратного або програмного забезпечення, який може бути виділений процесу для виконання певних операцій і використовується ним протягом деякого інтервалу часу [22]. До ресурсів належать процесорний час, оперативна пам'ять, зовнішні накопичувачі даних, мережеві інтерфейси, файли, записи баз даних, засоби синхронізації, а також різноманітні логічні об'єкти програмного середовища. Особливістю ресурсів є те, що їх кількість завжди обмежена, тоді як потреби процесів у ресурсах можуть суттєво перевищувати наявні можливості системи. У зв'язку з цим операційна система повинна здійснювати облік ресурсів, контролювати їх розподіл та забезпечувати справедливий доступ до них усіх процесів.

Для дослідження причин виникнення взаємоблокувань важливим є аналіз властивостей ресурсів та їх класифікація. Залежно від характеру використання ресурси комп'ютерної системи поділяють за кількома критеріями: фізичною природою, можливістю спільного використання, можливістю примусового вилучення та

кількістю доступних екземплярів. Кожна з наведених характеристик безпосередньо впливає на ймовірність виникнення взаємоблокувань та визначає складність їх усунення.

За фізичною природою ресурси прийнято поділяти на апаратні та програмні [11]. До апаратних ресурсів належать центральний процесор, оперативна пам’ять, пристрої введення-виведення, мережеве обладнання, зовнішні накопичувачі та інші фізичні компоненти обчислювальної системи. Особливості використання апаратних ресурсів значною мірою визначаються архітектурною організацією обчислювальних компонентів, структурою процесорних ядер та способом виконання спеціалізованих обчислювальних операцій [10]. Програмні ресурси являють собою об’єкти програмного середовища, доступ до яких регулюється операційною системою або прикладними програмами. До них належать файли, бібліотеки, семафори, м’ютекси, сокети, області спільної пам’яті та записи баз даних. Незважаючи на різну природу походження, як апаратні, так і програмні ресурси можуть бути джерелом виникнення взаємоблокувань, якщо доступ до них організований неналежним чином.

Важливу роль у виникненні взаємоблокувань відіграє можливість спільного використання ресурсів. За цією ознакою всі ресурси поділяються на поділювані та неподілювані. Поділювані ресурси допускають одночасне використання кількома процесами без взаємного впливу на результати їх роботи. Наприклад, декілька процесів можуть одночасно читати один і той самий файл або використовувати спільну бібліотеку функцій без виникнення конфліктів. У таких випадках проблема взаємоблокувань практично не виникає, оскільки відсутня необхідність монопольного володіння ресурсом [21].

На відміну від поділюваних, неподілювані ресурси можуть використовуватися лише одним процесом у певний момент часу. Якщо ресурс уже зайнятий, усі інші процеси повинні очікувати його звільнення. Саме ця особливість створює передумови для формування циклічних залежностей між процесами. Класичним при-

кладом неподілюваного ресурсу є принтер. Якщо один процес виконує друк документа, інші процеси не можуть використовувати пристрій до завершення поточної операції. Аналогічна ситуація спостерігається при роботі з критичними секціями програмного коду та механізмами блокування записів баз даних.

Класифікацію ресурсів показано на рисунку 1.1.



Рисунок 1.1 – Класифікація ресурсів комп'ютерної системи [23]

Окрему категорію становлять відновлювані та невідновлювані ресурси [128]. Відновлюваними називаються ресурси, які можуть бути тимчасово вилучені у процесу без втрати результатів його роботи. Наприклад, сторінки оперативної пам'яті можуть бути переміщені до файлу підкачки, після чого процес продовжить виконання після повторного отримання необхідних даних. Невідновлювані ресурси не допускають такого механізму перерозподілу. Примусове вилучення невідновлюваного ресурсу зазвичай призводить до втрати інформації або порушення логічної ці-

лісності процесу. Саме тому операційні системи повинні застосовувати більш складні алгоритми керування такими ресурсами, а ризик виникнення взаємоблокувань у цьому випадку суттєво зростає.

Ще однією важливою характеристикою є кількість доступних екземплярів ресурсу. Одноекземплярні ресурси існують у системі лише в одному примірнику, що значно підвищує рівень конкуренції між процесами. Прикладом може бути спеціалізований пристрій введення-виведення або окремий системний файл. Якщо процес отримав такий ресурс, усі інші процеси змушені очікувати його звільнення. Багатоекземплярні ресурси представлені декількома однаковими одиницями, що дозволяє зменшити інтенсивність конкуренції. Однак навіть у цьому випадку взаємоблокування залишаються можливими, якщо кількість доступних екземплярів є меншою за сукупні потреби процесів.

З метою опису життєвого циклу процесів використовується модель його станів. Вона є одним із базових інструментів опису поведінки процесів в операційних системах. Вона визначає можливі стани процесу та правила переходу між ними в процесі виконання.

У моделі з п’ятьма станами (рисунок 1.2) процес може перебувати в одному з таких станів: створення процесу (New), виконання (Running), готовність до виконання (Ready), очікування (Blocked або Waiting) та завершення процесу (Exit) Це дозволяє відображати життєвий цикл процесу від моменту його створення до завершення виконання. Особливий інтерес становить стан Waiting, у якому процес може перебувати необмежено довго внаслідок очікування ресурсу, зайнятого іншим процесом.

Для аналізу взаємодії процесів і ресурсів у теорії операційних систем широко використовуються графи розподілу ресурсів. Такі графи дозволяють формалізувати процес виділення ресурсів та виявляти циклічні залежності між процесами. Вершини графа представляють процеси та ресурси, а дуги відображають запити й факти виділення ресурсів. Приклад відповідної моделі наведено на рисунку 1.3.

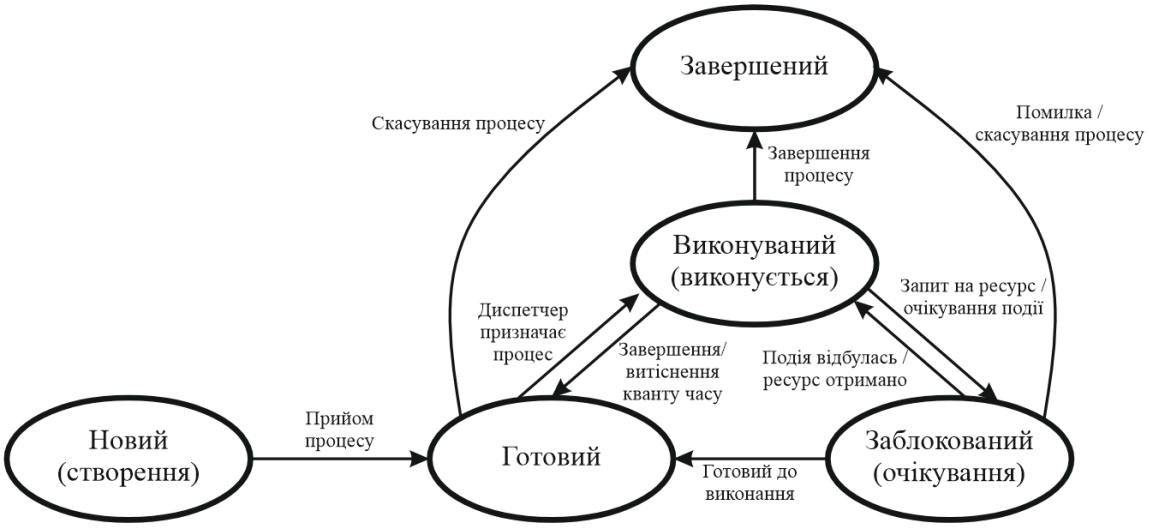


Рисунок 1.2 – Модель процесу з п’ятьма станами [128]

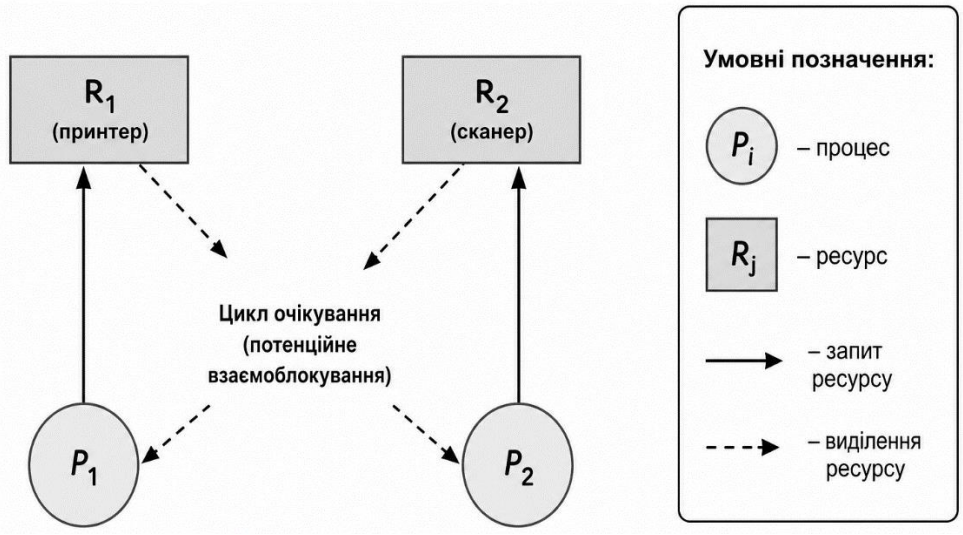


Рисунок 1.3 – Граф розподілу ресурсів між двома процесами[22]

Наявність циклу в графі розподілу ресурсів є однією з ознак взаємоблокування. Для систем, у яких кожен ресурс представлений лише одним екземпляром, цикл у графі означає взаємоблокування. Саме тому графові моделі широко використовуються під час розроблення алгоритмів виявлення та аналізу взаємоблокувань у сучасних операційних системах і системах керування базами даних.

Граф розподілу ресурсів (рисунок 1.4) являє собою орієнтований граф, вершинами якого є процеси та ресурси. При цьому процеси традиційно зображуються

колами, а ресурси – прямокутниками. Дуга від процесу до ресурсу означає запит ресурсу, а дуга від ресурсу до процесу свідчить про факт його виділення.

Перевагою графів розподілу ресурсів є висока наочність та можливість виявлення циклів. Для систем з одноекземплярними ресурсами наявність циклу є достатньою ознакою існування взаємоблокування.

Недоліком моделі є зростання складності графа зі збільшенням кількості процесів і ресурсів. Для великих комп'ютерних систем графи можуть містити тисячі вершин і дуг, що ускладнює їх практичне використання [30].

Для спрощення аналізу взаємоблокувань використовується граф очікування. Він є модифікацією графа розподілу ресурсів і використовується переважно в системах керування базами даних.

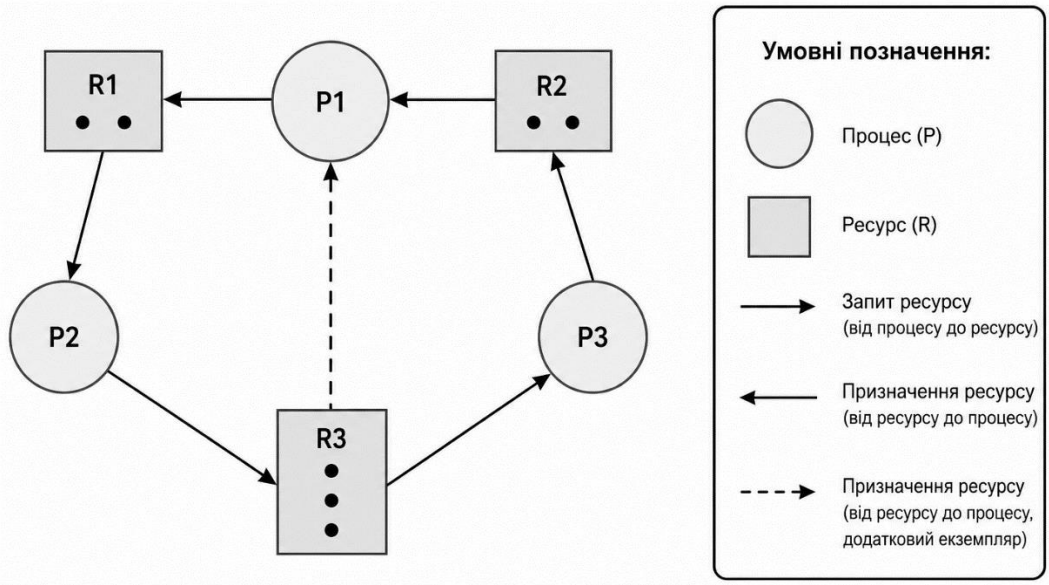


Рисунок 1.4 – Граф розподілу ресурсів та групи процесів[24]

На відміну від графа розподілу ресурсів, граф очікування (рисунок 1.5) містить лише вершини, що відповідають процесам або транзакціям. Дуга між двома вершинами означає, що один процес очікує завершення іншого процесу.

Основною перевагою графа очікування є його компактність. Відсутність вершин ресурсів значно зменшує складність аналізу та дозволяє швидко знаходити цикли очікування. Саме тому більшість сучасних СКБД використовує графи очікування для реалізації алгоритмів виявлення взаємоблокувань.

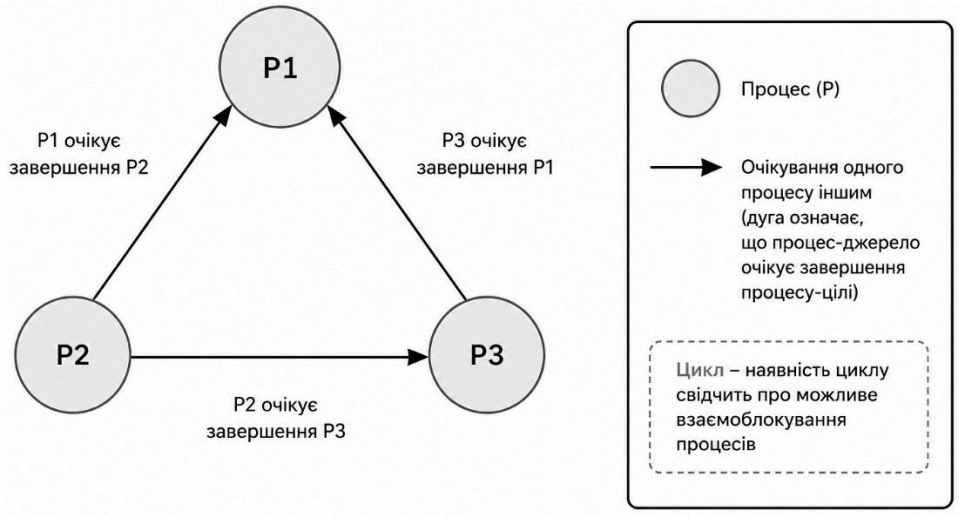


Рисунок 1.5 – Граф очікування процесів[100]

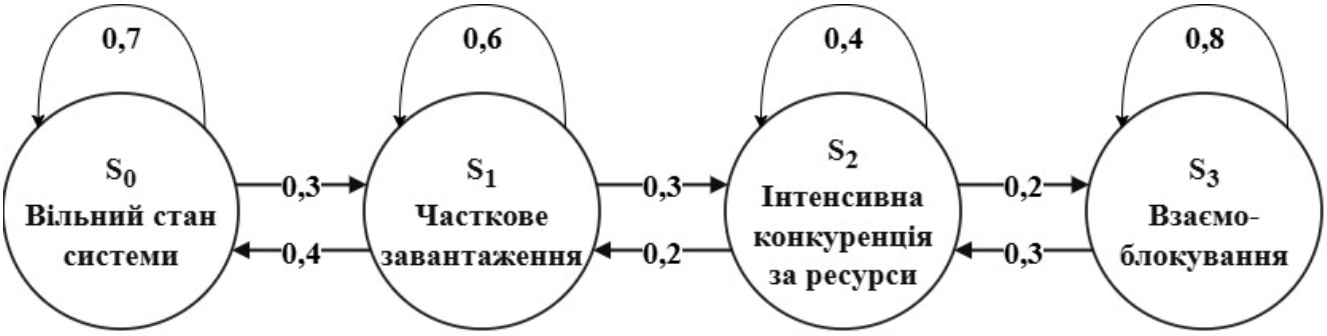
Графові моделі дозволяють визначати факт існування взаємоблокування, однак не дають можливості оцінити ймовірність його виникнення. Для розв’язання цієї задачі застосовуються стохастичні моделі, серед яких найбільшого поширення набули марковські моделі.

Марковські моделі базуються на теорії випадкових процесів і описують поведінку системи через множину станів та ймовірності переходів між ними. Особливістю марковського процесу є те, що майбутній стан системи залежить лише від її поточного стану і не залежить від попередньої історії розвитку [46].

У контексті дослідження взаємоблокувань кожен стан марковської моделі (рис. 1.6) може відповідати певній конфігурації процесів і ресурсів. Наприклад, система може перебувати в нормальному стані, стані підвищеного навантаження, стані конфлікту або стані взаємоблокування. Система зі стануS_0 з імовірністю 0,3 переходить у стан часткового завантаження S_1. Зі стану S_2 існує ймовірність 0,2 перейти у стан взаємоблокування S_3. Із стану взаємоблокування S_3 з імовірністю 0,3 система може повернутися до стану S_2 (наприклад, після завершення одного з процесів), або з імовірністю 0,8 залишатися у цьому стані.

Перевагою марковських моделей є можливість кількісного оцінювання ризику виникнення взаємоблокувань та прогнозування поведінки системи в майбут-

ньому. Такі моделі активно використовуються в дослідженнях високонавантажених інформаційних систем, хмарних платформ і розподілених обчислювальних середовищ.



Звідки / Куди	S_0	S_1	S_2	S_3
S_0	0,7	0,3	0	0
S_1	0,4	0,6	0,3	0
S_2	0	0,2	0,4	0,2
S_3	0	0	0,3	0,8

Рисунок 1.6 – Приклад станів марковської моделі

У кожному випадку механізми формування взаємоблокувань мають свої особливості, однак їх основою залишається конкуренція за обмежені ресурси та порушення узгодженості їх розподілу. Найбільш поширені класи задач, у яких спостерігається виникнення взаємоблокувань, наведено в таблиці 1.1.

Таблиця 1.1 – Класи задач сучасних комп’ютерних систем, у яких виникають взаємоблокування

Клас системи	Типові ресурси	Основні причини виникнення
1	2	3
Операційні системи	файли, пам’ять, пристрої введення-виведення	неправильний порядок захоплення ресурсів

Кінець таблиц 1.1

1	2	3
Багатопроцесні застосунки	mutex, semaphore, monitor	вкладені блокування
Системи керування базами даних	блокування записів і таблиць	конфлікти транзакцій
Системи реального часу	процесорний час, датчики, виконавчі механізми	пріоритетні конфлікти та блокування

Подамо типовий механізм виникнення взаємоблокування між двома процесами графічно (рис. 1.7).

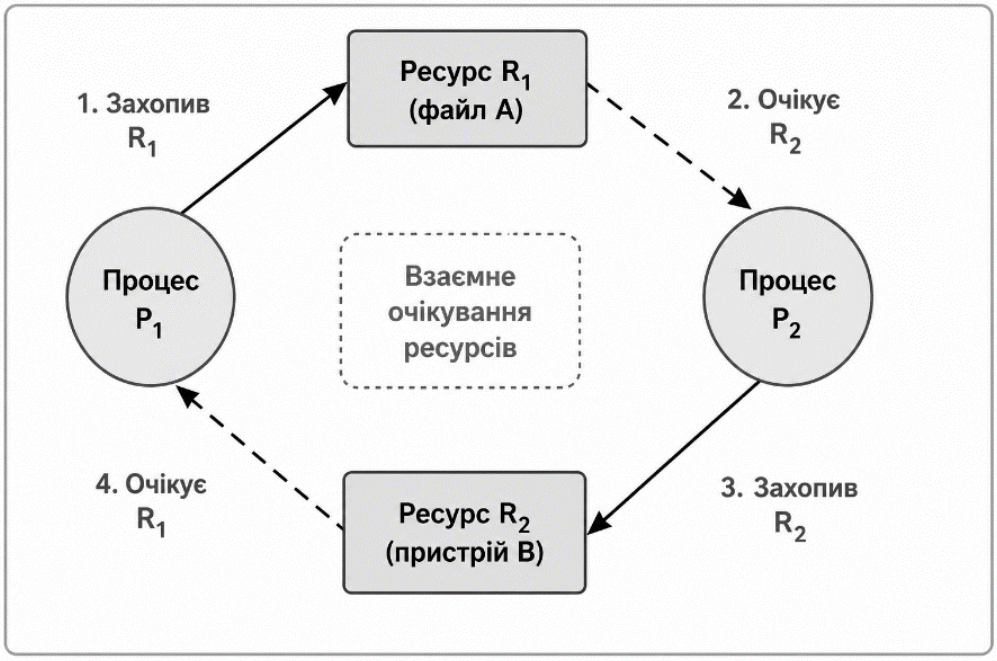


Рисунок 1.7 – Приклад взаємоблокування процесів в операційній системі

Особливе місце серед сучасних комп’ютерних систем займають системи керування базами даних. Для забезпечення цілісності інформації більшість СКБД використовує механізми блокування записів, сторінок або таблиць. Під час виконання транзакції певні об’єкти бази даних можуть бути заблоковані до завершення опе-

рації. Якщо декілька транзакцій одночасно працюють із взаємопов'язаними даними, між ними можуть виникати конфлікти. Класичним прикладом є ситуація, коли перша транзакція заблокувала таблицю А та очікує доступ до таблиці В, тоді як друга транзакція вже володіє таблицею В та очікує доступ до таблиці А. У результаті жодна з транзакцій не може завершити виконання, а система переходить у стан взаємоблокування [13; 159].

У сучасних корпоративних інформаційних системах подібні ситуації виникають досить часто через високий рівень одночасного доступу до даних. Саме тому практично всі промислові СКБД реалізують спеціальні алгоритми виявлення взаємоблокувань та примусового завершення однієї з транзакцій для розриву циклу очікування (рис. 1.8). Такі механізми реалізовані в PostgreSQL, Oracle Database, Microsoft SQL Server та інших системах керування базами даних.

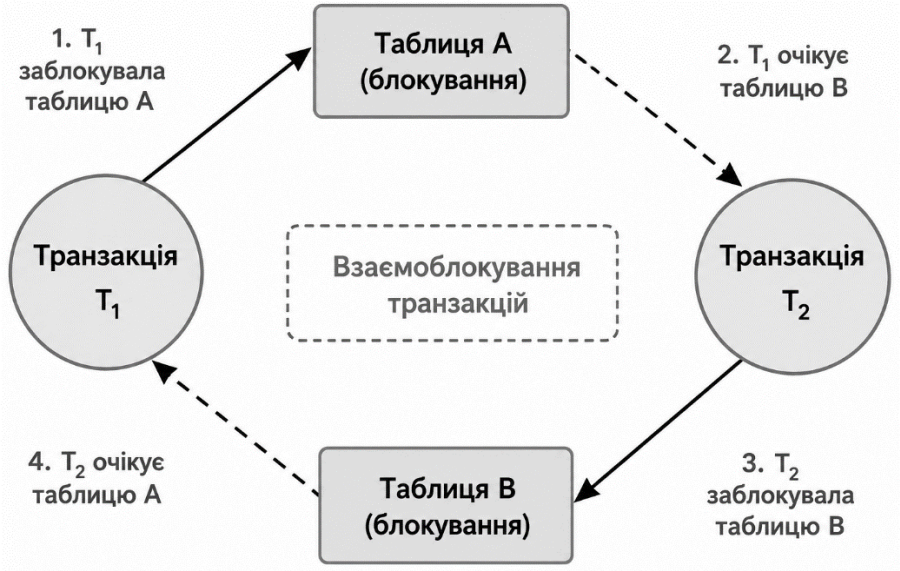


Рисунок 1.8 – Приклад взаємоблокування транзакцій у системі керування базами даних

Проблема взаємоблокування процесів залишається актуальною для сучасних комп'ютерних систем, оскільки її виникнення зумовлюється конкуренцією за обмежені ресурси, складністю міжпроцесної взаємодії та динамічним характером навантаження. Проведений аналіз показує, що наявні графові, стохастичні та систе-

мні механізми здебільшого орієнтовані на виявлення вже сформованих циклів очікування, діагностику помилок синхронізації або відновлення працездатності після критичного стану. Водночас вони не забезпечують універсального завчасного оцінювання ризику взаємоблокування довільних користувацьких процесів під час штатного функціонування системи.

1.2 Аналіз методів вирішення задачі взаємоблокування

Проблема взаємоблокування процесів належить до числа найбільш складних задач управління ресурсами в комп'ютерних системах. Її актуальність зумовлена тим, що виникнення взаємоблокувань призводить до часткової або повної втрати працездатності програмних компонентів, зниження продуктивності системи та порушення виконання критично важливих обчислювальних процесів. У сучасних операційних системах застосовуються різні підходи до боротьби з взаємоблокуваннями, які можна поділити на такі основні групи: методи ігнорування взаємоблокувань, методи запобігання взаємоблокуванням, методи уникнення взаємоблокувань, методи виявлення та усунення взаємоблокувань [128].

Кожна із зазначених груп методів орієнтована на вирішення окремих аспектів проблеми та має власні переваги й обмеження.

Методи ігнорування взаємоблокувань ґрунтуються на припущенні, що взаємоблокування виникають рідко, а витрати на їх постійне запобігання, уникнення або виявлення можуть бути більшими, ніж наслідки окремого збою. При такому підході система не виконує аналізу можливих циклів очікування, а перекладає їх усунення на прикладне програмне забезпечення, адміністратора або механізми перезапуску служб. Перевагою є відсутність значних накладних витрат на підтримання глобального графа ресурсів, матриць стану чи перевірку кожного запиту. Недоліком є те, що взаємоблокування не усувається автоматично на рівні всієї ОС і може призводити до зависання процесів, потоків або окремих служб.

Універсальні операційні системи Linux, Windows, FreeBSD і macOS переважно використовують саме цей підхід для більшості користувацьких процесів і ресурсів. Це пояснюється складністю повного контролю всіх типів ресурсів, різнорідністю прикладних програм, великою кількістю потоків, файлів, сокетів, м'ютексів, транзакцій і зовнішніх пристроїв. Такі ОС надають примітиви синхронізації, тайм-аути, засоби завершення процесів, трасування та діагностики, але не гарантують автоматичної відсутності взаємоблокувань у всіх прикладних сценаріях [21].

Методи запобігання взаємоблокуванням базуються на порушенні однієї з необхідних умов їх виникнення, відомих як умови Коффмана: взаємне виключення, утримання й очікування, невідчужуваність ресурсу та циклічне очікування. Порушення умови взаємного виключення передбачає організацію ресурсу як спільно доступного через кешування, буферизацію, спулінг або режим спільного читання. Перевагою є усунення монопольного захоплення ресурсу, недоліком – обмежена застосовність, оскільки критичні структури, пристрої та об'єкти синхронізації часто потребують виключного доступу. У Linux, Windows, FreeBSD і macOS цей підхід використовується для файлів, кешів, буферів і черг виведення [14].

Порушення умови утримання й очікування полягає в тому, що процес отримує всі потрібні ресурси наперед або звільняє вже зайняті ресурси перед новим запитом. Перевагою є простота запобігання очікуванню із заблокованими ресурсами, недоліком – неефективне резервування та потреба наперед знати ресурсні потреби. У сучасних ОС це переважно реалізується на рівні прикладної логіки, драйверів, серверів, пулів потоків і з'єднань.

Порушення умови невідчужуваності ґрунтується на примусовому вилученні ресурсу або відкаті процесу. Метод придатний для витіснюваних ресурсів, зокрема процесорного часу, сторінок пам'яті та скасованих операцій введення-виведення. Його перевага – можливість розірвати потенційне блокування без завершення процесу; недолік – непридатність для м'ютексів, файлів у режимі запису, пристроїв і транзакцій без спеціального механізму відновлення. У Linux і Windows витіснення

CPU та пам’яті є штатними механізмами, але користувацькі блокування автоматично не вилучаються.

Порушення умови циклічного очікування реалізується через упорядкування ресурсів. Кожному ресурсу або класу ресурсів задається ранг, а захоплення дозволяється лише у визначеному порядку. Перевагою є низька вартість виконання та придатність для ядер ОС, драйверів і багатопотокових серверів; недоліком – складність підтримання повної ієрархії ресурсів. У Linux для контролю такої дисципліни використовується lockdep, у FreeBSD – WITNESS, у Windows – Driver Verifier для драйверів.

Неблокуючі захоплення та тайм-аути обмежують тривалість очікування ресурсу. Операції неблокуючого захоплення повертають відмову, якщо ресурс недоступний, а тайм-аут перериває очікування після заданого часу. Перевагою є простота реалізації, недоліком – відсутність формальної гарантії відсутності взаємоблокування та можливість зациклення повторних спроб без фактичного прогресу. У Linux застосовуються pthread_mutex_trylock і timed-lock механізми POSIX, у Windows – TryEnterCriticalSection і функції очікування з тайм-аутом.

Протоколи пріоритетної стелі та пріоритетного успадкування застосовуються переважно у системах реального часу. Вони обмежують інверсію пріоритетів і знижують ризик циклічного очікування для задач із відомими ресурсами. Недоліком є потреба наперед визначити задачі, пріоритети та критичні секції. Підтримка передбачена POSIX-м’ютексами через PTHREAD_PRIO_PROTECT і PTHREAD_PRIO_INHERIT та використовується в QNX Neutrino й інших RTOS [49].

Заборона на переривання є одним із найпростіших способів забезпечення неподільного виконання критичної ділянки. Суть методу полягає в тому, що процес на час роботи з критичним ресурсом блокує переривання, унаслідок чого інший процес не може перервати його виконання та звернутися до того самого ресурсу. Перевагою є простота реалізації та мінімальні додаткові витрати для коротких кри-

тичних секцій. Недоліком є непридатність для багатопроцесорних та багатоядерних систем у загальному випадку, оскільки заборона переривань на одному ядрі не зупиняє виконання процесів на інших ядрах. Крім того, тривале вимкнення переривань знижує реактивність системи, тому в сучасних ОС Linux, Windows, FreeBSD і macOS цей підхід застосовується лише на рівні ядра, драйверів або низькорівневих критичних операцій, але не як універсальний механізм синхронізації користувацьких процесів [52-53].

Строге чергування передбачає почерговий доступ процесів до критичної секції за допомогою спільної змінної, яка визначає, чия черга виконувати критичну операцію. Метод запобігає одночасному входу двох процесів у критичну секцію та тим самим усуває неконтрольоване захоплення спільного ресурсу. Перевагою є простота логіки та відсутність потреби у складних системних механізмах. Недоліком є неефективне очікування: процес може бути змушений чекати своєї черги навіть тоді, коли інший процес не використовує ресурс. Через це метод погано масштабується, створює зайве активне очікування та в сучасних ОС практично не використовується як штатний механізм, залишаючись переважно навчальною моделлю синхронізації.

Алгоритм Деккера є класичним програмним методом взаємного виключення для двох процесів, який використовує прапорці зацікавленості процесів і змінну черги. Він дає змогу узгодити доступ до критичної секції без спеціальних апаратних інструкцій блокування. Перевагою алгоритму є забезпечення взаємного виключення, відсутність взаємного блокування між двома процесами та врахування ситуації, коли обидва процеси одночасно намагаються увійти до критичної секції. Недоліком є обмеження лише двома процесами, складність порівняно зі строгим чергуванням, активне очікування та залежність від коректного порядку читання і запису змінних у пам’яті. У сучасних Linux, Windows, FreeBSD і macOS алгоритм Деккера не використовується як практичний системний механізм, оскільки його замінено м’ютексами, семафорами та атомарними операціями.

Алгоритм Петерсона є спрощеним програмним методом взаємного виключення для двох процесів, що поєднує прапорці готовності процесів і змінну пріоритету входу. Він забезпечує коректний доступ до критичної секції, якщо обидва процеси діють за визначеним протоколом. Перевагою є простота порівняно з алгоритмом Деккера, формальна зрозумілість і гарантування взаємного виключення для двох учасників. Недоліками є активне очікування, обмеження на кількість процесів, слабка практична придатність для сучасних багатоядерних систем і залежність від моделі пам’яті процесора. Тому в реальних ОС алгоритм Петерсона розглядається переважно як теоретичний приклад, тоді як практична синхронізація в Linux, Windows, FreeBSD і macOS реалізується через системні примітиви блокування та апаратно підтримувані атомарні інструкції [18].

Методи уникнення взаємоблокувань базуються на недопущенні переходу системи в небезпечний стан. Алгоритм банкіра передбачає попереднє оголошення максимальних потреб процесів і перевірку безпечності кожного виділення ресурсу. Перевагою є формальний критерій рішення, недоліком – потреба у точних даних про ресурси. У Linux, Windows, FreeBSD і macOS як штатний механізм ядра не використовується [22].

Метод безпечного стану полягає у виділенні ресурсу лише тоді, коли після цього зберігається можливість завершення всіх процесів. Перевагою є універсальність моделі, недоліком – обчислювальна складність і залежність від точності опису стану. У Linux, Windows Server, FreeBSD і Solaris окремі елементи такого підходу проявляються у квотах, лімітах ресурсів, пулах і контейнерній ізоляції [9; 31; 63].

Граф розподілу ресурсів із дугами-заявками застосовується для ресурсів з одним екземпляром. Ресурс не виділяється, якщо перетворення заявки на виділення створює цикл. Перевагою є простота моделі, недоліком – слабка придатність для динамічних систем і ресурсів із багатьма екземплярами. У Linux, Windows, FreeBSD і macOS використовується переважно як аналітична модель [54].

Попереднє оголошення максимальних потреб задає верхні межі ресурсів, які може запитати процес, без їх обов'язкового попереднього резервування. Перевагою є менше простоювання ресурсів, недоліком – залежність від коректності заявлених меж. Підхід придатний для QNX Neutrino, VxWorks та інших RTOS, а також для контейнерів, квот і спеціалізованих планувальників у Linux та Windows [21].

Динамічний аналіз запитів і відкладене виділення передбачають перевірку кожного запиту під час виконання. Навіть вільний ресурс не надається, якщо це переводить систему в небезпечний стан. Перевагою є врахування поточної ситуації, недоліком – додаткова затримка обробки запитів. У Linux і Windows частіше реалізується на рівні СКБД, брокерів повідомлень, вебсерверів і пулів потоків, а не ядра ОС [26-27; 42; 44; 51].

Прогнозування ризику взаємоблокування використовує ознаки процесів і ресурсів: час очікування, утримувані ресурси, тип запиту, пріоритет, історію блокувань і залежності. Перевагою є придатність для динамічних систем, недоліком – імовірнісний характер результату. У Linux, Windows і FreeBSD може реалізовуватися як агент моніторингу, модуль трасування або компонент керування ресурсами, але не є типовою функцією ядра [29].

Методи виявлення та усунення взаємоблокувань базуються на пошуку вже наявного або потенційного циклу очікування та подальшому звільненні ресурсів [45; 57]. Аналіз графа очікування подає систему як множину процесів або потоків, де дуга означає очікування ресурсу, утримуваного іншим процесом. Цикл у графі є ознакою взаємоблокування. Перевагою є простота інтерпретації, недоліком – потреба точно знати власника ресурсу. У Windows цей підхід реалізовано у Wait Chain Traversal та Analyze wait chain, у Linux може виконуватися через трасування, eBPF-інструменти, профілювальники та аналіз дампів.

Аналіз графа розподілу ресурсів використовує вершини процесів і ресурсів, а дуги позначають запити та фактичні виділення. Для ресурсу з одним екземпляром цикл вказує на взаємоблокування, для кількох екземплярів потрібна додаткова пе-

ревірка доступних одиниць. Перевагою є деталізація залежностей, недоліком – висока вартість підтримання повного графа. У Linux, Windows, FreeBSD і macOS метод частіше застосовується у налагоджувачах, аналізаторах дампів і спеціалізованих моніторах.

Матричний алгоритм виявлення описує стан через матриці виділених і запитуваних ресурсів та вектор доступних ресурсів. Алгоритм визначає процеси, які можуть завершитися, умовно звільняє їх ресурси, а непозначені процеси вважає заблокованими. Перевагою є коректність для ресурсів, що подаються кількісно, недоліком – потреба у повному обліку стану. Метод придатний для СКБД, симуляцій, навчальних моделей і спеціалізованих диспетчерів ресурсів [23].

Періодична перевірка системи передбачає запуск виявлення через задані інтервали або після подій, зокрема перевищення часу очікування, зростання черги чи накопичення заблокованих потоків. Перевагою є контроль накладних витрат, недоліком – затримка між виникненням і виявленням взаємоблокування. У Linux, Windows і FreeBSD такий підхід реалізується зовнішніми службами моніторингу, watchdog-механізмами, трасуванням і діагностикою серверних платформ.

Дослідження порядку блокувань виявляє фактичні або потенційні порушення порядку захоплення ресурсів. Засіб контролю фіксує класи блокувань, порядок їх отримання та можливі циклічні залежності. Перевагою є виявлення дефектів ще під час тестування, недоліком – додаткові додаткові витрати та можливі попередження про потенційні, а не фактичні проблеми. Прикладами є Linux lockdep, FreeBSD WITNESS і Windows Driver Verifier Deadlock Detection.

Примусове завершення процесу усуває взаємоблокування шляхом знищення одного або кількох процесів із циклу очікування. Після завершення процесу ресурси звільняються, і решта процесів може продовжити роботу. Перевагою є простота та швидке відновлення доступності ресурсів, недоліком – ризик втрати даних і порушення незавершених операцій. У Linux, Windows, FreeBSD і macOS завершення процесів є штатною адміністративною дією, але застосовується як крайній захід [128].

Примусове вилучення ресурсу, відкат і перезапуск використовуються для відновлення після виявленого взаємоблокування. Вилучення можливе лише для ресурсів, стан яких можна безпечно зберегти або відновити; відкат повертає процес чи транзакцію до контрольного стану; перезапуск застосовується для служб, які простіше відновити повністю. Перевагою є практичність, недоліком – втрата поточного стану та неусунення первинної причини. У Linux перезапуск може виконувати systemd, у Windows – Service Control Manager, у контейнерних середовищах – політики restart.

Порівняння методів вирішення задачі взаємоблокування доцільно виконувати не лише за класичними алгоритмами, а й за підходами, що відповідають основним групам методів: ігнорування, запобігання, уникнення, виявлення та усунення взаємоблокувань. Для порівняння обрано методи, що реалізовані в ОС сімейства Windows та Linux, а також в SQL серверах. Оцінювання виконано за шкалою від 0 до 5, де більше значення відповідає більшому прояву відповідного параметра для конкретного методу. Такий підхід дозволяє показати не абсолютну перевагу окремого методу, а його практичні обмеження в умовах функціонування сучасних комп'ютерних систем (таблиця 1.2).

Таблиця 1.2 – Порівняльна характеристика методів вирішення задачі взаємоблокування

№	Метод	П1	П2	П3	П4	П5	П6	П7	П8	П9	П10
1	2	3	4	5	6	7	8	9	10	11	12
1	Ігнорування взаємоблокувань	1	0	1	0	0	5	5	0	3	5
2	Порушення умови взаємного виключення	3	2	4	0	1	4	4	4	1	3
3	Попереднє виділення або звільнення ресурсів	2	5	4	0	4	2	3	5	1	2
4	Примусове вилучення ресурсу або відкат	4	2	5	0	3	3	3	4	3	3
5	Упорядкування ресурсів	3	3	3	0	1	4	4	5	1	4
6	Неблокуючі захоплення та тайм-аути	2	1	2	2	2	4	5	3	2	4
7	Пріоритетна стеля / пріоритетне успадкування	4	4	3	1	2	3	4	4	1	2

Кінець таблиці 1.2

1	2	3	4	5	6	7	8	9	10	11	12
8	Заборона на переривання	1	0	1	0	5	1	4	5	2	1
9	Строге чергування	1	0	1	5	4	1	0	3	1	1
10	Алгоритм Деккера	4	0	1	5	3	1	0	4	1	1
11	Алгоритм Петерсона	3	0	1	5	3	1	0	4	1	1
12	Алгоритм банкіра	4	5	5	0	3	2	0	5	1	2
13	Граф розподілу ресурсів із дугами-заявками	3	4	4	0	3	2	1	5	1	2
14	Динамічний аналіз запитів і відкладене виділення	4	3	4	0	3	3	2	5	1	4
15	Аналіз графа очікування	3	2	3	0	2	3	4	3	2	4
16	Матричний алгоритм виявлення	3	3	5	0	2	3	2	3	2	3
17	Періодична перевірка системи	2	2	3	0	2	4	3	2	2	4
18	Діагностика порядку блокувань	3	3	3	0	2	4	4	4	1	3

Позначення параметрів порівняння: П1 – Складність реалізації; П2 – Потреба в попередніх даних про ресурси; П3 – Прив’язка до типу та кількості ресурсів; П4 – Активне очікування; П5 – Зниження продуктивності; П6 – Масштабованість; П7 – Підтримка в сучасних ОС; П8 – Своєчасність запобігання або реакції; П9 – Ризик втрати даних або стану; П10 – Придатність для динамічних систем.

Складність реалізації показує, наскільки важко впровадити метод у реальній системі з багатьма процесами, потоками й ресурсами. Потреба в попередніх даних про ресурси важлива для методів уникнення, оскільки алгоритм банкіра та споріднені підходи потребують знання максимальних потреб процесів. Прив’язка до типу та кількості ресурсів відображає, чи можна застосувати метод до різнорідних ресурсів: пам’яті, файлів, м’ютексів, сокетів, пристроїв і транзакцій. Активне очікування залишено як окремий параметр, оскільки воно суттєво відрізняє строге чергування, алгоритми Деккера й Петерсона від практичних механізмів сучасних ОС.

Параметр зниження продуктивності потрібний для оцінювання практичної ціни методу: наприклад, заборона переривань або попереднє резервування ресурсів можуть помітно знижувати реактивність чи ефективність системи. Масштабованість показує, чи придатний метод для багатьох процесів, а не лише для двох учасників критичної секції. Підтримка в сучасних ОС дає змогу відокремити теоретичні алгоритми від реально використовуваних механізмів Linux, Windows, FreeBSD, macOS та RTOS [18].

Своєчасність запобігання або реакції характеризує, чи метод не допускає небезпечний стан наперед, чи лише реагує після появи проблеми. Ризик втрати даних або стану важливий для методів усунення, зокрема примусового завершення процесів, відкату чи перезапуску служб. Придатність для динамічних систем показує, наскільки метод може працювати в умовах, коли кількість процесів, ресурсів і запитів змінюється під час виконання. Саме цей параметр важливий для обґрунтування потреби в прогнозуванні взаємоблокувань.

Практичне застосування методів вирішення задачі взаємоблокування значною мірою визначається архітектурою операційної системи, призначенням комп'ютерної системи та типами ресурсів, доступ до яких необхідно координувати. Тому, крім порівняння окремих алгоритмів, доцільно проаналізувати реалізовані механізми недопущення, виявлення та усунення взаємоблокувань у сучасних операційних системах та системах керування базами даних.

У таблиці 1.3 наведено порівняння таких систем за методом недопущення взаємоблокування, способом відновлення працездатності, наявністю засобів прогнозування, особливостями реалізації та основними сферами застосування. Це дало змогу визначити, наскільки наявні системні механізми орієнтовані на завчасне оцінювання небезпечного стану або лише на реагування після виникнення блокування.

Таблиця 1.3 – Порівняння механізмів недопущення, вирішення та прогнозування взаємоблокувань у сучасних системах

Система	Метод недопущення	Метод вирішення	Прогнозування взаємоблокувань	Особливості	Де застосовується
1	2	3	4	5	6
Linux 712	Перевірка циклу; контроль порядку; молодший процес примусово завершується	Скасування запиту; повторна спроба	Частково: блокування ядра	Файлові блокування та блокування ядра	Сервери, хмари, вбудовані системи, робочі станції, Android
Microsoft Windows 11 / Windows Server 2025	Аналіз ланцюга очікування; контроль порядку захоплення блокувань у драйверах	Лише діагностика ланцюга очікування та перевірка драйверів	Частково: блокування драйверів	Засоби переважно діагностичні	ПК, корпоративні станції, сервери, доменні та хмарні середовища

Продовження таблиці 1.3

1	2	3	4	5	6
FreeBSD 151	Контроль ієрархії блокувань; перевірка файлових блокувань	Відмова від небезпечного очікування; попередження про неправильний порядок	Частково: блокування ядра	Контроль порядку захоплення блокувань ядра	Сервери, мережеві сервіси, сховища, міжмережеві екрани, вбудовані системи
OpenBSD 79	Контроль порядку захоплення блокувань; перевірка файлових блокувань	Відмова від небезпечного очікування; повідомлення ядра про порушення	Частково: блокування ядра	Вказує проблемні блокування та місця в коді	Захищені сервери, шлюзи, міжмережеві екрани, захищені мережі та мережеві сервіси
NetBSD 101	Перевірка файлових блокувань і м'ютексів; контроль порядку блокувань ядра	Відмова від небезпечного блокування; повідомлення про помилку	Ні	Перевіряє м'ютекси, блокування читання-запису та інші примітиви	Сервери, робочі станції, вбудовані й дослідницькі системи
macOS	Перевірка блокувань POSIX/BSD і м'ютексів	Відмова від небезпечного блокування	Ні	Не аналізує стан усієї системи	Комп'ютери Mac; користувацькі, освітні й корпоративні середовища
Oracle Solaris	Перевірка файлових блокувань і блокувань записів до переходу в очікування	Скасування небезпечної операції блокування	Ні	Перевірка виконується до блокування процесу	Корпоративні сервери SPARC/x86, ZFS, зони Solaris
IBM z/OS 320	Перевірка файлових блокувань і блокувань записів; м'ютекси з контролем помилок	Відмова від небезпечного очікування	Ні	Для звичайних м'ютексів виявлення немає	Мейнфрейми IBM Z, банки, державні й транзакційні системи
QNX Neutrino	М'ютекси з тайм-аутом і контролем помилок; успадкування пріоритету	Припинення очікування; підвищення пріоритету власника	Ні	Керований час очікування	Автомобільні, вбудовані, промислові, медичні та критичні до безпеки системи
MINIX 3 330	Мікроядерна ізоляція служб і ресурсів	Ізоляція або перезапуск служби	Ні	Ізоляція підвищує надійність; прогнозування немає	Освіта, наукові дослідження та дослідні вбудовані системи
IBM z/VM 74	Ізоляція віртуальних машин; планування ресурсів гіпервізором	Зупинка, перезапуск або ізоляція гостьової ОС	Ні	Не аналізує блокування прикладних процесів у гостьових системах	Мейнфрейми IBM Z і LinuxONE; велика кількість гостьових ОС
Oracle Database (СУБД)	Граф очікування; контроль блокувань рядків і таблиць	Відкат оператора запиту; повідомлення про помилку	Ні	Між транзакціями, а не процесами ОС	Корпоративні БД, банки, системи управління, аналітика, центри обробки даних

Кінець таблиці 1.3

1	2	3	4	5	6
PostgreSQL 18 (СУБД)	Граф очікування після завершення заданого часу	Переривання однієї транзакції	Ні	Перевірка починається після заданого часу очікування	Веб- і корпоративні системи, ГІС, аналітика, хмарні БД
Microsoft SQL Server 2025 (СУБД)	Фоновий засіб виявлення; аналіз графа очікування	Вибір і відкат транзакції, обраної жертвою	Ні	Формує граф циклів очікування; враховує пріоритет і вартість відкату	Корпоративні БД, системи управління, аналітика, Azure, локальні сервери

Порівняльний аналіз показав, що в більшості розглянутих систем застосовано контроль порядку захоплення ресурсів, перевірку циклів очікування, тайм-аути та відмову від небезпечного очікування. Усунення сформованого взаємоблокування реалізовано шляхом скасування запиту, примусового завершення процесу, перезапуску служби або відкату транзакції. Зазначені дії забезпечили відновлення працездатності системи, однак створили ризик втрати поточного стану та збільшення часу виконання процесів.

Встановлено, що часткове прогнозування передбачено лише для окремих блокувань ядра або драйверів у Linux, Windows, FreeBSD та OpenBSD. Відповідні засоби забезпечили контроль порядку захоплення блокувань, але не виконали комплексного аналізу ресурсної поведінки довільних користувацьких процесів. У системах керування базами даних аналіз графа очікування виконано для транзакцій після формування циклу очікування або завершення заданого часу очікування.

Отже, проведений аналіз підтвердив обмежений або реактивний характер наявних механізмів вирішення задачі взаємоблокування. Вони не забезпечили завчасного оцінювання ризику переходу довільних користувацьких процесів у стан взаємоблокування під час штатного функціонування комп’ютерної системи. Це обґрунтувало доцільність застосування методів машинного навчання та інтелектуального аналізу даних для аналізу сукупності ознак процесів і ресурсів та виявлення тенденції до формування небезпечної ресурсної ситуації до фактичного виникнення взаємоблокування.

1.3 Методи машинного навчання та інтелектуального аналізу для прогнозування взаємоблокування процесів

Методи машинного навчання та інтелектуального аналізу даних доцільно застосовувати для прогнозування взаємоблокувань у тих випадках, коли стан процесів і ресурсів змінюється динамічно, а повний формальний опис майбутніх запитів неможливий. У такому разі рішення приймається не лише за фактом наявності циклу очікування, а й за сукупністю ознак, що можуть передувати взаємоблокуванню [32-39; 75; 85; 87].

Дерева рішень можуть використовуватися для класифікації станів процесів за ознаками часу очікування, кількості утримуваних ресурсів, типу запитуваного ресурсу, пріоритету та наявних залежностей між процесами. Перевагою є простота інтерпретації, оскільки результат подається у вигляді зрозумілих умовних правил. Недоліком є схильність до перенавчання та обмежена здатність описувати складні залежності. Реалізація в ОС є відносно простою, особливо якщо модель працює поза ядром як модуль моніторингу або компонент диспетчера ресурсів [67; 94].

Методи Random Forest і Gradient Boosting підвищують точність прогнозування за рахунок використання ансамблю моделей. Вони можуть оцінювати ризик взаємоблокування за множиною системних ознак і враховувати складніші комбінації параметрів, ніж окреме дерево рішень. Перевагою є висока якість класифікації на табличних даних, недоліком – більші витрати пам’яті та процесорного часу, а також гірша пояснюваність результату. У середовищі ОС такі методи доцільніше реалізовувати як зовнішній аналітичний модуль, а не як частину ядра [41; 43; 55; 76].

Методи виявлення аномалій, зокрема Isolation Forest та автоенкодери, можуть застосовуватися для пошуку нетипових станів процесів, які відрізняються від нормальної поведінки системи. Їх перевагою є можливість роботи без повної розмітки всіх випадків взаємоблокувань. Недоліком є те, що виявлена аномалія не завжди означає саме взаємоблокування, а може бути наслідком пікового навантаження

або нестандартної, але коректної роботи процесів. Складність реалізації в ОС є середньою або високою через потребу в накопиченні статистики, фільтрації шуму та контролі хибних спрацювань [56; 64; 86].

Рекурентні нейронні мережі, зокрема RNN, LSTM і GRU, орієнтовані на аналіз послідовностей станів процесів у часі. Вони можуть прогнозувати розвиток ситуації до взаємоблокування на основі історії очікувань, захоплень ресурсів і системних викликів. Перевагою є здатність враховувати часову динаміку, недоліком – потреба у великих навчальних вибірках, значних обчислювальних ресурсах і складність пояснення результату. Для ОС така реалізація є складною і доцільна переважно у вигляді дослідницького або зовнішнього прогнозного модуля [50; 59; 70; 79; 83].

Графові методи та графові нейронні мережі природно відповідають задачі взаємоблокування, оскільки процеси, ресурси й очікування можуть бути подані як граф залежностей. Такі методи дозволяють аналізувати структуру взаємодії між процесами та виявляти небезпечні конфігурації. Перевагою є відповідність природі задачі, недоліком – потреба у побудові та постійному оновленні повного графа залежностей. У реальній ОС це ускладнює реалізацію через додаткові витрати на трасування, синхронізацію та зберігання графової структури [48; 62; 66; 81; 93].

Асоціативні правила можуть використовуватися для виявлення типових комбінацій ознак, які часто передують взаємоблокуванню. Наприклад, правила можуть пов’язувати тривале очікування, певний тип ресурсу, кількість утримуваних об’єктів і повторювану послідовність запитів із підвищеним ризиком блокування. Перевагою є простота та інтерпретованість, недоліком – можливе формування великої кількості надлишкових або малозначущих правил. У середовищі ОС метод краще застосовувати на етапі аналізу журналів і навчання, а не безпосередньо в критичному контурі виділення ресурсів.

Метод Fuzzy C-Means може застосовуватися для нечіткої кластеризації станів процесів за ознаками використання ресурсів, часу очікування, пріоритету, кількості

утримуваних ресурсів і типу поточного запиту [77-78]. На відміну від жорсткої класифікації, один стан може належати до кількох кластерів із різними ступенями належності, що є важливим для перехідних станів між нормальною роботою та передумовами взаємоблокування. Перевагою є придатність для нечітких і неповністю визначених станів ОС, недоліком – потреба у нормалізації ознак, виборі кількості кластерів і чутливість до якості навчальних даних [69; 71-74]. Складність реалізації є середньою, оскільки основні обчислення доцільно виконувати на етапі навчання поза ядром ОС [80; 92; 109].

Нечіткі правила, зокрема сформовані за методом Wang–Mendel, дозволяють перетворити навчальні приклади у базу правил для оцінювання ризику взаємоблокування. Такі правила можуть мати вигляд залежностей між часом очікування, кількістю утримуваних ресурсів, пріоритетом процесу та рівнем ризику. Перевагою є пояснюваність і можливість скорочення бази правил до невеликого набору найбільш значущих залежностей. Недоліком є залежність від якості функцій належності, навчальної вибірки та процедури відбору правил. Для ОС метод є прийнятним за умови, що формування правил виконується поза критичним контуром, а в реальному часі використовується лише компактна база правил [40; 47; 58; 65; 89; 99].

Поєднання FCM на етапі навчання та нечіткого логічного висновку на етапі прогнозування є доцільним для задачі прогнозування взаємоблокувань процесів. FCM використовується для виділення типових груп станів і формування навчальної основи для правил, після чого створюється невелика база нечітких правил [88; 96-97]. У робочому режимі система не виконує повторну кластеризацію, а лише застосовує нечіткий логічний висновок до поточного вектора ознак процесу. Перевагою є поєднання адаптивності навчання з малою обчислювальною складністю прогнозу, що робить підхід придатним для роботи поблизу контуру керування ресурсами ОС. Недоліком є потреба у якісному попередньому навчанні, коректному доборі ознак і перевірці стабільності правил на різних навантаженнях [84; 91; 95; 109; 111-112].

Результати аналізу методів машинного навчання та інтелектуального аналізу для прогнозування взаємоблокування процесів представлено в таблиці 1.4.

Таблиця 1.4 – Методи машинного навчання та інтелектуального аналізу

Метод	Опис	Переваги	Недоліки	Складність реалізації в ОС
1	2	3	4	5
Дерева рішень	Класифікують стан процесу за ознаками часу очікування, кількості ресурсів, типу запиту, пріоритету та залежностей.	Простота, зрозумілі правила, легка інтерпретація.	Можливе перенавчання, слабша робота зі складними залежностями.	Низька
Random Forest / Gradient Boosting	Прогнозують ризик взаємоблокування за набором системних ознак процесів і ресурсів.	Вища точність, стійкість до шуму, робота зі складними комбінаціями ознак.	Більші витрати CPU/RAM, нижча пояснюваність.	Середня / висока
Isolation Forest / автоенкодери	Виявляють аномальні стани процесів, які можуть передувати взаємоблокуванню.	Можуть працювати без повної розмітки даних.	Аномалія не завжди означає саме взаємоблокування.	Середня / висока
RNN / LSTM / GRU	Аналізують послідовність станів процесу в часі: очікування, захоплення ресурсів, системні виклики.	Добре враховують часову динаміку.	Потребують багато даних і ресурсів, складно пояснюються.	Висока
Графові методи / GNN	Аналізують граф залежностей "процес – ресурс – очікування".	Природно відповідають структурі задачі взаємоблокування.	Потрібне постійне формування й оновлення графа залежностей.	Висока
Асоціативні правила	Виявляють типові комбінації ознак, що часто передують взаємоблокуванню.	Простота, інтерпретованість, придатність для аналізу журналів.	Можлива велика кількість надлишкових правил.	Низька / середня

Кінець таблиці 1.4

1	2	3	4	5
FCM / Fuzzy C-Means	Нечітко кластеризує стани процесів на нормальні, ризикові та передблокувальні.	Добре працює з перехідними станами і нечіткими межами класів.	Потрібні нормалізація ознак, вибір кількості кластерів і якісні навчальні дані.	Середня
Нечіткі правила / Wang–Mendel	Формує базу нечітких правил для оцінювання ризику взаємоблокування за навчальними прикладами.	Пояснюваність, можливість отримати компактну базу правил.	Якість залежить від функцій належності, вибірки та відбору правил.	Середня
FCM + нечіткий логічний висновок	На етапі навчання FCM виділяє типові стани, після чого формується мала база нечітких правил; на етапі прогнозу виконується лише нечіткий логічний висновок.	Поєднує адаптивність навчання з малою складністю прогнозу; придатний для роботи поблизу контуру керування ресурсами ОС.	Потребує якісного попереднього навчання, добору ознак і перевірки стабільності правил.	Середня на навчанні, низька на прогнозі

Аналіз методів машинного навчання та інтелектуального аналізу даних показав, що їх застосування є доцільним для прогнозування взаємоблокувань процесів у динамічних комп’ютерних системах, де повний формальний опис майбутніх запитів до ресурсів є ускладненим або неможливим. Класичні моделі класифікації, ансамблеві методи, засоби виявлення аномалій, рекурентні та графові нейронні мережі забезпечують різні можливості щодо аналізу станів процесів, однак часто мають обмеження, пов’язані зі складністю реалізації, потребою у великих навчальних вибірках або недостатньою пояснюваністю результатів.

1.4 Постановка задачі дослідження

Комп’ютерні системи функціонують в умовах одночасного виконання значної кількості процесів, які конкурують за використання процесорного часу, опера-

тивної пам’яті, пристроїв введення-виведення, файлів, об’єктів синхронізації та інших системних ресурсів. Ефективність роботи таких систем значною мірою залежить від здатності операційної системи організовувати коректний розподіл ресурсів між процесами, підтримувати узгодженість їх виконання та запобігати ситуаціям, що призводять до втрати керованості або зниження продуктивності.

Однією з найбільш критичних ситуацій, що може виникати під час конкурентного використання ресурсів, є взаємоблокування процесів. Воно виникає тоді, коли два або більше процесів перебувають у стані очікування ресурсів, зайнятих іншими процесами, і жоден із них не може продовжити виконання без зовнішнього втручання. Такий стан знижує ефективність використання ресурсів комп’ютерної системи, ускладнює виконання прикладних задач, може призводити до зависання програмного забезпечення, втрати даних або необхідності примусового завершення процесів.

Відомі підходи до вирішення проблеми взаємоблокування процесів зазвичай поділяють на методи ігнорування, запобігання, уникнення та виявлення взаємоблокувань. Кожен із цих підходів має певні переваги, однак їх практичне застосування в сучасних комп’ютерних системах обмежується низкою чинників. Методи ігнорування не забезпечують цілеспрямованого реагування на небезпечні стани. Методи виявлення реагують переважно після фактичного виникнення взаємоблокування, коли частина процесів уже перебуває у критичному стані. Методи запобігання та уникнення можуть вимагати попереднього знання потреб процесів у ресурсах, жорстких обмежень на порядок їх виділення або складних процедур перевірки безпечності стану системи.

У сучасних операційних системах такі обмеження набувають особливого значення, оскільки кількість одночасно активних процесів, складність програмного забезпечення та інтенсивність звернення до системних ресурсів постійно зростають. Процеси можуть динамічно змінювати свою поведінку, утримувати різні типи ресурсів, формувати нові запити під час виконання та переходити між станами зале-

жно від поточного навантаження системи. Тому використання лише статичних правил розподілу ресурсів або періодичного виявлення вже сформованого взаємоблокування не забезпечує достатньої своєчасності реагування.

З огляду на це актуальною є задача не лише виявлення взаємоблокування після його виникнення, а прогнозування передумов його формування. Для цього необхідно перейти від аналізу окремих фактів блокування процесів до оцінювання їх ресурсної поведінки у динаміці. Такий підхід потребує формалізованого опису процесу, який відображає його ідентифікаційні параметри, часові характеристики, зайняті ресурси, запитувані ресурси та особливості взаємодії з іншими процесами. Саме тому доцільним є використання сигнатури процесу як вектора ознак, придатного для подальшої обробки методами інтелектуального аналізу даних.

Водночас поведінка процесів у комп'ютерній системі не завжди може бути однозначно описана жорсткими логічними умовами. Один і той самий стан може мати різний рівень небезпечності залежно від кількості зайнятих ресурсів, часу перебування процесу в системі, типу запитуваного ресурсу, характеру очікування та взаємодії з іншими процесами. У таких умовах доцільним є використання апарату нечіткої логіки, який дає змогу працювати з частковою належністю станів до небезпечних класів і формувати кількісну оцінку ризику виникнення взаємоблокування.

Окремою проблемою є побудова бази нечітких правил. Повністю ручне формування правил потребує значної кількості експертних оцінок, ускладнюється зі збільшенням кількості ознак і може не враховувати приховані закономірності у вибірці станів процесів. Тому доцільним є використання нечіткої кластеризації Fuzzy C-Means, яка дозволяє автоматизовано виділяти типові сценарії ресурсної поведінки процесів і на їх основі формувати правила нечіткого логічного висновку. Такий підхід зменшує залежність від повного експертного налаштування та забезпечує адаптивність бази правил до змін у поведінці процесів.

Для практичного застосування розробленого підходу необхідна архітектура засобів прогнозування стану взаємоблокування процесів, яка забезпечує взаємодію між середовищем виконання процесів, модулем збору даних, модулем побудови

сигнатур, модулем формування або використання бази нечітких правил та модулем прогнозування. Така архітектура повинна забезпечувати розділення процедур навчання і оперативного прогнозування, щоб у процесі функціонування системи не виникала потреба постійно виконувати ресурсомістку кластеризацію або перебудову всієї бази правил.

Таким чином, основна задача дослідження полягає у розробленні методу та засобів прогнозування стану взаємоблокування процесів у комп'ютерних системах на основі формалізованого подання ресурсної поведінки процесів, автоматизованого формування бази нечітких правил та нечіткого логічного висновку, що забезпечує завчасне оцінювання ризику переходу процесів у стан взаємоблокування.

З урахуванням зазначеного формулюються такі задачі дослідження:

1. Провести аналіз відомих моделей, методів та алгоритмів вирішення задачі взаємоблокування процесів у комп'ютерних системах і визначити їхні обмеження щодо завчасного прогнозування стану взаємоблокування, урахування ресурсної поведінки процесів та придатності до реалізації в сучасних операційних системах.

2. Розробити архітектуру засобів прогнозування стану взаємоблокування процесів у комп'ютерних системах, яка забезпечує збирання даних про процеси та ресурси, формування сигнатур процесів, побудову бази нечітких правил і виконання прогнозування ризику переходу процесів у стан взаємоблокування.

3. Удосконалити метод побудови сигнатури процесу в комп'ютерній системі шляхом формалізованого перетворення ідентифікаційних, ресурсних, часових та поведінкових характеристик процесу у впорядкований вектор ознак, придатний для подальшої кластеризації, нечіткого аналізу та прогнозування стану взаємоблокування.

4. Удосконалити метод побудови бази нечітких правил для прогнозування стану взаємоблокування процесів на основі нечіткої кластеризації Fuzzy C-Means, що забезпечує автоматизоване виділення типових сценаріїв ресурсної поведінки процесів і формування правил нечіткого логічного висновку з меншим рівнем залежності від експертного налаштування.

5. Розробити метод прогнозування стану взаємоблокування процесів у комп'ютерній системі, який на основі сигнатур процесів, поточного ресурсного стану системи та сформованої бази нечітких правил забезпечує завчасне оцінювання ризику переходу процесів у стан взаємоблокування і формування підстав для прийняття превентивних рішень щодо керування ресурсами.

6. Розробити програмні засоби реалізації запропонованих методів і провести експериментальну перевірку достовірності прогнозування стану взаємоблокування процесів з оцінюванням якості за показниками Accuracy, Precision, Recall, Specificity, F1-score, а також виконати оцінювання часової складності та порівняння розробленого методу з відомими підходами.

1.5 Висновки до першого розділу

Дослідження проблеми взаємоблокування процесів у комп'ютерних системах та існуючих методів її вирішення показало наступні результати:

– функціонування сучасних комп'ютерних систем супроводжується інтенсивною конкуренцією процесів за обмежені ресурси, при цьому основною причиною виникнення взаємоблокувань є неузгодженість процедур монопольного захоплення та вивільнення неподілюваних ресурсів;

– існуючі моделі подання станів процесів і ресурсних залежностей (графи розподілу ресурсів, графи очікування, марковські моделі) переважно орієнтовані на виявлення вже сформованих циклів очікування і не забезпечують ефективного завчасного оцінювання ймовірності їх виникнення;

– традиційні методи ігнорування взаємоблокувань не усувають проблему на рівні операційної системи, перекладаючи її на прикладне програмне забезпечення або адміністратора, що неминуче призводить до зависання процесів та деградації пропускної здатності системи;

– застосування методів запобігання та уникнення взаємоблокувань (зокрема, алгоритму банкіра) накладає жорсткі обмеження на порядок розподілу ресурсів,

вимагає точного попереднього знання максимальних потреб процесів і є малопридатним для умов динамічної зміни навантаження;

– методи виявлення та усунення взаємоблокувань мають реактивний характер і здебільшого потребують застосування ресурсомістких процедур відновлення: примусового завершення процесів, відкату задач або перезапуску системи, що створює високий ризик втрати даних;

– наявні підходи недостатньо повно враховують комплексну сукупність ідентифікаційних, ресурсних, часових і поведінкових характеристик процесів під час аналізу їхньої взаємодії;

– встановлено науково-технічне протиріччя між гострою необхідністю забезпечення достовірного та своєчасного прогнозування переходу процесів у стан взаємоблокування та обмеженими можливостями відомих методів, які функціонують лише за фактом настання критичної ситуації або застосовують жорсткі наперед задані ліміти.

Відомі методи та алгоритми не забезпечують універсального завчасного оцінювання ризику взаємоблокування довільних процесів під час штатного функціонування комп’ютерної системи. Враховуючи актуальність проблеми, необхідна розробка нових моделей та методів, які забезпечуватимуть перехід від реактивних механізмів виявлення до проактивних засобів прогнозування небезпечних станів. З метою підвищення достовірності та своєчасності прогнозування стану взаємоблокування процесів актуальним є розроблення методу аналізу ресурсної поведінки на основі формування інформативних сигнатур процесів, а також створення комплексу моделей для автоматизованого оцінювання ризику переходу системи у стан взаємоблокування із використанням нечіткої кластеризації (Fuzzy C-Means) та нечіткого логічного висновку.

Основні результати розділу опубліковані у працях [118; 123].

РОЗДІЛ 2
МОДЕЛІ ПРОГНОЗУВАННЯ СТАНУ ВЗАЄМОБЛОКУВАННЯ ПРОЦЕСІВ У КОМП'ЮТЕРНИХ СИСТЕМАХ

2.1 Архітектура системи прогнозування взаємоблокування процесів

У класичних моделях життєвого циклу процесу, які подано детально в роботі [23] взаємоблокування не виділено окремим базовим станом. У двостановій моделі (стани: виконання, невиконання) воно приховане у стані невиконання, у тристановій (стани: виконання, готовність до виконання, очікування) – у стані очікування, у п'ятистановій (стани: створення, виконання, готовність до виконання, заблокований, завершення) – у стані "заблокований", а у семистановій (стани: створення, виконання, готовність до виконання, заблокований, завершення, готовність до виконання і призупинений, заблокований і призупинений) – у станах "заблокований" або "заблокований і призупинений". Окрім процесів, що вже перебувають у взаємоблокуванні, в даних станах також присутні процеси, що тільки очікують ресурси, які не є неподілюваними. Таке очікування не є критичним для стану КС. Для класичних методів виявлення та усунення взаємоблокувань такі моделі є придатним, бо вони дозволяють здійснювати реактивні заходи по факту виявлення взаємоблокувань. Проте для застосування проактивних заходів така модель не підходить, оскільки не виділяє стан взаємоблокування як окремий. Тому в роботі взаємоблокування позиціоновано як особливу стійку конфігурацію кількох процесів, які перебувають у нескінченому ресурсному очікуванні. Ресурсне очікування розглянуто як ситуацію, за якої процес уже сформував запит до недоступного ресурсу, однак цикл взаємного очікування ще не виник. Сформованому взаємоблокуванню відповідає ситуація, за якої такий цикл уже зафіксовано.

Отже, взаємоблокування є не просто тривалим очікуванням окремого процесу, а колективною властивістю конфігурації "процеси – ресурси". Момент

переходу до ресурсного очікування прийнято за початок інтервалу спостереження, а появу циклу – за граничну подію, до якої має бути отримано прогноз. Прогнозування спрямовано на аналіз процесів, що вже очікують ресурс, але ще не перейшли до взаємоблокування. Саме цей проміжний стан створює часову можливість для завчасного оцінювання небезпеки та підготовки превентивного рішення.

Для подальшого дослідження за основу оберемо п’ятистанову модель процесу, оскільки вона забезпечує необхідний і водночас достатній рівень деталізації життєвого циклу процесу для аналізу ресурсного очікування. На відміну від тристанової моделі, вона окремо відображає створення та завершення процесу. Це дає змогу однозначно визначати межі його існування, відокремлювати процеси, які ще не допущені до виконання або вже не беруть участі в розподілі ресурсів, і зосередити спостереження на переходах між станами виконання, готовності до виконання та блокування. Саме стан "заблокований" містить необхідну для дослідження ознаку (рис. 2.1), бо процес сформував запит до недоступного ресурсу й не може продовжувати виконання.

Семистанна модель додатково розрізняє призупинені стани, пов’язані з вивантаженням процесу з оперативної пам’яті або його адміністративним призупиненням. Такі стани важливі для керування пам’яттю та планування, однак не є обов’язковими для встановлення ресурсних залежностей, що призводять до взаємоблокування. Їх залучення збільшило б складність подання і залежність від особливостей конкретної операційної системи, не додаючи істотної інформації про утримувані й запитувані ресурси. Тому п’ятистанна модель використовується як базова. Вона узгоджується з типовими механізмами ядра ОС, достатньо повно відображає життєвий цикл процесу та дає змогу деталізувати стан блокування без зміни штатної моделі процесів.

Таке подання не розширює штатну модель ядра ОС новим станом. Запропонована система аналізує розвиток ресурсного очікування всередині стану "заблокований" і пов’язує класичні моделі процесу з моделями сигнатури, стану

комп’ютерної системи та оцінювання ризику.

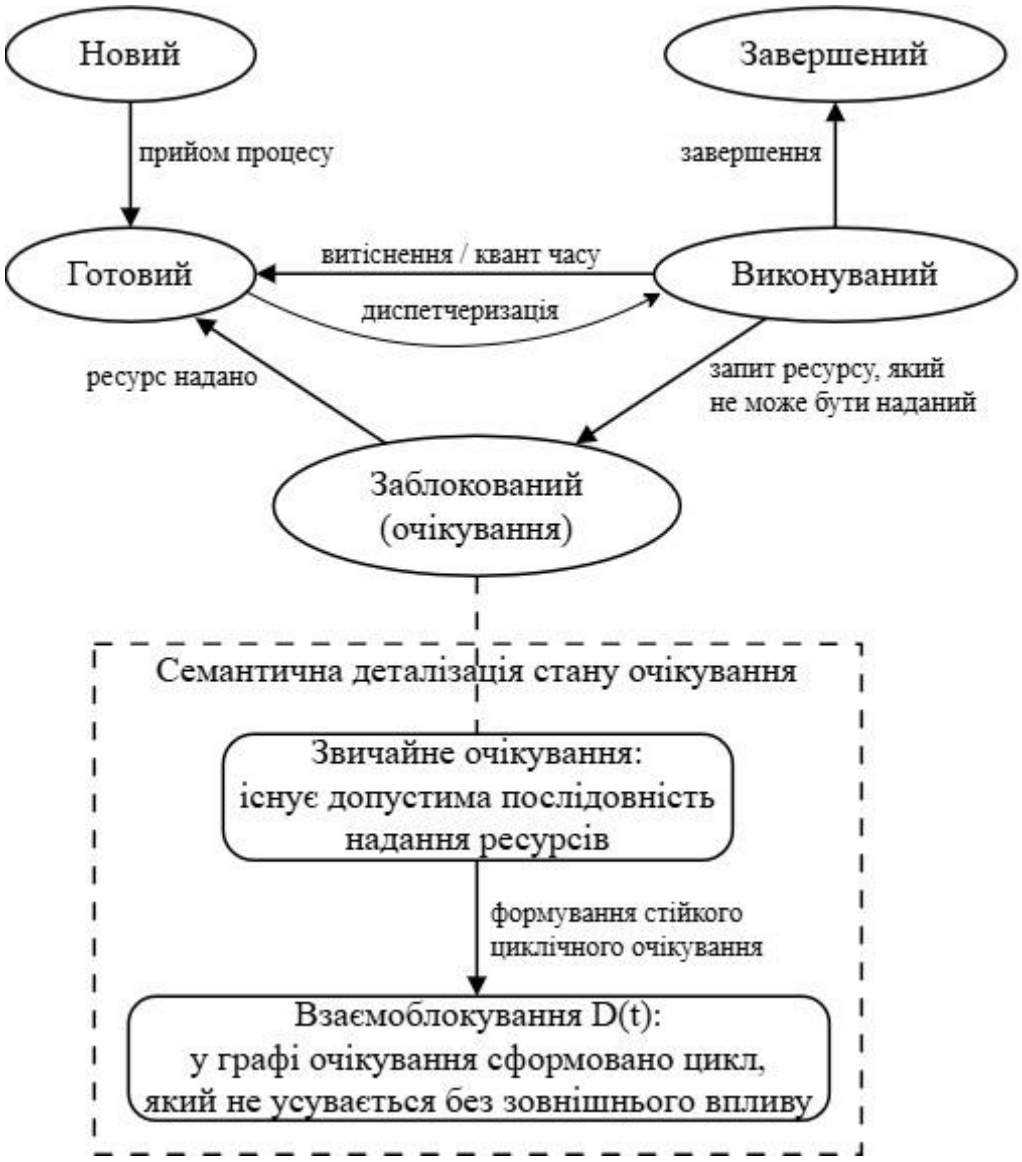


Рисунок 2.1 – Місце стану взаємоблокування у п’ятистановій моделі процесу

Взаємоблокування не вводиться як додатковий стан окремого процесу, оскільки воно виникає лише як властивість сукупності процесів і ресурсів, між якими сформовано циклічне очікування (рис. 2.2). П’ятистанна модель визначає момент переходу процесу до ресурсного очікування, а запропонована система встановлює, чи є це очікування звичайним, чи воно розвивається у напрямі формування взаємоблокування.

Традиційні підходи до вирішення задачі взаємоблокування процесів здебільшого орієнтовані на обмеження порядку розподілу ресурсів або на

виявлення вже сформованого циклу очікування. Перший підхід знижує рівень паралелізму та може призводити до неповного використання ресурсів, тоді як другий потребує відновлення працездатності системи після виникнення критичного стану. За умов змінного складу процесів, динамічної інтенсивності запитів і непередбачуваного порядку захоплення ресурсів доцільним є завчасне оцінювання ризику взаємоблокування [116-119]. Для реалізації такого оцінювання запропоновано архітектуру системи прогнозування, яка забезпечує перетворення низькорівневих даних операційних систем на формалізований опис ресурсної поведінки процесів, формування бази нечітких правил та отримання прогнозної оцінки. До основних вимог належать мінімальне втручання в роботу ядра ОС, урахування динаміки станів, уніфіковане подання різнорідних ресурсів, відокремлення навчання від оперативного прогнозування, пояснюваність результату та можливість періодичного оновлення бази правил [119-120].

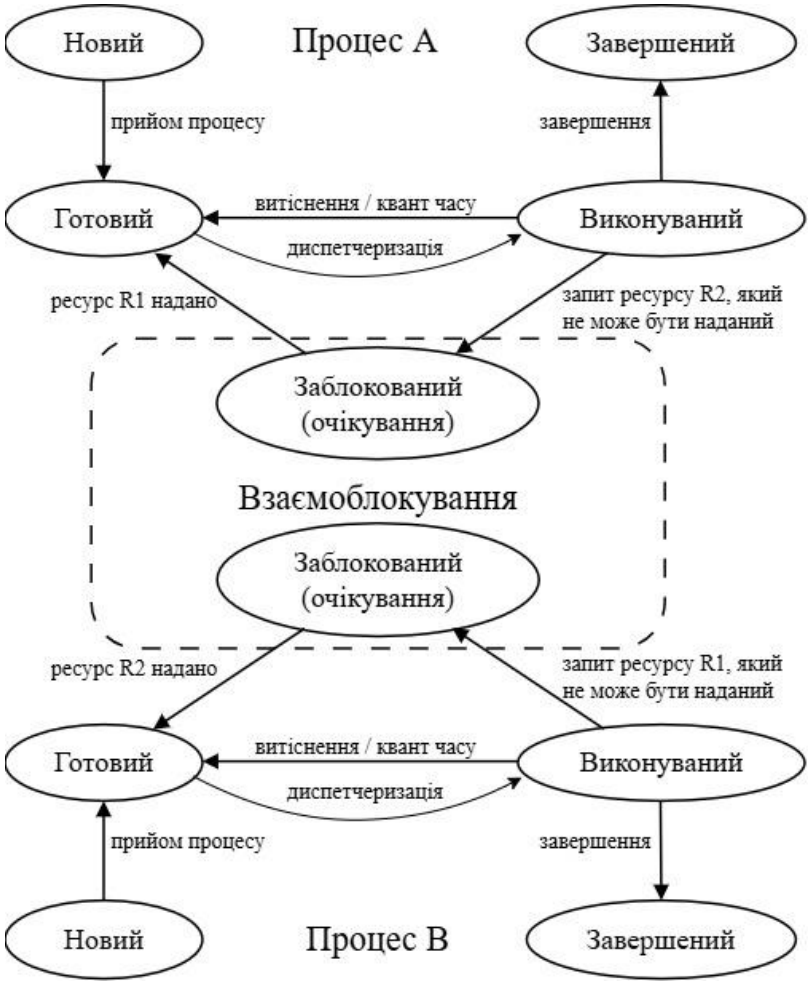


Рисунок 2.2 – Місце взаємоблокування процесів у п’ятистановій моделі

У класичному підході (рис. 2.3) збережено ланцюг "процеси – ядро ОС – системні ресурси". На основі системних даних формується поточна сукупність сигнатур процесів і стан комп’ютерної системи. Запобігання й уникнення реалізуються через наперед задані обмеження або перевірку безпечності стану до виділення ресурсу. Якщо ці засоби не охопили конкретний сценарій, то графи ресурсної залежності аналізуються після виникнення очікувань, а керуючий вплив застосовується лише після виявлення циклу і має реактивний характер, що стосується завершення процесу, відкату, перезапуску або примусового звільнення ресурсу.



Рисунок 2.3 – Інформаційні потоки класичного підходу, доповненого системою прогнозування стану взаємоблокування процесів

Застосуємо до тих самих системних даних прогнозний підхід. Спочатку формуються сигнатури процесів і поточний стан комп’ютерної системи. На етапі навчання історичні сигнатури пов’язуються з фактичними відомостями про

подальше виникнення або відсутність взаємоблокування. На цій основі формується навчальна вибірка, виконується нечітка кластеризація, оцінюється небезпечність типових ресурсних станів і будується база нечітких правил. На робочому етапі поточна сигнатура процесу, що очікує ресурс, опрацьовується сформованою базою правил, після чого визначається рівень ризику та перелік процесів-кандидатів для превентивного впливу.

Навчальний приклад вважається небезпечним у тому випадку, коли на момент спостереження взаємоблокування ще не зафіксовано, але воно виникає протягом наступного інтервалу спостереження.

Таке подання пов’язує поточну ресурсну конфігурацію з її майбутнім наслідком. Завдяки цьому база нечітких правил може бути застосована до поточної сигнатури до фактичної фіксації взаємоблокування. Отриманий результат використовується для посиленого моніторингу, відкладення некритичного запиту або зміни черговості доступу без глобального обмеження паралельного виконання інших процесів.

Архітектуру системи зображено на рисунку 2.4. Вона включає ресурси комп’ютерної системи, ядро ОС та запропоновану систему, у межах якої виділено контури навчання й оперативного прогнозування. Таке подання зберігає штатну взаємодію процесів із планувальником і менеджером ресурсів, а аналітичні функції реалізує як окрему надбудову.

Побудова архітектури ґрунтується на принципах модульності, часової узгодженості даних і розмежування обчислювальних режимів. Модульність забезпечує можливість заміни засобів збирання системних подій або способу кодування ресурсів без зміни загальної процедури прогнозування. Часова узгодженість передбачає прив’язування відомостей про стан процесу, утримувані ресурси та новий запит до одного моменту спостереження. Розмежування режимів дає змогу виконувати кластеризацію та побудову правил періодично, тоді як у процесі штатної роботи застосовується вже сформована база правил.

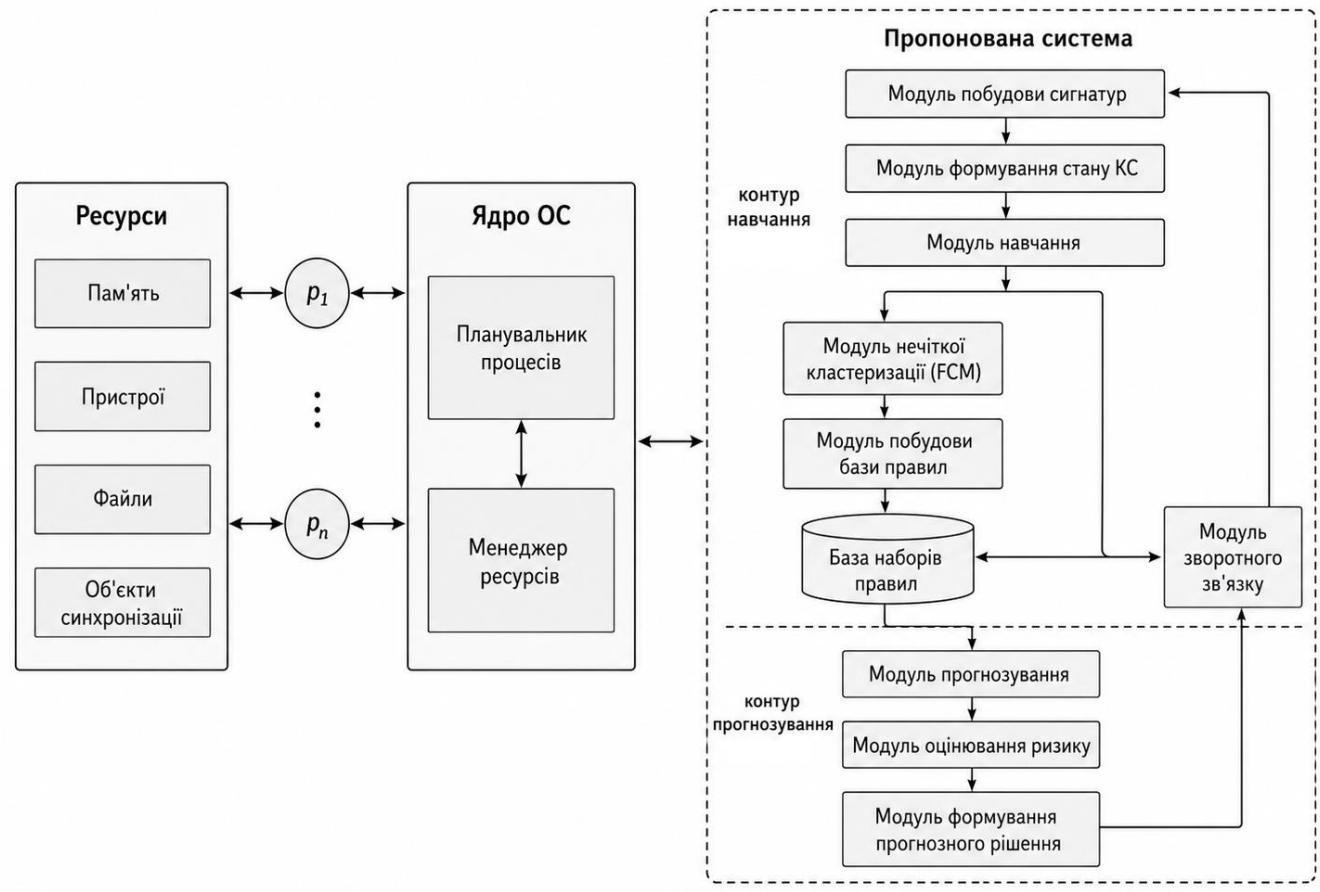


Рисунок 2.4 – Архітектура системи прогнозування взаємоблокування процесів

Першим функціональним компонентом системи є модуль побудови сигнатур, що формує числове відображення процесу ОС в сигнатуру процесу в системі прогнозування.

Модуль формування стану КС об’єднує сигнатури активних процесів.

Етап навчання виконується до штатного функціонування КС. Модуль навчання формує матрицю ознак фіксованої розмірності та виконує нечітку кластеризацію Fuzzy C-Means. Застосування нечіткого підходу зумовлене відсутністю чіткої межі між штатним очікуванням і станом, що безпосередньо передує взаємоблокуванню. Кожний приклад характеризується ступенями належності до кількох кластерів, що дає змогу відобразити поступове зростання небезпеки.

Етап навчання реалізує послідовність взаємопов’язаних операцій: формування вибірки, визначення параметрів FCM, оцінювання якості розбиття,

змістову інтерпретацію кластерів, побудова бази правил. Кількість кластерів задається відповідно до прийнятої системи станів, а параметри нечіткості та критерій зупинки уточнюються в моделі навчання.

На етапі прогнозування формується прогнозне рішення яке розглядається як кількісно обґрунтована оцінка майбутнього взаємоблокування, що в подальшому використовується для прийняття рішення про подальше виконання процесу.

Функціональне призначення основних компонентів узагальнено в таблиці 2.1.

Таблиця 2.1 – Функціональні компоненти системи прогнозування взаємоблокування процесів

Компонент	Вхідні дані	Основна функція	Результат
Модуль побудови сигнатур	Системні події, параметри процесів і ресурсів	Формування узгодженого вектору ознак	Сигнатура процесу
Модуль формування стану КС	Сигнатури активних процесів	Об’єднання сигнатур у часовий зріз	Стан комп’ютерної системи
Модуль навчання та FCM	Історичні стани і прогнозні маркери	Формування вибірки та нечітка кластеризація	Параметри кластерів
Модуль побудови бази правил	Центри кластерів і ступені належності	Інтерпретація кластерів та формування правил	База нечітких правил
Модуль прогнозування	Поточна сигнатура, стан КС і база правил	Нечіткий логічний висновок	Оцінка ризику
Модуль формування рішення	Оцінка ризику і параметри процесу	Визначення класу стану та попередження	Прогнозне рішення

Узагальнений інформаційний потік має наступний вигляд:

1) для навчання:

системні події → сигнатури процесів → стан КС → навчальна вибірка → нечіткі кластери → база правил → оцінка ризику;

2) для роботи:

системні події → сигнатури процесів → стан КС → прогнозне рішення.

Практична реалізація архітектури передбачає розмежування інваріантної та платформозалежної частин. До платформозалежної частини належать засоби отримання відомостей від конкретної операційної системи та перетворення її внутрішніх ідентифікаторів. Формат сигнатури, моделі стану, процедури навчання і нечіткого висновку залишаються інваріантними. Це створює передумови для перенесення системи між програмними платформами за умови розроблення відповідного модуля збирання даних.

На рисунку 2.4 запропонований підхід розміщено безпосередньо у класичній схемі: він отримує дані від ядра ОС, але не замінює планувальник, менеджер ресурсів. Класична схема має послідовність "системні події – граф ресурсних залежностей – виявлення циклу – відновлення" і формує результат у момент t_d, коли взаємоблокування вже виникло. Запропонована архітектура не замінює цей підхід, а доповнює його аналітичною надбудовою. Для правильно прогнозованого випадку попередження формується в момент $t_p < t_d$, а інтервал випередження становить $\Delta t_p = t_d - t_p > 0$. Отже, перевага за критерієм своєчасності випливає з наявності прогнозного горизонту Δt_p

Задамо згідно концепції з рисунку 2.4 архітектуру системи прогнозування взаємоблокування в комп'ютерних системах так:

$$M_{p,KS} = \langle M_S, M_{OS,pr}, M_{OS,pr,p}, M_R, M_{r,OS}, M_{act}, G_{p,KS} \rangle, \quad (2.1)$$

де M_S – множина елементів системи прогнозування стану взаємоблокування процесів в КС; $M_{OS,pr}$ – множина процесів в ОС в поточний момент часу, тобто в часовий інтервал дослідження; $M_{OS,pr,p}$ – множина процесів в ОС в поточний момент часу, тобто в часовий інтервал дослідження, для яких здійснюється прогнозування щодо потенційного взаємоблокування; M_R – множина ресурсів КС, які можуть запитувати процеси для свого виконання; $M_{r,OS}$ – множина поточного розподілу ресурсів КС між процесами; M_{act} – множина функцій реакції системи

прогнозування; $G_{p,KS}$ – граф зв'язків між множинами $M_S, M_{OS,pr}, M_{OS,pr,p}, M_R, M_{r,OS}, M_{act}$

Задамо множину елементів системи прогнозування стану взаємоблокування процесів в КС $M_S = \{M_{S_1}, \dots, M_{S_{10}}\}$, де M_{S_1} – модуль побудови сигнатур, M_{S_2} – модуль формування стану КС, M_{S_3} – модуль навчання, M_{S_4} – модуль нечіткої кластеризації, M_{S_5} – модуль побудови бази правил, M_{S_6} – база правил, M_{S_7} – модуль прогнозування, M_{S_8} – модуль оцінювання ризику, M_{S_9} – модуль формуванняпрогнозного рішення, $M_{S_{10}}$ – модуль зворотного зв'язку.

Задамо множину процесів в ОС $M_{OS,pr} = \{M_{OS,pr_i}\}, i = \overline{1, n_{OS,pr}}$, де $n_{OS,pr}$ – кількість процесів в ОС.

Задамо множину процесів в ОС, для яких здійснюється прогнозування, $M_{OS,pr,p} = \{M_{OS,pr,p_i}\}, i = \overline{1, n_{OS,pr,p}}$, де $n_{OS,pr,p}$ – кількість процесів в ОС, для яких здійснюється прогнозування, $M_{OS,pr,p} \subset M_{OS,pr}$.

Задамо множину ресурсів КС $M_R = \{M_{R_i}\}, i = \overline{1, n_R}$, де n_R – кількість ресурсів в КС. Визначимо M_{R_1} – обсяг оперативної пам'яті, M_{R_2} – процесори, M_{R_3} – пристрої введення-виведення, M_{R_4} – обсяг зовнішньої пам'яті, M_{R_5} – дескриптори файлів, M_{R_6} – семафори, M_{R_7} – м'ютекси, M_{R_7} – записи БД і т.д.

Задамо множину поточного розподілу ресурсів КС між процесами $M_{r,OS} = \{M_{r,OS_i}\}, i = \overline{1, n_{r,OS}}$, $n_{r,OS}$ – кількість пар процес/ресурс в КС, $M_{r,OS_i} = \left(M_{OS,pr_j}, M_{R_k}\right)$, M_{OS,pr_j} – j-й процес в ОС, що використовує M_{R_k} – k-й ресурс КС.

Задамо множину функцій реакції системи прогнозування $M_{act} = \{M_{act_1}, M_{act_2}\}$, де M_{act_1} – продовження роботи процесу, M_{act_2} – перезапуск процесу.

На графі зв'язків між множинами з формули (2.1) (рис. 2.5) відображено структурні та функціональні відношення між множинами архітектури системи прогнозування взаємоблокування.

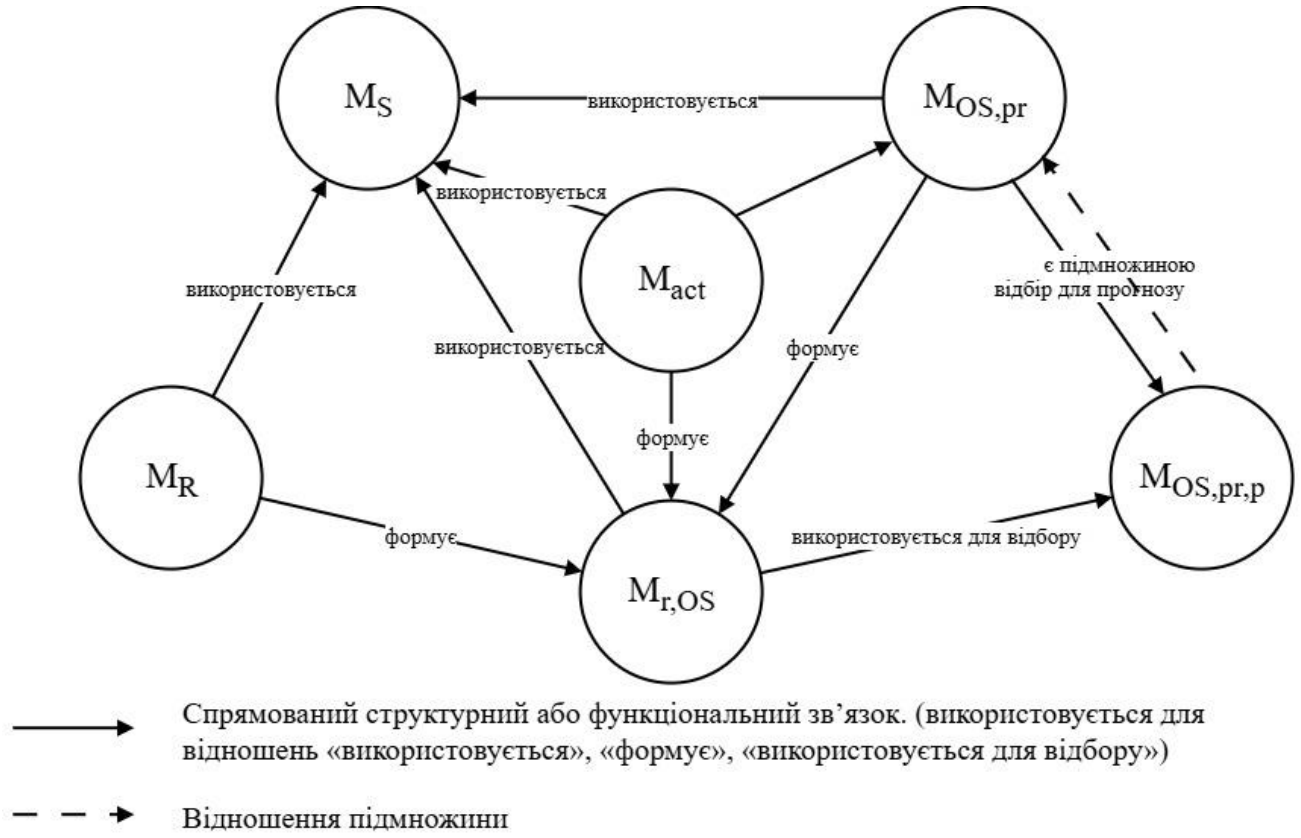


Рисунок 2.5 – Граф зв'язків між множинами

Множина ресурсів комп'ютерної системи, множина процесів операційної системи, множина поточного розподілу ресурсів між процесами та множина функцій реакції системи прогнозування входять до складу множини елементів системи прогнозування. Множина процесів, для яких виконується прогнозування, є підмножиною множини процесів операційної системи. Множина поточного розподілу ресурсів формується на основі взаємодії множини ресурсів та множини процесів ОС. У свою чергу, множина процесів для прогнозування виділяється з множини процесів ОС з урахуванням поточного розподілу ресурсів між процесами.

Задамо процес прогнозування згідно формули (2.1) так:

$$M_{prog} = \langle M_{OS,pr,p}, M_R, M_{r,OS}, M_1, M_2, M_3, P \rangle \quad (2.2)$$

де $M_{OS,pr,p}$ – множина процесів в ОС в поточний момент часу, тобто в часовий інтервал дослідження, для яких здійснюється прогнозування щодо потенційного взаємоблокування; M_R – множина ресурсів КС, які можуть запитувати процеси для

свого виконання; $M_{r,OS}$ – множина поточного розподілу ресурсів КС між процесами; M_1 – метод побудови сигнатури процесу; M_2 – метод побудови бази нечітких правил для прогнозування взаємоблокування процесів; M_3 – метод прогнозування взаємоблокування процесів; P – прогноз взаємоблокування процесів.

Визначимо прогноз взаємоблокування процесів так:

$$P: \left(M_{OS,pr,p}, M_R, M_{r,OS}\right) \xrightarrow{F} [0; 1] \tag{2.3}$$

де вхідними параметрами для здійснення прогнозу є $M_{OS,pr,p}$ – множина процесів в ОС в поточний момент часу, тобто в часовий інтервал дослідження, для яких здійснюється прогнозування щодо потенційного взаємоблокування; M_R – множина ресурсів КС, які можуть запитувати процеси для свого виконання; $M_{r,OS}$ – множина поточного розподілу ресурсів КС між процесами; F – функція пронозу.

Результатом прогнозу є ступінь належності до граничного або взаємозаблокованого стану.

Визначимо функцію прогнозу так:

$$F = F_{M_3}\left(F_{M_2}(F_{M_1}\left(M_{OS,pr}\right), M_R, M_{r,OS}), F_{M_1}(M_{OS,pr,p})\right) \rightarrow [0; 1] \tag{2.4}$$

де $M_{OS,pr}$ – множина процесів в ОС в поточний момент часу, тобто в часовий інтервал дослідження; M_R – множина ресурсів КС, які можуть запитувати процеси для свого виконання; $M_{r,OS}$ – множина поточного розподілу ресурсів КС між процесами; $M_{OS,pr,p}$ – множина процесів в ОС в поточний момент часу, тобто в часовий інтервал дослідження, для яких здійснюється прогнозування щодо потенційного взаємоблокування; F_{M_1} – функція методу побудови сигнатури процесу; F_{M_2} – функція методу побудови бази нечітких правил для погнозування взаємоблокувань процесів; F_{M_3} – функція методу прогнозування стану взаємоблокування процесів в КС;

Таким чином, запропонована архітектура поєднує штатні механізми ядра ОС із засобами формалізованого подання ресурсної поведінки, нечіткого навчання та

оперативного прогнозування. Розділення навчання і прогнозування дає змогу винести ресурсомісткі операції за межі обробки поточних запитів.

2.2 Модель сигнатури процесу в комп'ютерних системах

Визначимо параметри процесу, що є необхідними для побудови його сигнатури. Для цього розглянемо множину характеристик процесу, якими він володіє, у сучасних операційних системах. Операційна система створює процес, коли користувач запускає програму на виконання.

Визначимо процес і програму як дві сутності. Програма виконується в даному процесі. Програмою є машинний код і початкові дані, що містяться у виконуваному файлі на диску або скопійовані ОС для виконання в оперативну пам'ять. Для кожного процесу ОС створює додатковий набір даних, що називається середовищем виконання процесу або контекстом процесу. Дані із цього набору задамо атрибутами процесу, тобто ознаками, що відрізняють процеси між собою. Вони можуть бути як сталими, так і змінюватись протягом життєвого циклу процесу.

Для вирішення задачі прогнозування поведінки процесу представимо його набором характеристик (сигнатурою) [118; 121].

Визначення 1. Під сигнатурою процесу задамо сукупність його характеристик, яка однозначно ідентифікує поведінку процесу у комп’ютерній системі, а саме: ідентифікатор процесу, ідентифікатор батьківського процесу, ідентифікатор користувача якому належить процес, пріоритет процесу, квоти процесу (кількість пам'яті і процесорний час доступні процесу), лічильники операцій введення-виведення, дескриптори відкритих процесом файлів, лічильник зупинок процесу.

Подані вище формальні вирази задають каркас моделі, а описова частина конкретизує, як ці вирази інтерпретуються під час аналізу поведінки процесів у комп’ютерних системах. Процес розглядається в цьому контексті не лише як

одиниця планування операційної системи, а як об’єкт, який має власні ідентифікаційні ознаки, поточний ресурсний стан і запит до розподілюваного ресурсу.

Сигнатуру процесу використаємо для компактного представлення стану процесу в момент спостереження. У ній поєднано ознаки, що описують походження процесу, належність користувачу, поточні параметри виконання та ресурсну взаємодію. Таке поєднання є необхідним, оскільки взаємоблокування виникає не через окремий параметр, а через комбінацію зайнятих ресурсів, запиту до нового ресурсу та одночасної поведінки інших процесів. На рівні операційної системи однакові за назвою програми можуть мати різну поведінку.

Наприклад, два процеси, які створені одним виконуваним файлом, можуть належати різним користувачам, мати різні пріоритети, різні відкриті дескриптори та різні множини зайнятих ресурсів. Тому використання тільки ідентифікатора процесу або тільки імені програми не дозволяє отримати достатній опис для прогнозування небезпечного стану. У запропонованій моделі важливим є поділ ознак на ідентифікаційну та ресурсну складові.

Ідентифікаційна складова відображає те, що дозволяє віднести процес до певного класу виконання, а ресурсна складова показує, з якими елементами системи він уже взаємодіє. Саме ресурсна частина надалі використовується для побудови навчальної вибірки, кластеризації та формування правил прогнозування. Опис процесу в моделі не зводиться до фіксації його стану в ізольований момент. Кожна сигнатура має сенс у межах часової послідовності станів системи, оскільки взаємоблокування є динамічним явищем.

Для прогнозування важливо бачити не тільки те, що ресурс зараз зайнятий, але й те, як змінюється конфігурація процесів протягом наступного інтервалу спостереження.

З цієї причини в моделі використовується навчальна пара, де поточна сигнатура зв’язується з прогнозним маркером. Маркер показує, чи призводить зафіксований стан до взаємоблокування протягом заданого інтервалу часу. У такий

спосіб поточна поведінка процесу пов’язується з майбутнім наслідком, а не лише з уже зафіксованим фактом взаємоблокування [116-118; 122].

Перевагою такого підходу є можливість формувати правила на основі історії роботи системи. Якщо певні комбінації зайнятих і запитуваних ресурсів неодноразово передували взаємоблокуванню, то вони повинні мати більший вплив на оцінку ризику. Якщо ж подібні комбінації часто зустрічались у нормальному режимі та не призводили до блокування, відповідні кластери мають отримувати менше значення ризику.

Представимо ресурси КС з множини M_R (формула (2.1)) у вигляді множини ідентифікаторів ресурсів

$$R_e = \{r_i\}, i = \overline{1, n_r}, \tag{2.5}$$

де n_r – кількість ідентифікаторів ресурсів.

Подамо сигнатуру процесу наступною моделлю:

$$a(t) = (a_1^t, \dots, a_k^t, a_{k+1}^t, \dots, a_{z-1}^t, a_z^t, a_{z+1}^t) \tag{2.6}$$

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;

$$a_z^t = \begin{cases} id \text{ ресурса, якщо процес потребує додаткових ресурсів} \\ 0, \text{ якщо процес не потребує додаткових ресурсів} \end{cases} \tag{2.7}$$

$$\forall i \in \{k+1, z\}: a_i^t \in R_e$$

$$a_{z+1}^t = \begin{cases} 1, \text{якщо процес знаходиться в стані взаємоблокування} \\ 0, \text{якщо процес не знаходиться в стані взаємоблокування} \end{cases} \tag{2.8}$$

Параметри з індексами від 1 до k ідентифікують процес у системі, а параметри ресурсної частини описують ресурси, з якими процес взаємодіє.

У такому поданні одне значення відповідає номеру запитуваного ресурсу, а група значень описує ресурси, вже виділені процесу. Це забезпечує можливість аналізувати не лише поточне використання ресурсів, але й потенційний конфлікт між запитом процесу та ресурсами, зайнятими іншими процесами.

Формальні параметри сигнатури поставимо у відповідність реальним характеристикам процесу, які можуть бути отримані із структур керування процесами операційної системи в момент спостереження t. Відповідність між формальними позначеннями та реальними атрибутами процесу задамо так:

1) a_1^t – PID ідентифікатор процесу, який однозначно визначає процес у межах поточного сеансу функціонування операційної системи;

2) a_2^t – PPID ідентифікатор батьківського процесу, що характеризує походження процесу та його зв'язок з іншими процесами;

3) a_3^t – UID ідентифікатор користувача та групи, до якої належить користувач;

4) a_4^t – ознака системного процесу;

5) a_5^t – час перебування процесу в системі від моменту його запуску;

6) a_6^t – поточний пріоритет процесу, що впливає на порядок надання йому процесорного часу;

7) a_7^t – поточний пріоритет процесу, що впливає на порядок надання йому процесорного часу;

В залежності від конкретної ОС, ідентифікаційна частина сигнатури може бути доповнена додатковими атрибутами.

Сигнатура процесу є основним елементом подальшої обробки, оскільки саме вона перетворює різнорідні дані операційної системи в єдину структуру. До цієї структури можуть входити числові параметри, ідентифікатори, лічильники та ознаки, що відображають ресурсні зв'язки. Для прогнозування важливо, щоб сигнатури різних процесів можна було порівнювати між собою в одному просторі ознак. PID використовується для однозначного зв'язування поточної сигнатури з конкретним процесом у межах одного моменту спостереження. Однак PID не слід розглядати як ознаку, що сама по собі визначає небезпечність процесу. Його призначення полягає в ідентифікації об'єкта спостереження та зв'язуванні послідовних станів одного процесу в різні моменти часу.

PPID дозволяє враховувати походження процесу. Для деяких класів задач

дочірні процеси повторюють ресурсну поведінку батьківського процесу або успадковують відкриті дескриптори. Це може бути важливим для виявлення груп процесів, які одночасно претендують на однакові ресурси. Така інформація не замінює ресурсні ознаки, але допомагає краще інтерпретувати отримані кластери.

UID та пов’язані з ним ідентифікатори користувача або групи характеризують контекст доступу. Процеси різних користувачів можуть мати різні обмеження, різні квоти, різні набори дозволених операцій і різну інтенсивність звернення до файлових або синхронізаційних ресурсів. Тому ознаки належності процесу до користувача або групи є корисними для уточнення профілю його поведінки.

Пріоритет процесу відображає вплив планувальника на порядок отримання процесорного часу. Хоча пріоритет не є прямим ресурсним конфліктом, він впливає на те, як довго процес очікує виконання та як швидко він може звільнити зайняті ресурси.

У станах підвищеного навантаження це може посилювати ризик накопичення процесів, що очікують один на одного. Квоти процесу визначають межі використання обчислювальних ресурсів. Якщо процес наближається до встановлених обмежень, то його поведінка може змінюватися. Операції введення-виведення сповільнюються, виділення пам’яті затримується, а системні виклики можуть виконуватися довше. Такі зміни не завжди є взаємоблокуванням, але вони створюють умови, в яких конфлікти за ресурси стають більш ймовірними.

Лічильники операцій введення-виведення відображають інтенсивність обміну даними між процесом і пристроями або файловою підсистемою. Висока активність введення-виведення може бути нормальною для службових процесів, але в поєднанні з очікуванням критичного ресурсу вона може свідчити про ризик затримки інших процесів. Тому такі лічильники важливо розглядати разом з ресурсними ознаками.

Дескриптори відкритих файлів і об’єктів синхронізації мають безпосередній зв’язок із можливістю взаємоблокування. Якщо процес утримує дескриптор, який

потрібен іншому процесу, і одночасно очікує ресурс, зайнятий цим іншим процесом, виникає передумова циклічного очікування. Саме такі ситуації має виявляти модель до того, як вони перейдуть у стабільний стан взаємоблокування. Лічильник зупинок процесу або інші ознаки переривання виконання показують, наскільки часто процес переходить у стан очікування.

Якщо такі переходи пов’язані з ресурсними запитами, то вони можуть бути корисними для відокремлення нормального очікування від небезпечного. У моделі ці ознаки не замінюють маркер взаємоблокування, але уточнюють поведінковий профіль процесу.

Загалом сигнатура має бути достатньо повною, щоб відображати ресурсну взаємодію, але не надмірною. Якщо включити до неї всі доступні атрибути операційних систем, то простір ознак стане надто великим і кластеризація втратить інтерпретованість. Тому доцільно залишати ті параметри, що мають пояснюваний зв’язок із запитом, утриманням або звільненням ресурсів [126].

Ресурсна складова є ключовою для прогнозування взаємоблокування, оскільки саме вона відображає взаємодію процесу з розподілюваними об’єктами системи. До таких об’єктів можуть належати області пам’яті, файлові дескриптори, пристрої введення-виведення, блокування, семафори, м’ютекси, сокети та інші елементи, які не можуть одночасно використовуватися довільною кількістю процесів. Для моделі не обов’язково зберігати повний опис фізичної природи ресурсу. Достатньо мати унікальний ідентифікатор, який дозволяє порівнювати факти зайняття та очікування. Це спрощує подальше перетворення даних у вектор ознак і дозволяє однаково обробляти ресурси різних типів. Наприклад, файл, м’ютекс і пристрій можуть кодуватися за одним принципом, хоча на рівні операційної системи вони мають різну природу.

Окреме кодування зайнятих ресурсів і ресурсу, який очікується, є принципово важливим. Якщо процес просто утримує ресурс, це ще не означає, що він перебуває в небезпечному стані. Ризик виникає тоді, коли процес утримує певні ресурси та одночасно чекає на інший ресурс.

Саме комбінація утримання й очікування створює основу для циклічного очікування. У реальній системі кількість ресурсів, зайнятих різними процесами, може істотно відрізнятися. Один процес може утримувати лише один дескриптор, інший – десятки або сотні. Без перетворення в фіксований простір ознак такі сигнатури було б складно подавати до FCM, оскільки кластеризація потребує матриці з однаковою кількістю стовпців для всіх рядків [111-112].

Бінарне представлення ресурсів усуває цю проблему. Для кожного ресурсу задається окрема позиція у векторі, а значення позиції показує, чи пов’язаний процес із відповідним ресурсом. Завдяки цьому процеси з різною кількістю зайнятих ресурсів стають сумісними для математичної обробки. Розмірність вектора визначається множиною можливих ресурсів, а не кількістю ресурсів у конкретного процесу [125-126].

Запитуваний ресурс доцільно кодувати окремою групою ознак, а не змішувати з уже зайнятими ресурсами. Це дозволяє відрізнити ситуацію, коли процес володіє ресурсом, від ситуації, коли він його очікує. Для взаємоблокування ці стани мають різний зміст: утримання ресурсу створює залежність для інших процесів, а очікування ресурсу формує залежність самого процесу. Якщо процес не очікує жодного ресурсу, ресурсний запит кодується нульовим значенням або нульовим блоком ознак.

Така сигнатура може залишатися в навчальній вибірці, оскільки вона показує нормальну поведінку процесу. Однак під час оперативного прогнозування основна увага зосереджується на тих процесах, для яких уже зафіксовано факт ненадання запитуваного ресурсу.

Ресурсна модель також дає змогу враховувати ситуації, коли різні ресурси мають подібну роль у системі. Наприклад, кілька м’ютексів можуть захищати різні структури даних, але для алгоритму вони залишаються окремими ідентифікаторами. Це важливо, оскільки взаємоблокування часто виникає не через тип ресурсу, а через конкретний порядок його захоплення різними процесами.

Розроблено модель сигнатури процесу, яка забезпечує уніфіковане подання

ідентифікаційних, поведінкових і ресурсних характеристик окремого процесу. Її застосування дозволило привести різнорідні відомості про процеси та ресурси до єдиної структури, придатної для подальшого формування навчальної вибірки, виконання нечіткої кластеризації й оцінювання ризику взаємоблокування.

2.3 Модель стану комп'ютерних систем

Для прогнозування стану взаємоблокування недостатньо розглядати ресурсну поведінку кожного процесу ізольовано, оскільки небезпечна ситуація формується внаслідок взаємодії сукупності процесів і ресурсів. Тому необхідно перейти від сигнатури окремого процесу до формалізованого подання стану всієї комп'ютерної системи в певний момент часу. Таке подання має об'єднувати сигнатури активних процесів і відображати зміни їхнього складу, станів та ресурсних залежностей упродовж визначеного інтервалу спостереження. Це дасть змогу описати функціонування КС як послідовність взаємопов'язаних станів та виявляти тенденції формування небезпечних ресурсних ситуацій.

В КС одночасно функціонує множина процесів $M_{OS,pr}$ (2.1), тому відповідно множину сигнатур цих процесів у момент часу t задамо так:

$$A_t = \{a(t)_i\}, i = \overline{1, n_p}, \tag{2.9}$$

де $a(t)$ – сигнатура процесу; n_p – кількість процесів у КС.

Стан комп'ютерної системи в момент часу t визначено множиною сигнатур активних процесів. Стан системи розглядаємо як зріз взаємодії процесів з ресурсами в конкретний момент спостереження.

З метою представлення КС в момент часу t задамо її стан вектором w_t:

$$w_t = (A_t, \mathrm{t}, \mathrm{R}_e) \tag{2.10}$$

де A_t – множина сигнатур процесів КС в момент часу t; t – час; R_e – множина ідентифікаторів ресурсів КС.

Для опису роботи КС представимо множину її станів так:

$$W = \{\mathrm{w_t} | t_s \leq \mathrm{t} \leq t_e\}, \quad (2.11)$$

де w_t – стан КС в момент часу t; t_s – час початку спостереження; t_e – час завершення спостереження.

Кількість процесів у системі не є сталою. Процеси створюються, завершуються, переходять у стан очікування, змінюють пріоритети та звільняють ресурси. Через це модель стану повинна допускати змінну потужність множини сигнатур. Для формування навчальної вибірки це означає, що різні моменти часу можуть давати різну кількість навчальних прикладів.

Часовий параметр використовується для впорядкування спостережень. Без часу неможливо визначити, чи є певна сигнатура попередником взаємоблокування. Саме послідовність станів дозволяє встановити, що поточна конфігурація процесів через певний інтервал переходить у небезпечний стан.

Тому час у моделі є не допоміжною, а зв’язувальною ознакою. Інтервал спостереження має бути достатньо малим, щоб не пропустити швидкі переходи процесів, але достатньо великим, щоб не створювати надмірний обсяг даних. Якщо інтервал надто великий, короткочасні конфлікти можуть залишитися непоміченими. Якщо він дуже малий, то навчальна вибірка може містити багато майже однакових станів, що ускладнює кластеризацію.

Множина станів за період спостереження утворює основу для побудови навчальної вибірки. Кожний стан додає набір сигнатур, а кожна сигнатура надалі може отримати прогнозний маркер.

Множина станів W, задана виразом за формулою (2.7), є початковою часово впорядкованою сукупністю спостережень за роботою КС. Вона містить стани системи, сформовані в різні моменти часу, але безпосередньо ще не є навчальною вибіркою, оскільки сигнатури процесів у її складі не пов’язані з відомостями про подальший результат їх ресурсної взаємодії.

Для перетворення накопичених станів у навчальну вибірку необхідно встановити результат подальшого розвитку кожної зафіксованої ресурсної ситуації. Для цього після формування сигнатури процесу протягом заданого

часового інтервалу відстежується, чи відбувся його перехід у стан взаємоблокування. Якщо взаємоблокування виникло, то відповідний запис розглядається як приклад небезпечної ресурсної поведінки. Якщо процес продовжив виконання або отримав необхідний ресурс без утворення взаємоблокування, запис належить до прикладів безпечної поведінки.

Отже, навчальна вибірка утворюється з накопичених сигнатур процесів, для яких відомий фактичний результат подальшої ресурсної взаємодії. Такий підхід забезпечує зв’язок між параметрами процесу, його поточним ресурсним станом і реально зафіксованим переходом або не переходом у стан взаємоблокування. При цьому відомості про результат використовуються не як одна з ознак процесу, а як засіб подальшого оцінювання небезпечності типових станів, виявлених під час опрацювання навчальних даних.

Таким чином, результатом формування навчальної вибірки є впорядкована сукупність прикладів безпечної та небезпечної ресурсної поведінки процесів. Вона створює інформаційну основу для виявлення типових станів КС, визначення ступеня їх небезпечності та подальшого формування правил, за допомогою яких здійснюватиметься оцінювання поточних ресурсних ситуацій.

Навчальна вибірка формується шляхом систематичного проходження по часових зрізах роботи комп’ютерних систем. Модель стану дозволяє аналізувати не тільки окремий процес, але й контекст його виконання. Процес може виглядати безпечним у ізольованому розгляді, але ставати небезпечним у поєднанні з іншими процесами, які утримують потрібні йому ресурси [124].

У практичній реалізації стан формуватимемо на основі журналів операційної системи, системних викликів, таблиць процесів, подій захоплення і звільнення ресурсів, а також подій очікування. Джерело даних може бути різним, але для моделі важливо, щоб після збору всі ці дані були зведені до узгоджених сигнатур однакової структури. Подання станів у вигляді часової множини також дозволяє відокремити навчальний етап від етапу прогнозування.

На навчальному етапі система аналізує історичні стани та встановлює зв’язок

між сигнатурами і майбутніми взаємоблокуваннями. На етапі прогнозування використовується поточний стан, для якого вже не відомий майбутній результат, але доступна побудована база правил.

Таким чином, розроблена модель забезпечує формалізоване подання поточного та історичних станів комп'ютерної системи як часово впорядкованої сукупності сигнатур процесів. Це дає змогу враховувати не лише характеристики окремого процесу, а й загальний контекст його ресурсної взаємодії з іншими процесами, простежувати розвиток ресурсних залежностей та формувати навчальну вибірку з прикладів безпечної і небезпечної поведінки.

Водночас самої наявності навчальної вибірки недостатньо для оцінювання поточного стану системи. Необхідно визначити закономірності, за якими поєднання значень ознак процесу та його ресурсного оточення характеризуватиме різний ступінь наближення до взаємоблокування. Через динамічність станів КС та відсутність чітких меж між безпечними, граничними й небезпечними ситуаціями такі закономірності доцільно подати у формі нечітких правил.

Нечіткі правила повинні встановлювати зв'язок між ознаками ресурсної поведінки процесів і відповідним рівнем ризику виникнення взаємоблокування. Їх застосування дасть змогу опрацьовувати поточні сигнатури процесів, ураховувати часткову належність спостережуваного стану до декількох типових ситуацій та формувати узагальнену оцінку його небезпечності. Отже, наступним етапом дослідження є розроблення моделі процесу побудови бази нечітких правил на основі навчальної вибірки, сформованої з використанням запропонованої моделі стану комп'ютерної системи.

2.4 Модель процесу побудови бази нечітких правил

Розроблені моделі сигнатури процесу та стану КС утворили формалізовану основу для фіксації ресурсної взаємодії в послідовні моменти часу. Проте самі по собі ці моделі не визначають, які поєднання ознак свідчать про наближення процесів до взаємоблокування. Для встановлення таких закономірностей

накопичені спостереження необхідно пов’язати з їх подальшими наслідками та виділити типові сценарії ресурсної поведінки. Відсутність чіткої межі між допустимим очікуванням ресурсу та небезпечним станом зумовила застосування нечіткого групування навчальних даних. Отримані групи мають бути інтерпретовані за рівнем небезпеки та перетворені на правила, придатні для подальшого прогнозування.

Побудова навчальної вибірки проводиться під час спостереження за роботою КС. Для кожного часового зрізу фіксуються ресурсні характеристики процесів, а після завершення заданого інтервалу перевіряється фактичний результат розвитку відповідної ситуації. Зіставлення цих відомостей утворює навчальну пару, першим елементом якої є опис поточної ресурсної конфігурації, а другим – маркер подальшого виникнення або відсутності взаємоблокування. Потреба в такому поданні зумовлена тим, що схожі стани ресурсного очікування можуть мати різні наслідки та не можуть бути однозначно оцінені лише за поточними ознаками. Навчальна пара зберігає часовий зв’язок між спостережуваним станом і фактично встановленим результатом його розвитку. Ознакова частина пари надалі використовується під час нечіткої кластеризації, тоді як прогнозний маркер дає змогу визначити рівень небезпечності отриманих кластерів. Отже, навчальні пари формуються автоматично шляхом узгодження послідовних станів системи із зареєстрованими наслідками та утворюють вихідні елементи навчальної вибірки

Визначимо ресурсну сигнатуру процесу $a^*(t)$ множиною ідентифікаторів зайнятих ресурсів процесу та ідентифіктора запитуваного ресурсу процесу в момент часу t так:

$$a^*(t) = (a_{k+1}^t, \dots, a_{z-1}^t, a_z^t), \tag{2.12}$$

де $a_{k+1}^t, \dots, a_{z-1}^t$ – ідентифікатори ресурсів, які процес уже займає; a_z^t – ідентифікатор запитуваного ресурсу.

Сформуємо прогнозний маркер $y(a)^t$, який показує, чи перейде процес a у взаємоблокування протягом наступного інтервалу часу Δt. Визначимо [116-118] його так:

$$y(a)^t = \begin{cases} 1, & \exists \tau \in [t, t + \Delta t]: a_{z+1}^{\tau} = 1, \\ 0, & \text{інакше.} \end{cases} \quad (2.13)$$

Відповідно поточна сигнатура маркується як гранична, якщо для цього самого процесу протягом найближчого інтервалу Δt буде зафіксовано взаємоблокування.

З метою підготовки до кластеризації для кожної сигнатури сформуємо навчальну пару:

$$(a^*(t), y(a)^t). \quad (2.14)$$

де $a^*(t)$ – ресурсна сигнатура процесу, $y(a)^t$ – прогнозний маркер

У моделі розрізняються інтервал формування стану Δt_s та прогнозний горизонт Δt_p. Перший задає частоту оновлення сигнатур і визначає деталізацію спостереження, другий – часову межу, в межах якої перевіряється настання взаємоблокування. Надмірне збільшення Δt_s може призвести до пропуску короткочасної ресурсної залежності, а його зменшення – до зростання обсягу майже тотожних записів. Значення обох параметрів мають установлюватися відповідно до швидкості зміни процесів і допустимого часу формування прогнозу.

Сформовані стани накопичуються у сховищі спостережень. Для кожної сигнатури визначається прогнозний маркер, який фіксує, чи відбулося взаємоблокування протягом інтервалу Δt_p. Отже, навчальна вибірка встановлює зв'язок між поточною ресурсною конфігурацією та її подальшим наслідком. Значення інтервалу спостереження має забезпечувати фіксацію швидких переходів без надмірного дублювання майже однакових станів.

Навчальна вибірка формується на основі сигнатур, отриманих у різні моменти часу. Для кожної сигнатури визначається, чи пов'язана вона з переходом у взаємоблокування в найближчому часовому інтервалі. Така побудова дозволяє навчати модель не лише на явних випадках взаємоблокування, а й на попередніх станах, які передують небезпечній ситуації.

Прогнозний маркер виконує роль цільової ознаки, але не входить до матриці кластеризації. Це важливо, оскільки FCM має виявити природні групи ресурсних

станів без прямого нав’язування мітки. Після завершення кластеризації маркер використовується для оцінювання того, наскільки кожен кластер пов’язаний із майбутнім взаємоблокуванням. Такий підхід відрізняється від прямої класифікації [109; 111-112].

У прямій класифікації модель одразу навчається розділяти приклади за міткою. У запропонованій схемі спочатку формується нечітка структура простору ознак, а вже потім кожному кластеру надається інтерпретація через частку небезпечних прикладів. Це дає змогу отримати правила, які залишаються зрозумілими для аналізу [108; 110; 127].

Особливу увагу потрібно приділяти балансуванню навчальної вибірки. У більшості реальних систем нормальних станів значно більше, ніж станів, пов’язаних із взаємоблокуванням. Якщо це не враховувати, кластери можуть переважно описувати безпечну поведінку, а граничні стани залишаться погано представленими. Тому при формуванні вибірки доцільно контролювати наявність прикладів різного типу [126].

Навчальні сигнатури повинні охоплювати різні сценарії роботи системи: нормальне виконання, короткочасне очікування ресурсу, накопичення черги процесів, конфлікти доступу, фактичні взаємоблокування. Якщо в даних присутній лише один режим роботи, то модель не зможе надійно відрізнити небезпечну ситуацію від допустимого очікування.

Для кожного прикладу важливо зберігати зв’язок між підготовленим вектором ознак і вихідною сигнатурою. Це необхідно для інтерпретації результатів, перевірки кластерів і вибору процесу, на який потрібно вплинути. Матриця для FCM містить числові вектори, але після прогнозування рішення має бути застосоване до конкретного PID.

Якщо дані збираються протягом тривалого часу, у вибірці можуть з’являтися застарілі шаблони поведінки. Наприклад, після оновлення програмного забезпечення або зміни політики доступу характер використання ресурсів може змінитися. Тому модель доцільно періодично перенавчати або доповнювати

новими прикладами, щоб правила відображали актуальний режим роботи системи. Перед поданням до FCM дані мають бути очищені від неповних або суперечливих записів.

Якщо в сигнатурі відсутній ідентифікатор процесу, некоректно вказано ресурс або неможливо встановити часовий зв'язок, такий запис може спотворити кластеризацію. Тому формування навчальної вибірки повинно включати етап перевірки коректності зібраних даних.

Сформуємо навчальну вибірку для одного моменту часу системи, як множину навчальних пар так:

$$A^*(t) = \{(x(t), y(x)^{,t}) \mid \exists x \in w_{t,A_t}\} \quad (2.15)$$

де $x(t)$ – множина зайнятих та додаткового ідентифікаторів ресурсів процесу в момент часу t; $y(x)^t$ – прогнозний маркер; w_{t,A_t} – вектор стану КС в момент часу t.

Загальну навчальну вибірку за станами системи W за період від t_s до t_e визначимо так:

$$A^* = \bigcup_{t\in[t_b,t_e]} A^*(t) \quad (2.16)$$

де t_s – час початку спостереження; t_e – час завершення спостереження; $A^*(t)$ – навчальна вибірка для одного моменту часу.

Метод FCM працює з числовою матрицею, тому вихідні сигнатури необхідно перетворити у вектори однакової розмірності. Це перетворення не змінює змісту моделі, а лише приводить різнорідні ресурсні ознаки до форми, придатної для кластеризації. Найважливішою вимогою є збереження інформації про те, які ресурси процес утримує та який ресурс він очікує [111-112; 126].

З метою перетворення ідентифікаторів ресурсів у фіксований вектор використаємо множину ідентифікаторів всіх ресурсів, наявних в системі (R_e).

Для зайнятих ресурсів сформуємо бінарну ознаку так:

$$\forall r_j \in R_e: h_j^{(s),t} = \begin{cases} 1, & r_j \in \{a_{k+1}^{(s),t}, \dots, a_{z-1}^{(s),t}\}, \\ 0, & r_j \notin \{a_{k+1}^{(s),t}, \dots, a_{z-1}^{(s),t}\}, \end{cases} \qquad j = 1, \dots, |R_e| \qquad (2.17)$$

де $h_j^{(s),t} = 1$ означає, що процес a у момент часу t займає ресурс r_j; R_e – множина ідентифікаторів ресурсів КС; r_j – ідентифікатор ресурсу КС; $a_{k+1}^{(s),t}, \dots, a_{z-1}^{(s),t}$ – ідентифікатори ресурсів, які процес уже займає;

Окремо кодуємо ресурс, який процес очікує, так:

$$q_j^{(s),t} = \begin{cases} 1, & a_z^{(s),t} = r_j, \\ 0, & a_z^{(s),t} \neq r_j, \end{cases} \qquad j = 1, \dots, m \qquad (2.18)$$

де $a_z^{(s),t}$ – ідентифікатор запитуваного ресурсу

Якщо процес не очікує ресурс, тобто $a_z^{(s),t} = 0$, тоді

$$q_j^{(s),t} = 0, \qquad j = 1, \dots, m.$$

де $q_{\text{j}}^{(s),t}$ – j-й нормалізований ідентифікатор запитуваного ресурсу, m – кількість ресурсів КС.

Після цього кожна сигнатура процесу подається у вигляді фіксованого вектора:

$$\tilde{a}^{(s)}(t) = \left(h_1^{(s),t}, \dots, h_m^{(s),t}, q_1^{(s),t}, \dots, q_m^{(s),t}\right), \qquad (2.19)$$

де $h_1^{(s),t}, \dots, h_m^{(s),t}$ – нормалізовані ідентифікатори ресурсів, які процес уже займає; $q_1^{(s),t}, \dots, q_m^{(s),t}$ – нормалізовані ідентифікатори ресурсів, які процес запитує.

Розмірність такого вектора стала для всіх процесів:

$$\dim(\tilde{a}^{(s)}(t)) = 2m \qquad (2.20)$$

де m – кількість ресурсів КС; $\tilde{a}^{(s)}(t)$ – нормалізована сигнатура процесу.

Отже, процес із двома зайнятими ресурсами і процес із п’ятдесятьма зайнятими ресурсами подаються в одному просторі ознак однакової розмірності.

Бінарне кодування зайнятих ресурсів дозволяє уникнути залежності від

порядку їх переліку в сигнатурі. Якщо процес утримує ресурси з певними ідентифікаторами, відповідні позиції у векторі отримують одиничні значення. Якщо ресурс не зайнятий процесом, позиція залишається нульовою. Таке подання є простим для обчислення і добре інтерпретується після кластеризації [125-126].

Окремий блок ознак для запитуваного ресурсу показує, який саме ресурс став причиною очікування. Це важливо для розрізнення двох ситуацій. Процес може утримувати ресурс і не очікувати нічого або утримувати той самий ресурс і очікувати на інший. У першому випадку ризик буде низьким, а у другому – істотно вищим, якщо інші процеси формують зворотну залежність.

Фіксована розмірність вектору забезпечує коректне формування матриці X. Кожний рядок матриці відповідає одній сигнатурі, а кожний стовпець – одній ознаці. Завдяки цьому FCM може обчислювати відстань між прикладами та визначати центри кластерів. Без єдиної розмірності порівняння сигнатур було б математично некоректним. На етапі підготовки ознак важливо не змішувати часовий параметр із вектором ресурсного стану, якщо час використовується лише для маркування майбутнього переходу.

Час фіксує порядок спостережень і допомагає сформувати прогнозний маркер, але сам по собі не повинен створювати штучну близькість або віддаленість між процесами в просторі ресурсних ознак. Якщо певні ресурси майже не зустрічаються в навчальній вибірці, то їх ознаки можуть бути маловпливовими. Проте вилучати їх потрібно обережно, оскільки рідкісний ресурс може бути критичним для окремого сценарію взаємоблокування. Рішення про скорочення ознак має прийматися після аналізу частоти використання ресурсів і впливу на якість прогнозування.

Після перетворення сигнатури у вектор ознак необхідно зберегти відповідність між рядком матриці та початковим процесом. Це дозволяє після кластеризації повернутися від абстрактного рядка X до конкретної сигнатури, PID і ресурсів. Без цього зв’язку результат моделі буде важко використати для управлінського рішення в операційних системах.

З усіх таких перетворених сигнатур, що входять до навчальної вибірки A^*, формуємо матрицю даних для FCM так:

$$X = \left[a_{s_1}^{t_1}; a_{s_2}^{t_2}; \dots; a_{s_{|A^*|}}^{t_{|A^*|}}\right] \tag{2.21}$$

де $a_{s_i}^{t_i}$ – внормована сигнатура і-го процесу, $|A^*|$ – кількість навчальних сигнатур у вибірці.

Оскільки кожна сигнатура має розмірність $2m$, то розмірність матриці визначимо так:

$$|X| = |A^*| \times 2m, \tag{2.22}$$

де $|A^*|$ - кількість навчальних сигнатур у вибірці, m – кількість ресурсів.

До матриці X включаємо лише перетворені сигнатури a_s^t. Час t та прогнозний маркер $y(a)^t$ не включаємо до матриці FCM. Маркер $y(a)^t$ зберігається для відповідного рядка матриці та використовується після кластеризації для оцінювання ризику кластера.

Внесок компонентів сигнатури визначається їхньою роллю у формуванні ресурсних залежностей, тобто ознаки утримуваних ресурсів відображають зайняті об’єкти, а запитуваного ресурсу - напрям очікування. Вилучення будь-якої з цих складових порушує повноту опису ресурсного конфлікту. Ідентифікатори PID, PPID, UID та часові характеристики не входять до вектору кластеризації, тому не можуть безпосередньо впливати на її результат.

Для кластеризації виділимо чотири кластери $c = 4$. Вибір такої кількості кластерів відповідає чотирьом ресурсним станам: нормальна робота, очікування ресурсу без критичного ризику, граничний стан, взаємоблокування.

Результат роботи FCM задамо так:

$$FCM(X, 4) \to \{v_j, u_{ij}\}, j = \overline{1,4}, i = \overline{1, |A^*|} \tag{2.23}$$

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.

FCM обрано тому, що ресурсні стани процесів не завжди мають чіткі межі. Один і той самий процес може частково відповідати нормальному режиму, але водночас мати ознаки граничного стану. Жорстке віднесення до одного класу в такій ситуації може приховати ризик. Нечітка кластеризація дозволяє зберегти ступені належності до кількох станів одночасно. У цільовій функції FCM як базову міру близькості використано квадрат евклідової норми; обґрунтування цього вибору для бінарних ресурсних сигнатур наведено нижче [108-109; 111-112].

Центр кластера інтерпретується як типовий ресурсний стан певної групи процесів. Якщо кластер відповідає нормальній роботі, то його центр відображає комбінації ресурсів, які часто зустрічаються без небезпечних наслідків.

Якщо кластер пов’язаний із взаємоблокуванням, то його центр описує комбінації ознак, близькі до історичних небезпечних прикладів. Ступінь належності показує не абсолютну істину, а міру близькості конкретної сигнатури до центру кластера. Це дозволяє працювати з невизначеністю, яка є типовою для динамічних систем. Процес може одночасно мати значну належність до кластера очікування ресурсу та помірну належність до граничного кластера, що є важливою ознакою для прогнозування.

Використання чотирьох кластерів пов’язане з логікою переходу від безпечного стану до взаємоблокування. Перший кластер описує нормальну роботу, другий – очікування ресурсу без критичного ризику, третій – граничний стан, четвертий – взаємоблокування. Така структура необхідна для побудови правил і прийняття рішення.

Кількість кластерів не є довільним числом, а відповідає змістовній інтерпретації станів. Якщо кластерів буде менше, то граничні ситуації можуть змішатися з нормальним очікуванням. Якщо кластерів буде значно більше, то правила стануть менш зрозумілими, а система втратить простоту інтерпретації. Чотири кластери забезпечують баланс між деталізацією та практичністю.

Параметр нечіткості впливає на те, наскільки розмитими будуть межі між кластерами. При меншому значенні належність наближається до жорсткого поділу,

а при більшому – приклади розподіляються між кластерами рівномірніше [111-112; 114].

Для задачі прогнозування взаємоблокування надмірно жорсткий поділ небажаний, оскільки небезпечні стани часто формуються поступово. Результати FCM не слід розглядати окремо від прогнозного маркера. Сам алгоритм кластеризації групує приклади за схожістю, але не знає, який кластер є небезпечним.

Небезпечність визначається після кластеризації через аналіз маркерів тих сигнатур, які з високим ступенем належності входять до відповідного кластера. У практичному використанні після навчання доцільно перевірити склад кожного кластера. Для цього аналізуються типові ресурси, середні значення ознак, частка процесів із прогнозним маркером та приклади сигнатур з найбільшою належністю. Така перевірка допомагає підтвердити, що кластер має зрозумілу інтерпретацію та може бути використаний для формування правила.

У стандартній процедурі нечіткої кластеризації С-середніх (FCM) розбиття простору ознак визначається шляхом мінімізації зваженої суми квадратів відстаней між навчальними векторами та центрами кластерів. Для прийнятих у роботі позначень цільову функцію визначимо так [109; 111-112]:

$$J_{m_f}(U,V) = \sum_{i=1}^{n}\sum_{j=1}^{c} u_{ij}^{m_f} \parallel x_i - v_j \parallel_2^2 \qquad (2.24)$$

де n – кількість навчальних сигнатур; c – кількість кластерів; u_{ij} – ступінь належності і-ї сигнатури до j-го кластера; $m_f > 1$ – параметр нечіткості; v_j – центр j-го кластера. Відстань між сигнатурою та нечітким центром визначимо через евклідову норму так:

$$d_E(x_i, v_j) = \parallel x_i - v_j \parallel_2 = \sqrt{\sum_{k=1}^{2m}(x_{ik} - v_{jk})^2} \qquad (2.25)$$

де x_{ik} – параметри нормалізованої сигнатури процесу; v_{jk} - центр j-го кластера; m – кількість ресурсів.

Квадрат евклідової відстані забезпечує існування аналітичного правила оновлення центрів. Для фіксованої матриці належності мінімум квадратичної функції досягається у зваженому середньому навчальних векторів [111; 112], яке визначимо так:

$$v_j = \frac{\sum_{i=1}^{n} u_{ij}^{m_f} x_i}{\sum_{i=1}^{n} u_{ij}^{m_f}} \tag{2.26}$$

де x_i – параметри нормалізованої сигнатури процесу; u_{ij} – ступінь належності і-ї сигнатури до j-го кластера; $m_f > 1$ – параметр нечіткості.

Вибір евклідової норми як базової міри близькості для бінарно кодованих ресурсних сигнатур обґрунтовується такими теоретико-методологічними положеннями та розрахунками розрізнювальної здатності метрик [125-126].

1. Математичний зв'язок із відстанню Геммінга на вихідній вибірці. Для довільної пари суто бінарних векторів стану x_i та x_ℓ різниця кожної пари компонентів набуває значень -1, 0 або 1. Оскільки для множини {-1, 0, 1} квадрат різниці чисельно збігається з її модулем, квадрат евклідової відстані є строго еквівалентним ненормованій відстані Геммінга (d_H) [125] і визначимо його так:

$$d_E^2(x_i, x_\ell) = \sum_{k=1}^{2m}(x_{ik} - x_{\ell k})^2 = \sum_{k=1}^{2m}|x_{ik} - x_{\ell k}| = d_H(x_i, x_\ell) \tag{2.27}$$

де $x_{ik},\ x_{\ell k}$ – параметри нормалізованих довільних сигнатур i-го та ℓ-го процесів; m – кількість ресурсів.

Нехай маємо два вектори сигнатур у просторі розмірності $2m = 6$ (де перші 3 компоненти – утримання ресурсів, наступні 3 – запити): процес $x_1 = (1,1,0,0,0,0)$ (утримує 2 ресурси, запитів немає) та процес $x_2 = (1,1,0,0,1,1)$ (утримує ті самі ресурси, але додатково запитує 2 нових). Для цієї пари станів покомпонентна розбіжність становить: $d_H(x_1, x_2) = |1-1| + |1-1| + |0-0| + |0-0| + |0-1| + |0-1| = 2$. Відповідно, евклідова відстань дорівнює: $d_E(x_1, x_2) = \sqrt{(1-1)^2 + (1-1)^2 + (0-0)^2 + (0-0)^2 + (0-1)^2 + (0-1)^2} = 1{,}414$. Цей

чисельний результат демонструє, що на етапі оцінювання первинних бінарних даних евклідова метрика лінійно корелює з кількістю незбіжних бітів системи ($d_E = \sqrt{d_H}$), повністю зберігаючи дискретну топологію простору ознак.

2. Неперервний характер нечітких прототипів (центрів кластерів). Згідно з виразом для обчислення v_j, навіть за умови виняткової бінарності вихідних даних, координати центрів кластерів після усереднення стають дійснозначними й належать інтервалу [0; 1]. Значення кожної компоненти v_{jk} інтерпретується як ступінь характерності (частота появи) ознаки утримання чи запиту конкретного ресурсу для певного кластера. Застосування класичних дискретних метрик на етапі прогнозування вимагало б штучної бінаризації центра за порогом, що призвело б до втрати інформації про перехідні або неоднорідні стани [111-; 112; 125].

Нехай розрахований центр кластера "Взаємоблокування" має дробові координати: v = (0.9; 0.8; 0.1; 0.05; 0.95; 0.9), що відображає високу ймовірність утримання перших двох ресурсів та критичну частоту запитів на останні два ресурси. Якщо поточний стан нового процесу становить $x_3 = (1,1,0,0,1,1)$, то класична дискретна відстань Геммінга без штучного округлення центра v втрачає математичний зміст. Натомість евклідова відстань коректно обчислює міру близькості до нечіткого прототипу: $d_E(x_3, v) = \sqrt{(1-0{,}9)^2 + (1-0{,}8)^2 + (0-0{,}1)^2 + (0-0{,}05)^2 + (0-0{,}95)^2 + (0-0{,}9)^2} = 0{,}274$. Значення $d_E \approx 0{,}274$ (близьке до 0) чисельно підтверджує високу математичну збіжність процесу x_3 із нечітким еталоном взаємоблокування, що дозволяє алгоритму FCM безпомилково обчислити високий ступінь належності u_{ij} за цільовою формулою.

3. Інформативність спільних нульових компонентів (порівняння з мірою Жаккара). На відміну від коефіцієнтів Жаккара (K_J) та Дайса, які орієнтовані на сильно розріджені дані й повністю ігнорують збіги виду 0–0 (відсутність ознаки), евклідова норма враховує такі збіги як вагомий чинник близькості станів. У задачі прогнозування взаємоблокувань факт того, що процес не утримує критичні ресурси й

не формує на них запити, є ключовим індикатором безпечного функціонування системи [125].

Розглянемо два безпечні процеси в системі з великою кількістю ресурсів ($2m = 20$), які використовують різні ресурси, але більшість маркерів у них нульові (немає запитів і утримань): x_4 має лише одну одиницю на першій позиції, x_5 – лише одну одиницю на другій позиції. Решта 18 компонентів в обох векторах дорівнюють 0. Розрахунок коефіцієнта подібності Жаккара дає результат: $K_J(x_4, x_5) = \frac{|x_4 \cap x_5|}{|x_4 \cup x_5|} = \frac{0}{1+1-0} = 0$. Значення 0 свідчить про повну відсутність подібності між процесами, оскільки міра ігнорує 18 спільних нулів. Якщо зафіксувати стан системи, де всі процеси мають $K_J = 0$, алгоритм класифікації не зможе виділити єдиний кластер "Нормальна робота". Розрахунок евклідової відстані для цієї ж пари станів дає: $d_E(x_4, x_5) = \sqrt{(1-0)^2 + (0-1)^2 + 18 \cdot (0-0)^2} = 1{,}414$. При цьому максимальна можлива відстань у цьому просторі дорівнює $\sqrt{20} \approx 4{,}472$. Числовий показник $d_E = 1{,}412$ відносно максимуму 4,472 показує, що процеси знаходяться досить близько один до одного завдяки спільній пасивності (відсутності запитів критичних ресурсів). Це чисельно доводить перевагу евклідової відстані.

Проведений аналіз підтверджує, що вибір евклідової відстані є математично узгодженим із цільовою функцією FCM, забезпечує лінійну обчислювальну складність O(m) на одну ітерацію, коректно оперує нечіткими дійснозначними центрами й зберігає предметний зміст як наявних, так і відсутніх ресурсних зв'язків у комп'ютерній системі. Наведене обґрунтування визначає доцільність використання евклідової метрики в межах прийнятого бінарного подання ресурсних станів і класичного алгоритму Fuzzy C-Means.

Оцінювання ризику кластера.

Для кожного рядка матриці X відомий відповідний прогнозний маркер y_i, отриманий із $y(a)_t$. Він показує, чи перейшла відповідна сигнатура процесу у взаємоблокування протягом наступного інтервалу часу Δt.

Ризик j-го кластера обчислюється як зважена частка сигнатур, для яких прогнозний маркер дорівнює 1:

$$P_j = \frac{\sum_{i=1}^{|A^*|} u_{ij}^{m_f} \cdot y_i}{\sum_{i=1}^{|A^*|} u_{ij}^{mf}}, j = \overline{1,4} \quad (2.28)$$

де m_f – параметр нечіткості FCM; u_{ij} – ступінь належності і-ї сигнатури до j-го кластера; y_i – прогнозний маркер; $|A^*|$ – кількість навчальних сигнатур у вибірці.

Значення P_j показує, яка частка сигнатур, близьких до кластера j, у найближчому майбутньому переходила у взаємоблокування. Якщо P_j наближається до нуля, то кластер відповідає безпечному стану. Якщо P_j наближається до одиниці, то кластер відповідає станам із високим ризиком взаємоблокування.

Після кластеризації кожен кластер отримує оцінку ризику. Ця оцінка показує, наскільки часто сигнатури, близькі до кластера, переходили у взаємоблокування в майбутньому часовому інтервалі.

Таким чином, ризик не задається експертно, а визначається на основі навчальних даних і ступенів належності сигнатур до відповідного кластера. Зважування через ступені належності є важливим, оскільки приклади не належать кластерам однаково. Сигнатура, що майже збігається з центром кластера, повинна сильніше впливати на його оцінку, ніж сигнатура, що перебуває на межі між кількома кластерами. Саме тому при обчисленні ризику враховується нечітка належність, а не лише факт найбільшої близькості.

Якщо ризик кластера близький до нуля, це означає, що процеси з подібними ресурсними станами в історичних даних переважно не переходили у взаємоблокування. Такий кластер можна інтерпретувати як безпечний або низькоризиковий. Якщо ризик наближається до одиниці, кластер описує конфігурації, які часто передували або відповідали взаємоблокуванню.

Проміжні значення ризику мають найбільшу практичну цінність. Вони показують стани, які не є гарантовано небезпечними, але потребують додаткової уваги. Саме такі стани можуть відповідати граничним ситуаціям, коли взаємоблокування ще не сталося, але подальше очікування ресурсу або неправильний порядок звільнення може призвести до небезпечної конфігурації.

Оцінка ризику кластера також дозволяє порівнювати правила між собою.

Якщо два правила мають схожі передумови, але різні значення ризику, це може свідчити про недостатню якість кластеризації або про потребу уточнити ознаки. Якщо ж центри кластерів відрізняються, а ризики логічно узгоджуються з їх змістом, база правил є інтерпретованою.

У процесі експлуатації значення ризику можна використовувати для встановлення порогів реагування. Наприклад, при низькому ризику система може тільки реєструвати подію, при середньому – формувати попередження, а при високому – ініціювати активну дію для недопущення взаємоблокування. Такий механізм дозволяє уникати надмірного втручання в роботу процесів.

Оскільки ризик визначається за навчальними даними, він залежить від якості історичних спостережень. Якщо в даних мало прикладів фактичного взаємоблокування, оцінка дуже ризикованого кластера може бути нестійкою. У цьому випадку доцільно використовувати додаткову експертну перевірку або розширювати вибірку штучно змодельованими сценаріями. Важливо, що ризик кластера не обов'язково дорівнює поточній імовірності взаємоблокування для конкретного процесу. Він відображає історичну небезпечність групи схожих станів. Для конкретної сигнатури остаточна оцінка формується через ступені спрацювання правил і належність до третього та четвертого кластерних станів.

Генерація нечітких правил.

Кожен кластер, отриманий методом FCM, задає одне нечітке правило. Передумова правила визначається близькістю перетвореної сигнатури до центра кластера v_j, а висновок правила визначається значенням ризику P_j:

$$R_j: IF\ a_s^t \sim v_j\ THEN\ P_d = P_j, j = \overline{1,4} \qquad (2.29)$$

де $a_s^t \sim v_j$ – перетворена сигнатура процесу належить до кластера з центром v_j з відповідним ступенем належності u_{ij}.

Оскільки для кластеризації використовується чотири кластери, то базу правил визначимо так:

$$B = \{R_1, R_2, R_3, R_4\} \quad (2.30)$$

де R_1 – правило, що визначає належність сигнатури до кластера "нормальна робота"; R_2 – правило, що визначає належність сигнатури до кластера "очікування ресурсу без критичного ризику"; R_3 – правило, що визначає належність сигнатури до кластера "граничний стан"; R_4 – правило, що визначає належність сигнатури до кластера "взаємоблокування"

Правила формуються автоматично, на етапі навчання, на основі центрів кластерів FCM, а їх висновок визначається за прогнозними маркерами навчальних сигнатур. Це дозволяє пов’язати ресурсний стан процесу з оцінкою ризику переходу у граничний [110; 115; 127].

Кожен кластер, отриманий методом FCM, перетворюється на нечітке правило. Передумова правила задається близькістю сигнатури до центру кластера, а висновок – оцінкою ризику, пов’язаною з цим кластером. Така побудова дозволяє автоматизувати формування бази правил і зменшити залежність від ручного експертного опису всіх можливих комбінацій ресурсів.

На відміну від класичних експертних правил, у яких умови часто формуються вручну, правила на основі FCM виникають із структури даних. Це важливо для операційних систем, де кількість можливих комбінацій процесів і ресурсів є дуже великою. Автоматичне формування правил дозволяє виявляти типові шаблони без повного перебору всіх сценаріїв. Передумова правила не описується простим твердженням про наявність одного ресурсу.

Вона фактично відповідає близькості всього підготовленого вектору ознак до центру кластера. Тому правило враховує комбінацію зайнятих ресурсів, запитуваного ресурсу та, за потреби, додаткових характеристик процесу. Це робить правило більш гнучким.

Висновок правила задається не бінарним рішенням, а значенням ризику. Завдяки цьому система може працювати з різними рівнями небезпеки. Наприклад, правило, пов’язане з нормальним очікуванням ресурсу, може мати низький ризик, тоді як правило четвертого кластера матиме високий ризик і

використовуватиметься для активного реагування. База з чотирьох правил є компактною, але достатньою для інтерпретації основних станів.

Кожне правило відповідає одному з кластерів, а сукупність правил описує перехід від нормальної роботи до взаємоблокування. Такий формат зручний для подальшої інтеграції в модуль аналізу поведінки процесів операційної системи.

Автоматично сформовані правила потрібно перевіряти на змістову узгодженість. Якщо правило з високим ризиком відповідає центру, у якому відсутній запит на ресурс, це може означати помилку в даних або некоректне кодування ознак. Якщо ж високий ризик пов’язаний з очікуванням ресурсу при наявності зайнятих критичних ресурсів, таке правило має логічне пояснення.

Нечіткі правила також дозволяють не втрачати інформацію на межі кластерів. Якщо процес частково належить до кластера нормального очікування і частково до граничного кластера, обидва правила можуть спрацювати з різною силою. У результаті система отримує не жорстке рішення, а агреговану оцінку ризику, що краще відповідає природі динамічних процесів. Після навчання база правил може використовуватися повторно для поточних станів системи.

Для кожної нової сигнатури обчислюється ступінь близькості до центрів кластерів, після чого активуються відповідні правила. Це дозволяє швидко отримувати прогноз без повторної кластеризації всієї навчальної вибірки при кожній події ненадання ресурсу.

Для наочного подання результатів нечіткої кластеризації багатовимірні сигнатури процесів відображено на двовимірну площину узагальнених ресурсних характеристик (рисунок 2.6). Горизонтальна координата характеризує інтенсивність утримання та використання ресурсів, а вертикальна – вираженість ресурсного очікування і конфліктності сформованого запиту. Кожна точка площини відповідає ознаковій частині окремої навчальної пари, тоді як її прогнозний маркер використано для визначення змісту отриманого кластера. На площині сформовано чотири нечіткі області, які відповідають: нормальній роботі, очікуванню ресурсу, граничному стану та взаємоблокуванню. Перекриття областей

відображає можливість одночасної належності сигнатури до декількох класів. База нечітких правил визначає ступінь відповідності поточної сигнатури кожному класу, а прогнозований стан установлюється за правилом із найбільшим ступенем спрацювання.

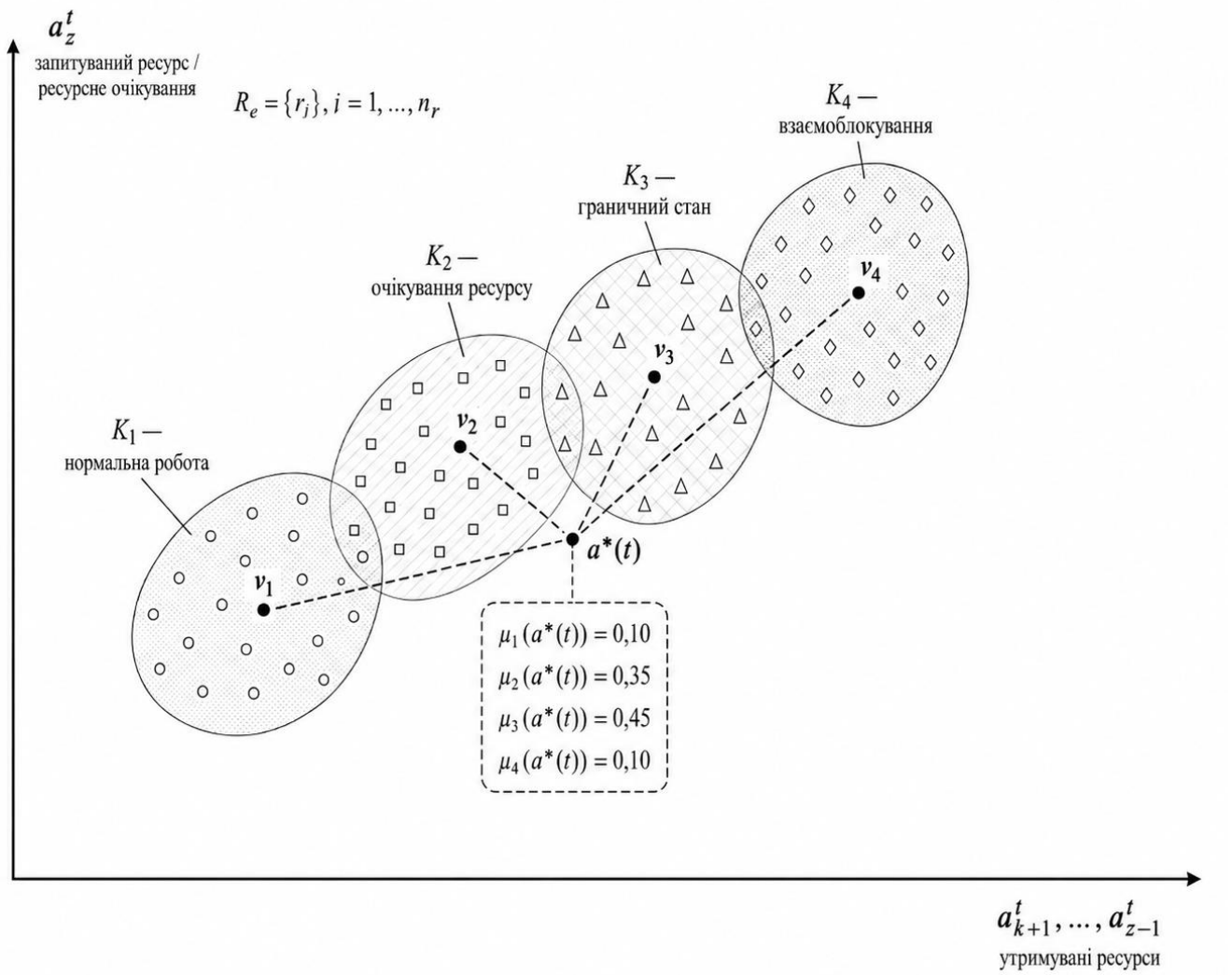


Рисунок 2.6 – Відображення навчальних сигнатур на площині ознак та їх нечітка класифікація за ресурсними станами

Розроблено модель формування бази нечітких правил, у межах якої ресурсні стани процесів перетворено на навчальні пари, приведено до векторів однакової розмірності та згруповано методом нечіткої кластеризації. Для кожного сформованого кластера визначено рівень ризику за фактичними наслідками розвитку навчальних станів, що забезпечило автоматичне одержання правил переходу від нормальної роботи до взаємоблокування. Водночас наявність бази

правил не забезпечує оцінювання поточного стану процесу під час функціонування комп’ютерної системи. Для застосування отриманої бази необхідно розробити модель прогнозування взаємоблокування процесів, яка визначатиме ступені належності нової сигнатури до кластерів, активуватиме відповідні правила та формуватиме підсумкову оцінку ризику виникнення взаємоблокування.

2.5 Модель прогнозування взаємоблокування процесів в комп'ютерних системах

Прогнозування активується не для кожної зміни стану системи, а насамперед після події, коли процес не отримав запитуваний ресурс. Такий підхід зменшує кількість непотрібних обчислень і фокусує аналіз на ситуаціях, у яких потенційно може виникнути взаємоблокування. Якщо процес не очікує ресурсу, його поточний стан не є прямим кандидатом на взаємоблокування [122].

Після фіксації події ненадання ресурсу формується оновлена множина сигнатур. Вона повинна відображати актуальний стан процесів після того, як один або кілька процесів перейшли в очікування. Саме ця оновлена множина використовується для відбору сигнатур, які мають ненульовий запит на ресурс і тому потребують прогнозної оцінки. Відбір сигнатур за наявністю запиту на ресурс є логічним обмеженням простору аналізу.

У великій системі може працювати багато процесів, але лише частина з них перебуває у стані очікування ресурсу. Аналіз усіх процесів без такого відбору збільшив би навантаження і міг би створювати надмірну кількість попереджень. Для кожної відібраної сигнатури виконується те саме перетворення в підготовлений вектор ознак, що використовувалося на етапі навчання.

Це є обов’язковою умовою коректності прогнозу: модель не може застосовуватися до даних, закодованих за іншими правилами. Тому множина ресурсів, порядок ознак і спосіб кодування повинні бути однаковими. Підготовлений вектор подається на базу нечітких правил. Кожне правило спрацьовує відповідно до близькості цього вектора до свого кластера.

Ступінь близькості нового ресурсного вектора до кожного центра обчислюється за тією самою евклідовою відстанню, що використовувалася на етапі навчання; це забезпечує узгодженість формування кластерів і подальшої активації правил.

Якщо вектор близький до центру четвертого кластера, ступінь спрацювання правила взаємоблокування буде високим. Якщо він ближчий до нормального або низькоризикового кластера, відповідні правила отримають більшу вагу.

Оцінювання належності до третього і четвертого кластерів має окремий зміст. Третій кластер інтерпретується як граничний стан, тобто стан, у якому взаємоблокування ще може не бути зафіксованим, але умови для нього вже сформовані. Четвертий кластер відповідає стану взаємоблокування або максимальному наближенню до нього за структурою ресурсних залежностей [118; 122].

Нечітке об’єднання належностей до третього та четвертого кластерів дозволяє не пропускати процеси, які перебувають між граничним і фактично заблокованим станом. Якщо враховувати лише четвертий кластер, система може реагувати запізно. Якщо враховувати лише третій, вона може не відрізнити критичну ситуацію від проміжного очікування. Об’єднання цих ознак дає збалансовану оцінку.

Результатом прогнозування є множина процесів, що мають підвищений ризик переходу до взаємоблокування або вже належать до стану взаємоблокування. Надалі ця множина може використовуватися для вибору керуючої дії. Така дія повинна бути обережною, оскільки помилкове завершення системного або критичного процесу може погіршити стан операційної системи.

Після виникнення події, пов’язаної з ненаданням процесу запитуваного ресурсу, стан комп’ютерної системи оновлюється. У момент часу $t + \Delta t$ формується оновлена множина сигнатур процесів:

$$A^{t+\Delta t} = \{a_i^{t+\Delta t} \mid i = 1, \dots, n_p\} \tag{2.31}$$

де $a_i^{t+\Delta t}$ – сигнатура i-го процесу.

До прогнозування подаються не всі сигнатури з множини $A^{t+\Delta t}$, а лише ті, що відповідають процесам з $M_{OS,pr,p}$ та у яких зафіксовано запит на ресурс з M_R. Умова відбору сигнатури має вигляд:

$$a_i^{t+\Delta t} \in A^{t+\Delta t} \vee a_{i,z}^{t+\Delta t} \neq 0 \tag{2.32}$$

де $a_i^{t+\Delta t}$ – сигнатура *i*-го процесу; $a_{i,z}^{t+\Delta t}$ – ідентифікатор запитуваного ресурсу.

Якщо $a_{i,z}^{t+\Delta t} = 0$, то процес не має актуального запиту на ресурс і до процедури прогнозування не подається. Якщо $a_{i,z}^{t+\Delta t} \neq 0$, то сигнатура використовується для подальшого оцінювання.

Для кожної такої сигнатури формується підготовлений вектор ознак:

$$a_i^{t+\Delta t} \rightarrow a_i^{*\,t+\Delta t} \tag{2.33}$$

де $a_i^{t+\Delta t}$ – сигнатура *i*-го процесу; $a_i^{*t+\Delta t}$ – нормована сигнатура *i*-го процесу.

Підготовлений вектор ознак подається на базу нечітких правил B, сформовану на етапі побудови бази правил згідно з формулою (2.30).

Кожне правило бази B відповідає одному з чотирьох кластерів, сформованих методом FCM згідно з формулою (2.29).

Правило R_l спрацьовує тим сильніше, чим ближче підготовлений вектор ознак $a_i^{*\,t+\Delta t}$ до центра кластера v_j. Тому ступінь спрацювання правила використовується як ступінь належності сигнатури до відповідного кластерного стану.

Результат роботи бази правил для сигнатури $a_i^{*\,t+\Delta t}$ подамо у вигляді вектора:

$$F_B\left(a_i^{*\,t+\Delta t}\right) = (\gamma_{i1}, \gamma_{i2}, \gamma_{i3}, \gamma_{i4}) \tag{2.34}$$

де $\gamma_{il} = \mu_{R_l}\left(a_i^{*\,t+\Delta t}\right), l = 1, \ldots, 4$; γ_{il} – ступінь спрацювання правила R_l для сигнатури $a_i^{*\,t+\Delta t}$; γ_{i1} відповідає ступеню належності до першого кластерного стану; γ_{i2} – до другого кластерного стану; γ_{i3} відповідає ступеню належності до третього кластерного стану; γ_{i4} – до четвертого кластерного стану.

Оскільки необхідно визначити належність сигнатури до станів "граничний" або "взаємоблокування", то використаємо нечітке об’єднання, яке задамо так:

$$\gamma_i^{3,4} = \gamma_{i3} \vee \gamma_{i4} = \max(\gamma_{i3}, \gamma_{i4}) \quad (2.35)$$

Тоді умову віднесення сигнатури до множини граничних процесів задамо так:

$$\left(a_{i,z}^{t+\Delta t} \neq 0\right) \wedge \left(\gamma_i^{3,4} > \theta\right) \Rightarrow a_i^{t+\Delta t} \in D_\theta \quad (2.36)$$

де $a_{i,z}^{t+\Delta t}$ – ідентифікатор запитуваного ресурсу; $\gamma_i^{3,4}$ – ступінь належності до третього або четвертого кластерного стану; θ – порогове значення для кластеру, визначене як 0,6; $a_i^{t+\Delta t}$ – сигнатура i-го процесу; D_θ –множина граничних процесів.

Або в розгорнутому вигляді через спрацювання правил γ_3 та γ_4 визначення буде так задане:

$$\forall a_i^{t+\Delta t} \in A^{t+\Delta t}: \left[a_{i,z}^{t+\Delta t} \neq 0 \wedge \max\left(\mu_{R_3}\left(a_i^{*\,t+\Delta t}\right), \mu_{R_4}\left(a_i^{*\,t+\Delta t}\right)\right) > \theta\right] \Rightarrow a_i^{t+\Delta t} \in D_\theta \quad (2.37)$$

де $a_i^{t+\Delta t}$ – сигнатура i-го процесу; $A^{t+\Delta t}$ – стан КС в момент часу $t + \Delta t$; $\mu_{R_3}\left(a_i^{*\,t+\Delta t}\right)$ – ступінь спрацювання правила R_3 для сигнатури $a_i^{*\,t+\Delta t}$; $\mu_{R_4}\left(a_i^{*\,t+\Delta t}\right)$ – ступінь спрацювання правила R_4 для сигнатури $a_i^{*\,t+\Delta t}$; θ – порогове значення для кластеру, визначене як 0,6; D_θ –множина граничних процесів.

З метою недопущення взаємоблокування процесів доцільно завершувати процес, що належить $M_{OS,pr,p}$, не є системним та має найменший час перебування у системі [123].

Визначимо PID ідентифікатор процесу, що має найменший час перебування в системі та не є системним.

Після виявлення множини граничних процесів необхідно визначити, який процес доцільно завершити або іншим способом вивести з конфлікту. У моделі обрано критерій найменшого часу перебування в системі серед процесів, що не є системними.

Такий критерій зменшує ризик втрати важливого довготривалого стану та обмежує втручання в роботу операційної системи. Виключення системних

процесів є необхідним обмеженням. Системні процеси часто забезпечують роботу ядра, служб, драйверів, файлової підсистеми, мережевої взаємодії або механізмів авторизації. Їх завершення може призвести до нестабільності всієї системи.

Тому навіть за високої належності до небезпечного кластера системний процес не повинен автоматично обиратися для знищення. Найменший час перебування в системі використовується як практичний критерій мінімізації втрат. Процес, який працює короткий час, імовірно, має менше накопичених результатів, менше незавершених операцій і менший вплив на інші компоненти.

Завершення такого процесу зазвичай є менш небезпечним, ніж завершення довготривалого процесу, який має багато напрацювань. Однак цей критерій не повинен розглядатися ізольовано від належності до небезпечних кластерів. Спочатку формується множина кандидатів, які мають достатній ступінь належності до граничного або четвертого кластера. Лише після цього серед них застосовується вибір за часом перебування.

Така послідовність гарантує, що процес обирається не просто за віком, а за участю в ризиковій конфігурації. У практичній системі замість негайного завершення процесу можуть застосовуватися м’якші дії: попередження адміністратора, відкладення надання ресурсу, примусове звільнення окремого дескриптора, зниження пріоритету або переведення процесу в контрольований стан очікування.

Вибір конкретної дії залежить від політики безпеки та можливостей операційної системи. Критерій несистемності процесу може визначатися за пріоритетом, власником, шляхом виконуваного файлу, службовою ознакою або політикою адміністратора. У формальній моделі достатньо вказати умову, що відповідний параметр пріоритету або типу процесу не дорівнює системному значенню.

У реалізації ця умова має бути пов’язана з конкретним способом маркування процесів. Якщо серед кандидатів немає жодного несистемного процесу, автоматичне завершення не виконується. У такому випадку система повинна

сформувати повідомлення про неможливість безпечного втручання або передати ситуацію на ручний розгляд. Це запобігає неконтрольованому впливу на критичні служби та зберігає стабільність комп'ютерної системи.

Таким чином, керуюче рішення складається з трьох послідовних етапів: виявлення ризикових сигнатур, відсікання системних процесів і вибір кандидата з мінімальним часом перебування. Така структура відповідає задачі недопущення взаємоблокування без надмірного втручання в нормальну роботу процесів і задається виразом:

$$a_{i,1}^{t+\Delta t}: \left[a_i^{t+\Delta t} \in D_\theta \wedge a_{i,4}^{t+\Delta t} \neq 1 \wedge a_{i,5}^{t+\Delta t} = \min\left\{a_{k,5}^{t+\Delta t} \mid a_k^{t+\Delta t} \in D_\theta \wedge a_{k,4}^{t+\Delta t} \neq 1\right\}\right] \quad (2.38)$$

де D_θ –множина граничних процесів; $a_{i,1}^{t+\Delta t}$ – ідентифікатор i-го процесу; $a_{i,4}^{t+\Delta t}$ – ознака системності i-го процесу; $a_{i,5}^{t+\Delta t}$ – час перебування i-го процесу в системі.

Запропонована модель орієнтована на ситуації, де можна зафіксувати факти зайняття та очікування розподілюваних ресурсів. Якщо операційна система або середовище виконання не надає таких даних, якість прогнозування буде обмеженою.

У такому випадку потрібно або розширювати механізм збору подій, або використовувати непрямі ознаки, наприклад тривале очікування системних викликів. Модель не гарантує повного усунення всіх взаємоблокувань. Її призначення полягає в ранньому виявленні станів, схожих на ті, що в історичних даних призводили до блокування. Якщо виникає новий тип взаємодії процесів, якого не було в навчальній вибірці, модель може оцінити його неточно.

Тому важливим є регулярне оновлення навчальних даних. Точність прогнозування залежить від правильного вибору інтервалу Δt. Якщо інтервал занадто короткий, частина реальних переходів до взаємоблокування може не потрапити в маркер.

Якщо інтервал занадто довгий, маркер може пов'язати поточну сигнатуру з віддаленою подією, яка не є її прямим наслідком. Тому Δt потрібно підбирати

експериментально. Модель також залежить від стабільності ідентифікаторів ресурсів. Якщо той самий логічний ресурс у різні моменти отримує різні ідентифікатори без можливості зіставлення, навчальна вибірка стає менш узгодженою.

Тому на етапі збору даних потрібно забезпечити коректне відображення ресурсів у сталі ідентифікатори. Нечітка природа моделі означає, що результат потрібно інтерпретувати як оцінку ризику, а не як абсолютне твердження.

Узагальнену послідовність реалізації моделі прогнозування стану взаємоблокування процесів зображено на рисунку 2.7.

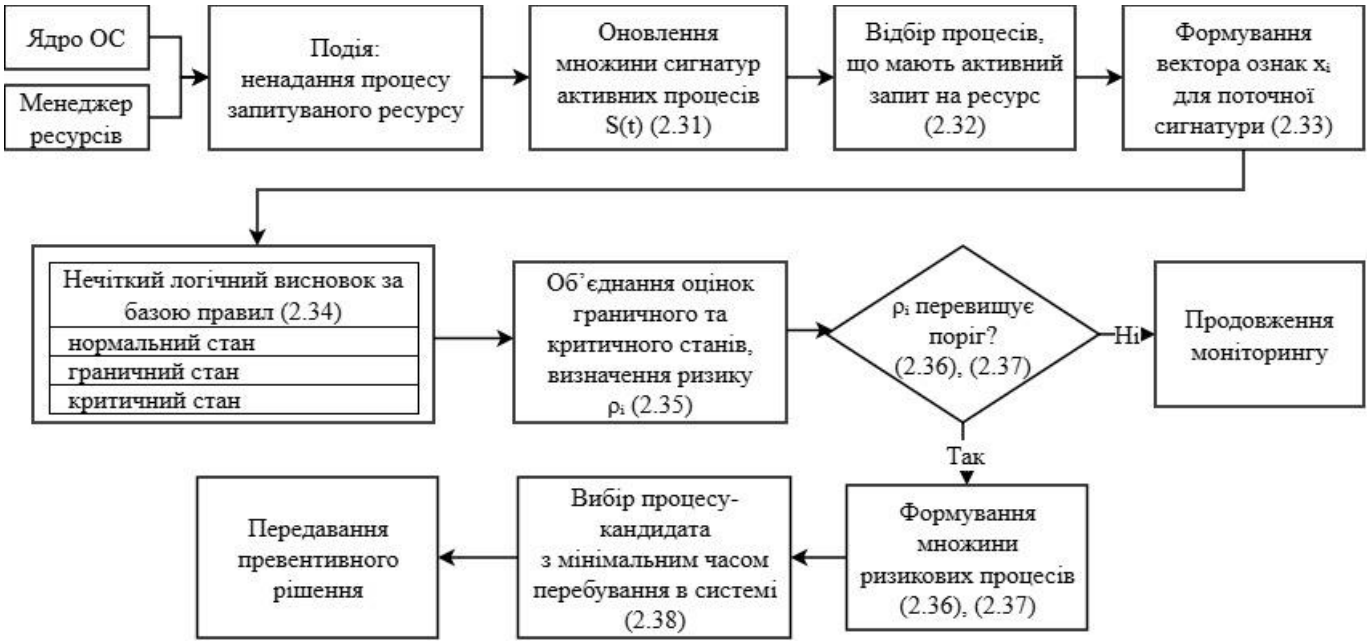


Рисунок 2.7 – Послідовність реалізації моделі прогнозування стану взаємоблокування процесів

Подана схема (рис. 2.7) деталізує контур архітектури системи та встановлює відповідність між її функціональними компонентами і введеними математичними співвідношеннями. Початковою подією для виконання прогнозування є неможливість негайного надання процесу запитуваного ресурсу. Після її фіксації від ядра операційної системи та менеджера ресурсів надходять актуальні відомості про стани процесів, утримувані ними ресурси і сформовані ресурсні запити.

На основі отриманих даних відповідно до виразу з формули (2.31) формується оновлена множина сигнатур процесів, яка характеризує поточний стан

комп’ютерної системи. Із цієї множини за умовою з формули (2.32) відбираються сигнатури процесів, що мають активний запит на ресурс, оскільки саме для них існує потенційна можливість розвитку ресурсного очікування до стану взаємоблокування. Кожна відібрана сигнатура перетворюється за співвідношенням за формулою (2.33) у підготовлений вектор ознак, структура і порядок компонентів якого узгоджені з даними, використаними під час формування бази нечітких правил.

Підготовлений вектор надходить до блока нечіткого логічного висновку, де на основі сформованої бази правил визначаються ступені його належності до кластерних станів. Результат спрацювання правил подається у вигляді вектору з формули (2.34). Для оцінювання небезпеки враховуються ступені належності до третього, граничного, та четвертого, критичного, кластерів, нечітке об’єднання яких виконується відповідно до виразу з формули (2.35). Порогові умови з формул (2.36) і (2.37) забезпечують формування множини процесів, ресурсна поведінка яких відповідає граничному стану або стану взаємоблокування.

На завершальному етапі з отриманої множини вилучаються системні процеси, після чого за критерієм з формули (2.38) визначається процес-кандидат із найменшим часом перебування в системі. Його ідентифікатор разом з оцінкою ризику передається до рівня прийняття превентивного рішення. Залежно від політики керування ресурсами результат може бути використаний для посиленого моніторингу, відкладення ресурсного запиту, зміни черговості доступу, призупинення або завершення некритичного процесу.

Таким чином, розроблена модель формалізує повний ланцюг оперативного прогнозування, тобто від виникнення події ненадання ресурсу та оновлення стану комп’ютерної системи до визначення множини ризикових процесів і формування підстави для превентивного керуючого впливу. Поєднання кластерних оцінок граничного та критичного станів дає змогу виявляти небезпечну ресурсну конфігурацію до остаточного формування циклу взаємного очікування. Виключення системних процесів і вибір кандидата з найменшим часом

перебування в системі обмежують можливі наслідки втручання. Водночас модель визначає склад вхідних даних, математичні залежності та послідовність перетворень, але не встановлює алгоритмічний порядок їх програмного виконання. Тому наступним етапом дослідження є розроблення методів побудови сигнатури процесу, формування бази нечітких правил і прогнозування стану взаємоблокування, які забезпечать практичну реалізацію запропонованих моделей.

2.6 Висновки до другого розділу

Удосконалено архітектуру засобів прогнозування стану взаємоблокування процесів у КС, яка передбачає поєднання модулів збору даних про процеси та ресурси, побудови сигнатур процесів, формування бази нечітких правил і прогнозування стану взаємоблокування. На відміну від відомих підходів, орієнтованих переважно на виявлення або усунення взаємоблокування після його виникнення, розроблена архітектура забезпечує завчасне оцінювання ризику переходу процесів у стан взаємоблокування та створює основу для прийняття превентивних рішень щодо керування ресурсами КС.

Для забезпечення функціонування розробленої архітектури запропоновано модель сигнатури процесу, модель стану КС, модель процесу побудови бази нечітких правил та модель прогнозування взаємоблокування процесів у КС. Розроблені моделі є основою для розроблення методів побудови сигнатури процесу, формування бази нечітких правил і прогнозування стану взаємоблокування, які забезпечують практичну реалізацію запропонованих моделей.

Основні результати розділу опубліковані у працях [118; 120; 121; 123; 124].

РОЗДІЛ 3
МЕТОДИ ПОБУДОВИ СИГНАТУРИ ПРОЦЕСУ І БАЗИ НЕЧІТКИХ ПРАВИЛ ТА ПРОГНОЗУВАННЯ СТАНУ ВЗАЄМОБЛОКУВАННЯ ПРОЦЕСІВ У КОМП'ЮТЕРНИХ СИСТЕМАХ

3.1 Метод побудови сигнатури процесу

В розробленій архітектурі системи прогнозування стану взаємоблокування процесів модуль побудови сигнатур є першим функціональним компонентом аналітичної надбудови та забезпечує перетворення параметрів процесів і відомостей про ресурси на узгоджений опис процесу. У формальному поданні процесу прогнозування цей етап реалізується методом M_1 у формулі (2.2), а у складі функції прогнозу – функцією F_{M_1} у формулі (2.4). Розроблення методу побудови сигнатури є безпосередньою реалізацією передбаченого архітектурою переходу від даних ядра ОС до формалізованої сигнатури процесу.

Необхідність виділення побудови сигнатури в окремий метод зумовлена різним рівнем формалізації архітектури, моделі та програмної реалізації. Архітектура визначає місце модуля й інформаційні зв'язки, а модель сигнатури задає склад і математичну структуру даних, проте вони не встановлюють алгоритмічної послідовності отримання атрибутів від конкретної ОС, їх зіставлення з позиціями сигнатури, синхронізації ресурсних відомостей та правил створення або оновлення запису процесу. Окремий метод локалізує ці операції та не допускає перенесення платформозалежних особливостей у наступні процедури кластеризації і нечіткого логічного висновку.

У формальній архітектурі згідно з формулою (2.1) побудова сигнатури пов'язує три групи системних відомостей: множину активних процесів ОС, множину ресурсів КС та множину їх поточного розподілу між процесами. Тому результат модуля побудови сигнатур не може бути простим копіюванням даних із таблиці процесів або менеджера ресурсів. Для кожного процесу необхідно встановити відповідність між його атрибутами, фактом володіння ресурсами та поточним запитом

на ресурс, після чого подати ці відомості в однаково інтерпретованій структурі. Саме така операція перетворення реалізується методом M_1 у формулі (2.2) і є початковою умовою для роботи методів M_2 та M_3.

Метод побудови сигнатури процесу призначений для формування актуального формалізованого опису процесу в комп'ютерній системі у визначений момент спостереження. Його завдання полягає у перетворенні різнорідних ідентифікаційних, часових, поведінкових і ресурсних відомостей, отриманих від ОС, у впорядковану структуру, що відповідає моделі сигнатури процесу та має однакову семантику для подальших компонентів системи прогнозування.

Відповідно до принципу розмежування інваріантної та платформозалежної частин, сформульованого в архітектурі системи, побудову сигнатури доцільно виконувати у два етапи. На першому етапі платформозалежний адаптер отримує доступні в конкретній ОС характеристики процесу та події взаємодії з ресурсами і приводить їх до спільних семантичних ознак. На другому етапі з уніфікованих значень формується інваріантна структура сигнатури за моделлю згідно з формулами (2.6)–(2.8). Узагальнену схему такого перетворення зображено на рисунку 3.1.

Для Linux та інших Unix-подібних ОС джерелами є записи про активні процеси, атрибути їх виконання, файлові дескриптори та об'єкти синхронізації, для Windows – відповідні параметри процесу і контексту, дескриптори системних об'єктів та відомості про очікування ресурсів. Для інших ОС або RTOS змінюється лише спосіб отримання первинних даних. Метод ґрунтується на забезпеченні семантичної відповідності отримуваних даних без використання наперед визначеного системного API. Незалежно від типу ОС значення системних параметрів мають бути однозначно відображені у відповідні позиції сформованої сигнатури процесу. Це дає змогу переносити систему між ОС шляхом заміни адаптера збирання даних без зміни наступних етапів прогнозування.

Часова узгодженість, закладена в архітектурі системи, для методу 3.1 означає, що відомості про процес, утримувані ресурси і невиконаний запит мають

характеризувати один і той самий момент спостереження. Якщо ці складові отримані в різні моменти без узгодження, може виникнути штучна конфігурація, якої фактично не існувало в КС. Тому метод розглядає системну подію як точку актуалізації сигнатури, після якої до множини A_t передається вже узгоджений запис процесу.



Рисунок 3.1 – Структура сигнатури процесу та її формування для різних типів ОС

Відповідно до моделі з формули (2.6) сигнатура має дві змістові складові. Ідентифікаційна складова охоплює параметри $a_1, \ldots, a_k$, які зв'язують запис із конкретним процесом і характеризують контекст його виконання. Ресурсна складова містить позиції $a_{k+1}, \ldots, a_{z-1}$ для ресурсів, уже утримуваних процесом, та позицію a_z для ідентифікатора запитуваного ресурсу або нульового значення відповідно до формули (2.7).

Векторну структуру цієї сигнатури з поділом на ідентифікаційну та ресурсну складові зображено на рисунку 3.2. Подання сигнатури у формі впорядкованого вектора визначає єдину структуру розміщення її компонентів, проте не передбачає їх

однорідності за фізичною природою та функціональним призначенням. Вектор у цьому випадку задає структуру та порядок ознак. Ідентифікаційні позиції характеризують процес і контекст його виконання, а ресурсні позиції відображають фактичні зв’язки "процес – ресурс".

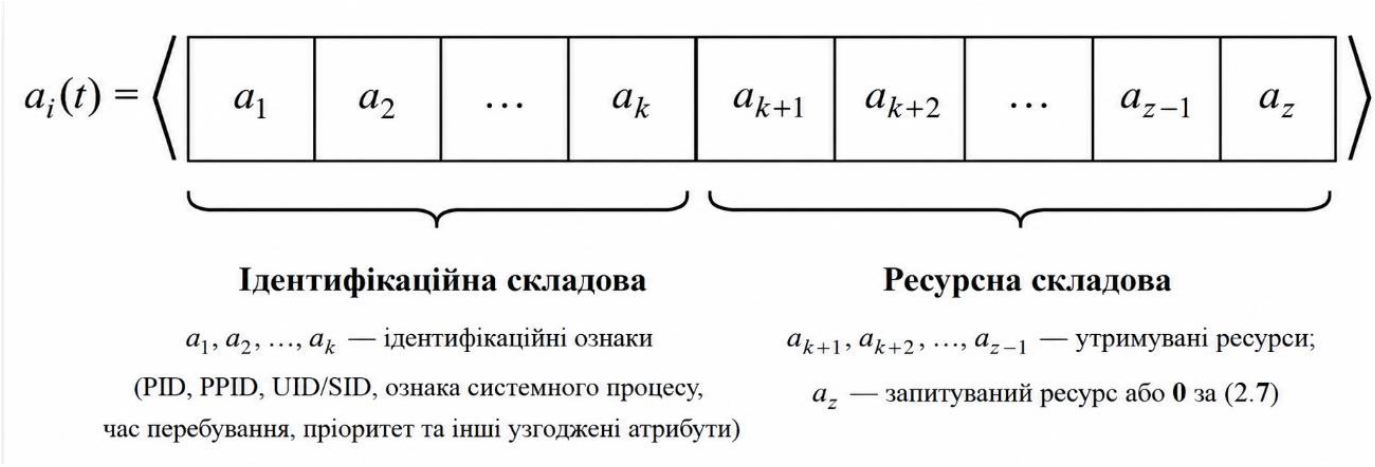


Рисунок 3.2 – Векторна структура сигнатури процесу з ідентифікаційною та ресурсною складовими

Межа між компонентами a_k та a_{k+1} на рисунку 3.2 є змістовою. Компоненти $a_1, \ldots, a_k$ утворюють ідентифікаційну частину і не описують безпосередньо факт ресурсного конфлікту. Вони забезпечують однозначне зв’язування сигнатури з процесом, його походженням та умовами виконання. Компоненти $a_{k+1}, \ldots, a_z$ утворюють ресурсну частину. Саме вона дає змогу визначити, які ресурси процес уже утримує і який додатковий ресурс він намагається одержати в поточний момент.

Особливе значення має остання позиція a_z. Наявність ресурсів у позиціях $a_{k+1}, \ldots, a_{z-1}$ сама по собі не означає небезпечного стану, оскільки штатний процес може утримувати один або декілька ресурсів без очікування. Ненульове значення компонента a_z відображає стан, за якого процес, зберігаючи за собою раніше виділені ресурси, формує запит на отримання додаткового ресурсу. Саме поєднання утримання й очікування надалі дозволяє аналізувати передумови формування циклічних ресурсних залежностей.

Компонент a_{z+1}, визначений у формулі (2.8), принципово відділено від оперативного вектору на рисунку 3.2. Його значення встановлюється після спостере-

ження за подальшим розвитком ситуації та використовується для утворення навчальної пари. Тому включення цього маркера до кроків методу 3.1 призвело б до змішування процедури спостереження з процедурою навчального маркування.

Вхідними даними методу є ідентифікатор процесу PID, базові та змінні параметри його виконання, відомості про утримувані й запитувані ресурси, а також поточна множина сигнатур. Множина ресурсів M_R задається в архітектурі згідно формули (2.1), а для побудови сигнатури використовується її представлення у вигляді множини ідентифікаторів R_e, визначене у формулі (2.5). Поточна множина сигнатур активних процесів A_t формується відповідно до формули (2.9).

Результатом застосування методу є сформована або актуалізована сигнатура поточного процесу $a(t)$, структура якої визначається сукупністю компонентів $a_1, \dots, a_z$ відповідно до формул (2.6)–(2.7), а також оновлена множина сигнатур A_t. Сформована множина використовується модулем подання стану комп'ютерної системи як вхідні дані для визначення її поточного стану w_t згідно з формулою (2.10).

Операція оновлення виконується так, щоб після завершення методу в A_t залишався один актуальний запис для кожного активного процесу. Коректність формування множини A_t є необхідною умовою достовірного подання поточного стану комп'ютерної системи. Відповідно до формули (2.9) множина A_t утворюється із сигнатур активних процесів, а згідно з формулою (2.10) використовується для формування стану w_t. Тому дублювання сигнатур, пропуск окремих процесів або несвоєчасне оновлення їх параметрів призводять до спотворення формалізованого подання поточного стану комп'ютерної системи.

Задамо процес побудови сигнатури процесу такими кроками.

Крок 1. Отримання даних про процес. На вхід методу подаємо PID процесу та системна подія, що стала підставою для актуалізації його стану. За PID однозначно визначається процес у поточному зрізі КС, після чого з платформозалежного адаптера отримуються доступні ідентифікаційні та ресурсні відомості.

Крок 2. Перевірка наявності сигнатури процесу. Визначаємо поточншу

множину сигнатур A_t, подана в формулі (2.9), і перевіряємо наявність у ній запису для процесу з отриманим PID. Якщо відповідної сигнатури немає, то процес вважається новим для поточного зрізу і виконується перехід до кроку 3. Якщо сигнатура вже існує, завантажується її поточний запис, після чого виконується перехід до кроку 4. Перевірка наявності процесу в A_t запобігає дублюванню записів і забезпечує часову узгодженість множини сигнатур, з якої надалі формується стан КС w_t за формулою (2.10). Для вже відомого процесу незмінні ідентифікаційні характеристики зберігаються, а параметри, що можуть змінюватися під час виконання, актуалізуються за новими системними даними.

Крок 3. Формування ідентифікаційної складової. Для нового процесу формуємо ідентифікаційну складову сигнатури: у позиції $a_1, \dots, a_k$ за моделлю з формули (2.6) заносяться PID, PPID, UID або відповідний ідентифікатор контексту користувача, ознака системного процесу, час перебування в системі, пріоритет та інші атрибути, передбачені прийнятим для конкретної ОС відображенням. Після ініціалізації запис переходить до спільної процедури формування ресурсної складової.

Крок 4. Формування ресурсної складової. Для нового або вже наявного процесу визначаємо актуальну множину ресурсів, які він утримує. Ідентифікатори цих ресурсів повинні належати множині R_e, яку визначено у формулі (2.5), і вони заносяться до позицій $a_{k+1}, \dots, a_{z-1}$ сигнатури відповідно до структури з формули (2.6). Якщо склад утримуваних ресурсів змінився, то попередні значення замінюються поточними.

Крок 5. Визначення запитуваного ресурсу. Перевіряємо наявність невиконаного запиту на додатковий ресурс. Відповідно до формули (2.7), якщо процес не очікує ресурс, то позиція a_z набуває нульового значення. Якщо очікування зафіксовано, то у цю позицію записується ідентифікатор запитуваного ресурсу з множини R_e. Це дозволяє відокремити факт утримання ресурсу від факту його очікування.

Крок 6. Формування та оновлення сигнатури. З актуалізованих

ідентифікаційної та ресурсної складових формується базова сигнатура для поточного процесу $a(t)$ за формулами (2.6)–(2.7). Для нового процесу вона додається до множини A_t, а для вже відомого процесу замінює попередній запис. Оновлення множини виконується відповідно до її подання у формулі (2.9).

Крок 7. Передавання результату. Метод завершує опрацювання поточної події та повертає сигнатуру $a(t)$ і оновлену множину A_t. Оновлена A_t передається модулю формування стану КС для побудови w_t за формулою (2.10).

Під час звільнення ресурсу його ідентифікатор вилучається з ресурсної складової поточної сигнатури. Якщо процес завершується, то його сигнатура більше не повинна входити до множини активних сигнатур A_t. Така операція не змінює формату сигнатури інших процесів, а лише актуалізує склад множини, з якої формується поточний стан КС. Отже, метод підтримує не тільки первинне створення вектора, але й його узгоджене супроводження протягом усього часу перебування процесу в системі.

Графічне подання послідовності кроків методу наведено на рисунку 3.3. На схемі показано два початкові сценарії – створення сигнатури нового процесу та завантаження вже наявного запису, після чого обидві гілки об’єднуються у спільну процедуру актуалізації утримуваних і запитуваних ресурсів. Така побудова забезпечує однакове формування ресурсної складової незалежно від того, чи процес спостерігається вперше, чи його сигнатура лише оновлюється.

Результатом виконання методу є базова сигнатура процесу в момент часу спостереження, яка має фіксовану семантичну структуру. Її ідентифікаційна частина встановлює, який процес і в якому контексті розглядається, а ресурсна частина відображає ресурси, що вже утримуються процесом, і поточний запит на додатковий ресурс. Якщо невиконаного запиту немає, то позиція a_z дорівнює нулю; за наявності очікування вона містить ідентифікатор відповідного ресурсу.

Оновлена множина A_t після виконання методу є узгодженим поточним зрізом сигнатур активних процесів і слугує вхідними даними для формування стану комп’ютерної системи за формулою (2.10). Таким чином, метод 3.1 виконує лише

фіксацію та уніфікацію стану процесу і його ресурсної взаємодії.

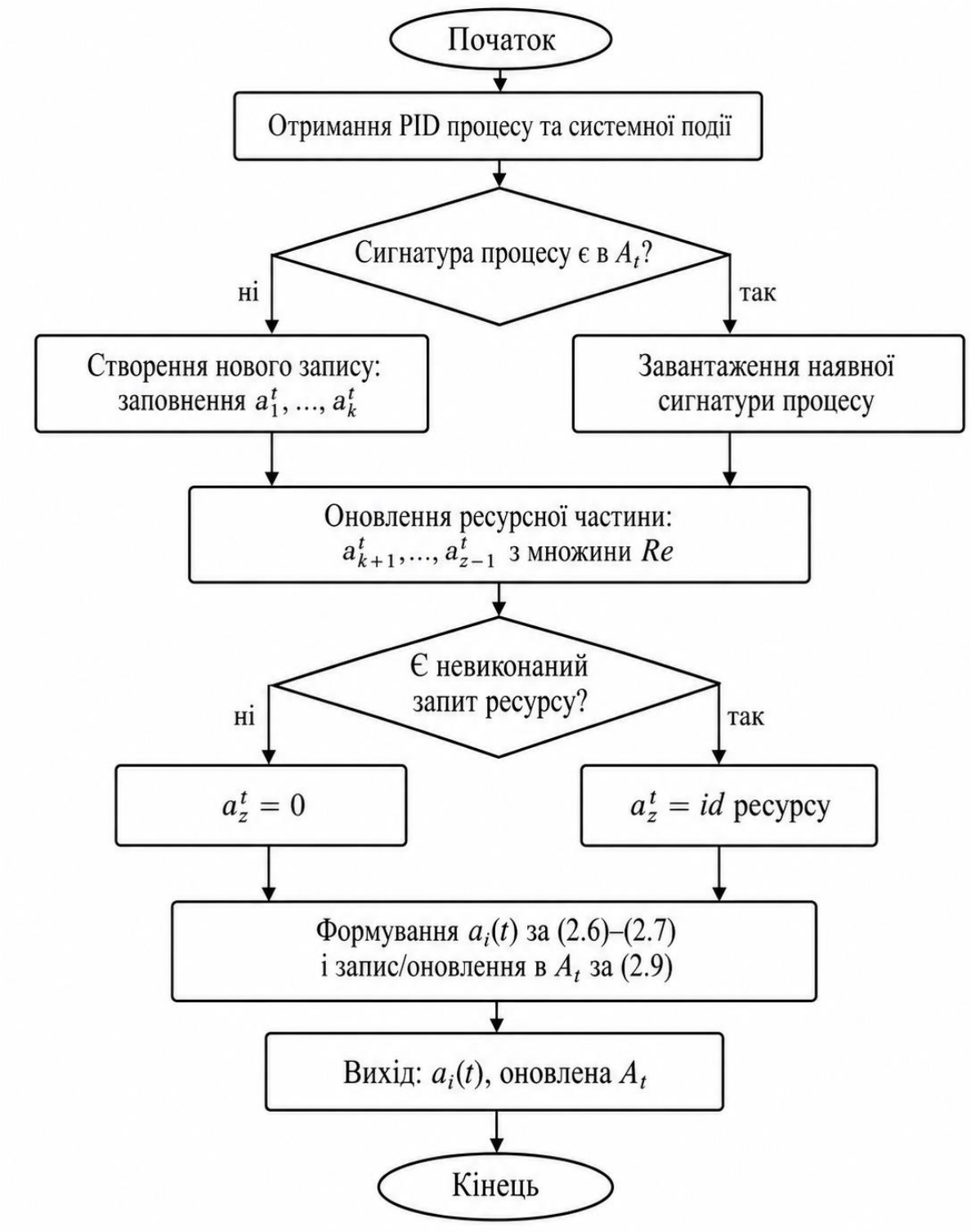


Рисунок 3.3 – Алгоритм побудови сигнатури процесу

На виході методу формується два взаємопов’язані результати різного рівня. Сигнатура $a(t)$ описує один процес, тоді як A_t відображає сукупність актуальних сигнатур усіх активних процесів. Це розмежування є суттєвим: метод 3.1 працює

локально з конкретним процесом, але оновлює структуру, яка далі використовується для подання глобального стану КС. Саме тому ідентифікаційна частина необхідна навіть тоді, коли в подальшій кластеризації головну роль відіграють ресурсні ознаки.

Для пояснення практичної роботи методу розглянемо приклад функціонування системи в реальному середовищі ОС. Нехай у комп'ютерній системі одночасно виконуються кілька користувацьких і службових процесів. Частина з них працює у звичайному режимі, частина звертається до файлів, об'єктів синхронізації або ділянок пам'яті, а окремі процеси очікують ресурси, уже зайняті іншими процесами.

У момент часу t_1 модуль побудови сигнатур отримує через платформозалежний адаптер інформацію про активний процес P_1 з PID = 1240. Для нього фіксуються ідентифікаційні параметри, передбачені моделлю сигнатури, а також ресурсні відомості. Нехай процес утримує ресурс r_2, наприклад файловий дескриптор, і ресурс r_5, наприклад об'єкт синхронізації.

Оскільки процес P_1 уперше з'явився в поточному зрізі системи, метод перевіряє множину A_t за формулою (2.9) і не знаходить запису з таким PID. Тому створюється нова сигнатура. Заповнюється її ідентифікаційна частина, після чого до ресурсної частини заносяться r_2 і r_5. Якщо на цей момент процес не очікує додаткового ресурсу, то позиція az набуває нульового значення. У скороченому вигляді, без показу невикористаних позицій, запис можна подати так:

$$a_1(t_1) = \{1240, \text{PPID}, \text{UID}, \text{priority}, \text{r}_2, \text{r}_5, 0\} \tag{3.1}$$

Цей запис означає, що процес P_1 ідентифікований у системі, утримує ресурси r_2 і r_5 та не має невиконаного запиту на інший ресурс. Після формування сигнатура додається до множини A_t, яка відображає поточний набір сигнатур активних процесів у момент t_1.

У наступний момент часу t_2 той самий процес продовжує виконання, але звертається до ресурсу r_7, який уже зайнятий іншим процесом P_2. Метод знову отримує PID = 1240 і перевіряє A_t. Оскільки запис для цього PID уже існує, то нова

сигнатура не створюється. При цьому актуалізуються змінні атрибути, перелік утримуваних ресурсів і позиція запитуваного ресурсу.

У результаті позиція a_z вже не дорівнює нулю, а містить ідентифікатор запитуваного ресурсу r_7. У скороченому поданні оновлена сигнатура має вигляд:

$$a_1(t_2) = \{1240, \text{PPID}, \text{UID}, \text{priority}, r_2, r_5, r_7\} \tag{3.2}$$

Зміст такої сигнатури полягає в тому, що процес P_1 утримує ресурси r_2 і r_5 та одночасно очікує ресурс r_7. Паралельно в системі може бути зафіксований процес P_2, наприклад із PID = 1384. Якщо він утримує ресурс r_7, але сам очікує ресурс r_2, його сигнатура у скороченому поданні може мати вигляд:

$$a_2(t_2) = \{1384, \text{PPID}, \text{UID}, \text{priority}, \ r_7, \ r_2\} \tag{3.3}$$

У такому випадку множина A_t у момент часу t_2 уже відображає не ізольовані параметри процесів, а їхню взаємопов'язану ресурсну конфігурацію. Завдання методу 3.1 полягає у коректному й однаковому для всіх процесів формуванні цих записів: визначенні процесу, актуалізації утримуваних ресурсів і фіксації запитуваного ресурсу.

Виходячи з архітектури, заданої за формулою (2.1), для побудови сигнатури використовуються множина процесів ОС $M_{OS,pr}$, множина ресурсів КС M_R та множина поточного розподілу ресурсів між процесами $M_{r,OS}$.

Для довільного активного процесу $M^i_{OS,pr} \in M_{OS,pr}$ його локальний стан у момент спостереження t подамо як сукупність ідентифікаційно-поведінкових параметрів, множини утримуваних ресурсів і поточного запиту на ресурс:

$$a(t)_{M^i_{OS,pr}} = \langle (a^t_1, \dots, a^t_k), (a^t_{k+1}, \dots, a^t_{z-1}), a^t_z \rangle, \tag{3.4}$$

де $(a^t_1, \dots, a^t_k)$ є впорядкованою сукупністю параметрів, що утворюють ідентифікаційну складову; $(a^t_{k+1}, \dots, a^t_{z-1})$ – ідентифікатори ресурсів, які процес $M^i_{OS,pr}$ фактично утримує у момент t; a^t_z – ідентифікатор додатково запитуваного ресурсу або 0 за відсутності запиту.

Множину ресурсів $(a^t_{k+1}, \dots, a^t_{z-1})$ визначимо з $M_{r,OS}$ як множину тих

ресурсів $M_R^j \in M_R$, для яких у поточному розподілі існує пара "процес $M_{OS,pr}^i$ – ресурс M_R^j":

$$\{a_{k+1}^t, \dots, a_{z-1}^t\} = \{r_j \in R_e \mid \langle M_{OS,pr}^i, M_R^j \rangle \in M_{r,OS}(t)\}, \quad (3.5)$$

Метод 3.1 задає відображення F_{M_1} локального стану $a(t)_{M_{OS,pr}^i}$ у базову оперативну сигнатуру $a(t)$, компоненти якої відповідають формулі (2.6), а остання ресурсна позиція a_z визначається за формулою (2.7).

$$F_{M_1}: a(t)_{M_{OS,pr}^i} \to a(t), \quad (3.6)$$

Обов'язковою умовою однозначності є канонічне кодування ресурсної складової. Оскільки $(a_{k+1}^t, \dots, a_{z-1}^t)$ за своєю природою є множиною, то її елементи не повинні породжувати різні сигнатури лише через різний порядок переліку ресурсів. Тому перед записом до позицій $(a_{k+1}, \dots, a_{z-1})$ утримувані ресурси впорядковуються відповідно до фіксованого порядку ідентифікаторів у множині R_e, заданій формулою (2.5). За такого правила перестановка одних і тих самих ресурсів не змінює сигнатуру.

Розглянемо однозначність формування сигнатури. Нехай одному й тому самому повному локальному стану $a(t)_{M_{OS,pr}^i}$ метод поставив у відповідність дві різні сигнатури $a(t)$ та $\acute{a}(t)$. Оскільки процес $M_{OS,pr}^i$ та сукупність $(a_1^t, \dots, a_k^t)$ не змінюються, усі компоненти $a_1, \dots, a_k$, що формують ідентифікаційну складову, повинні мати однакові значення в обох сигнатурах. Множина $(a_{k+1}^t, \dots, a_{z-1}^t)$ також однакова; після застосування канонічного порядку її елементи займають ті самі позиції $a_{k+1}, \dots, a_{z-1}$. Нарешті, для однакового a_z^t компонент a_z за правилом (2.7) також однаковий. Отже, всі відповідні компоненти двох векторів збігаються, тобто $a(t) = \acute{a}(t)$, що суперечить вихідному припущенню про їх відмінність. Таким чином, за фіксованих правил отримання й кодування даних одному повному локальному стану процесу відповідає рівно одна сигнатура.

Додатково розглянемо два різні повні локальні стани $a_i(t) \neq \acute{a}_j(\mathrm{t})$, що отримали одну й ту саму сигнатуру. Якщо стани відрізняються процесом або

ідентифікаційним контекстом, то відмінність повинна бути відображена щонайменше в одній із позицій $a_1, \dots, a_k$; зокрема, у межах поточного сеансу ОС різні процеси мають різні PID. Якщо стани відрізняються множинами утримуваних ресурсів $\left(a^t_{k+1,i}, \dots, a^t_{z-1,i}\right)$ та $\left(a^t_{k+1,J}, \dot{\dots}, a^t_{z-1,J}\right)$, то через унікальність ідентифікаторів ресурсів і канонічне кодування відрізнятиметься принаймні одна позиція ресурсної складової. Якщо ж відмінність полягає у запитуваному ресурсі, то різним буде a_z. У кожному випадку одержуємо суперечність з припущенням про рівність сигнатур. Отже, за умови повного та синхронізованого спостереження і безвтратного кодування відображення локального стану процесу в сигнатуру є ін'єктивним відносно ознак, включених до моделі.

Ненадлишковість сигнатури розглянемо як відсутність семантичного дублювання та функціональну необхідність кожної збереженої ознаки для розрізнення станів, що враховуються моделлю. Ідентифікаційна та ресурсна складові виконують різні функції. Перша складова визначає об'єкт і контекст спостереження, друга – фактичні відношення "процес – ресурс". Тому одна складова не може бути відновлена з іншої без втрати інформації про стан процесу.

Необхідність ключових компонентів визначається їхнім впливом на розрізнення станів. Якщо вилучити PID або іншу ознаку, яка однозначно зв'язує сигнатуру із процесом, то два різні процеси з однаковою ресурсною поведінкою можуть стати нерозрізнюваними. Якщо вилучити відомості про утримувані ресурси, стани з однаковим запитуваним ресурсом, але різними множинами вже зайнятих ресурсів, матимуть однакове подання, хоча створюють різні ресурсні залежності. Якщо вилучити позицію a_z, неможливо відрізнити процес, який лише утримує ресурси, від того самого процесу, який одночасно утримує їх і очікує додатковий ресурс. Саме остання відмінність є принциповою для аналізу передумов циклічного очікування.

Для додаткових ідентифікаційних, часових та поведінкових параметрів $a_1, \dots, a_k$ доцільно застосовувати критерій функціональної необхідності. Ознака залишається в сигнатурі лише тоді, коли існують допустимі локальні стани, які

збігаються за іншими компонентами, але відрізняються за цією ознакою, і така відмінність має зміст для ідентифікації процесу, узгодження його часової послідовності або інтерпретації ресурсної поведінки. Якщо певний параметр повністю визначається іншими компонентами і не додає можливості розрізнення станів, його включення не відповідає цьому критерію і такий параметр має бути вилучений. За дотримання цього правила сигнатура є ненадлишковою відносно прийнятої моделі. Кожна залишена позиція має окреме семантичне призначення, а дублювання одного й того самого факту в декількох компонентах не допускається.

Розглянуте вище обґрунтування одночасно визначає межу застосовності однозначного відображення. Якщо два фактично різні локальні стани одержали однакову сигнатуру, то за прийнятих правил кодування це означає порушення принаймні однієї з вихідних умов. Частина істотних даних не була отримана, ресурс або атрибут не входив до контрольованої множини, значення було застарілим, або дві відмінні системні сутності були зведені до одного коду. Тобто збіг сигнатур у такому випадку є наслідком недостатньої інформативності вхідного опису, а не неоднозначності самого методу.

Для двох різних локальних станів a і $\acute{a}$ рівність $F(a) = F(\acute{a})$ можлива лише за порушення повноти або синхронності вхідних даних, унікальності ідентифікаторів процесів і ресурсів чи безвтратності відображення. Оскільки $a \neq \acute{a}$, стани повинні відрізнятися хоча б за одним із трьох видів інформації: ідентифікаційними параметрами, множиною утримуваних ресурсів або запитуваним ресурсом. Відповідна відмінність обов’язково змінює одну з позицій сигнатури, тому $F(a) \neq F(\acute{a})$. Отримана суперечність означає, що початкове припущення про повноту даних було хибним. Отже, однакова сигнатура двох різних локальних станів може виникнути лише тоді, коли відмінність між цими станами не потрапила до набору спостережуваних або закодованих даних.

Розглянемо приклад перевірки властивостей сигнатури. Нехай, як у прикладі роботи методу 3.1, процес P_1 з PID = 1240 утримує ресурси r_2 і r_5 та очікує ресурс r_7. Після канонічного впорядкування ресурсів його скорочена сигнатура матиме

вигляд $\langle 1240, \dots, r_2, r_5, r_7 \rangle$. Для іншого процесу P_2 з PID = 1384 навіть за такої самої ресурсної конфігурації перша ідентифікаційна позиція відрізнятиметься, тому сигнатури не збігаються. Для того самого P_1 стан $\langle r_2, r_5;$ запит відсутній$\rangle$ і стан $\langle r_2, r_5;$ очікується $r_7\rangle$ також не можуть мати одну сигнатуру, оскільки в першому випадку $a_z = 0$, а в другому $a_z = r_7$.

Водночас припустимо, що процес P_1 додатково утримує ресурс r_9, але відомості про r_9 не надходять до модуля побудови сигнатур або цей ресурс відсутній у контрольованій множині R_e. Тоді ресурсні стани $\left(a^t_{k+1,i}, \dots, a^t_{z-1,i}\right) = (r_2, r_5)$ і $\left(a^t_{k+1,i}, \dots, a^t_{z-1,i}\right) = (r_2, r_5, r_9)$ можуть бути відображені однаково, оскільки відмінність r_9 втрачена ще до формування вектора. Аналогічно, якщо подія запиту r_7 не була зафіксована, то стан до запиту та стан очікування можуть обидва отримати $a_z = 0$. Такі приклади не спростовують однозначність методу. Вони показують, що її необхідною умовою є повнота контрольованих ресурсів і своєчасне отримання системних подій.

Розглянемо урахування часового дрейфу під час формування сигнатури. Однозначність сигнатури передбачає не лише повноту, а й часову узгодженість її компонентів. На практиці параметри процесу, перелік утримуваних ресурсів і поточний запит можуть зчитуватися не абсолютно одночасно. Нехай t_{start} є моментом початку формування сигнатури, t_{end} – моментом завершення, а фактичний час формування становить $\tau_f = t_{end} - t_{start}$. Позначимо через τ_{std} допустимий стандартний час формування сигнатури, встановлений для реалізації методу. Оскільки модель другого розділу використовує інтервал формування стану Δt, практично доцільно вимагати, щоб τ_{std} не перевищував Δt і був істотно меншим за нього, щоб одна сигнатура залишалася локальним часовим зрізом, а не сумішшю декількох послідовних станів.

$$\tau_f = t_{end} - t_{start}, \tau_f \le \tau_{std} \le \Delta \text{t}. \tag{3.7}$$

Для більш точної оцінки часового дрейфу можна врахувати моменти отримання окремих компонентів $t_1, \dots, t_z$ і визначити внутрішній часовий розкид

сигнатури як $\delta_t = \max(t_j) - \min(t_j)$. За ідеального синхронного знімка δ_t прямує до нуля. Якщо ж τ_f або δ_t перевищують допустиму межу, зростає ймовірність того, що різні частини вектора належать уже до різних фактичних станів процесу.

Наприклад, на початку збирання даних процес P_1 може утримувати r_2 і r_5. Поки модуль послідовно читає інші параметри, процес звільняє r_5, а потім формує запит на r_7. Якщо перелік утримуваних ресурсів був зчитаний до звільнення r_5, а запит – після його звільнення, побудована сигнатура міститиме одночасно r_5 і r_7. Така комбінація може не відповідати жодному реальному моменту роботи КС. У прогнозуванні це особливо небезпечно, оскільки штучно поєднані факти утримання й очікування здатні створити ознаки ресурсного конфлікту, якого фактично не було, або, навпаки, приховати реальну небезпечну залежність.

Тому при $\tau_f > \tau_{std}$ сформовану сигнатуру доцільно вважати часово неузгодженою та не передавати без перевірки до множини A_t. Практична реалізація методу може використовувати часову мітку або номер версії стану, прив'язаний до системної події, що ініціювала оновлення. Після збирання параметрів необхідно перевірити, чи не відбулися протягом цього інтервалу зміни в розподілі ресурсів $M_{r,OS}$ для поточного процесу. Якщо версія або послідовність подій змінилася, знімок відкидається і формується повторно для нового актуального моменту. Альтернативою є побудова сигнатури безпосередньо з упорядкованого потоку системних подій, у якому кожна подія атомарно актуалізує лише відповідну частину вже наявної сигнатури.

Отже, часовий дрейф не змінює математичне правило відображення F_{M_1}, але може порушити його вихідну передумову – належність усіх компонентів одному стану $a(t)_{M^i_{OS,pr}}$. Якщо ця умова порушена, то метод отримує не повний узгоджений стан, а комбінацію даних з різних моментів, що еквівалентно недостатній або некоректній інформації для побудови достовірної сигнатури. Контроль τ_f, δ_t і зміни $M_{r,OS}$ упродовж збирання даних забезпечує збереження однозначності сигнатури та її відповідності поточному часовому зрізу КС.

Таким чином, формування сигнатури методом 3.1 є однозначним та ін’єктивним відносно локальних станів процесу за одночасного виконання чотирьох умов: унікальної ідентифікації процесів і ресурсів, повноти контрольованих даних, канонічного кодування ресурсної складової та часової узгодженості зібраних параметрів. Ненадлишковість забезпечується функціональним розмежуванням компонентів і вилученням ознак, які не додають здатності розрізняти стани. Збіг сигнатур різних локальних станів або поява сигнатури, що не відповідає жодному фактичному стану, є діагностичною ознакою неповноти, втрати або часової неузгодженості вхідних даних і має бути підставою для повторного формування запису.

Отже, розроблений метод побудови сигнатури процесу реалізує метод M_1 з формули (2.2) та функцію F_{M_1} з формули (2.4), забезпечуючи перетворення платформозалежних системних даних на інваріантний вектор $a(t)$. У векторі за формулами (2.6)–(2.7) розмежовано ідентифікаційну та ресурсну складові, а навчальний маркер a_{z+1} за формулою (2.8) винесено за межі оперативної процедури. Сформована сигнатура включається до множини A_t за формулою (2.9), яка використовується для подання стану w_t за формулою (2.10). Таким чином, метод формує узгоджений та інформативний опис процесу, придатний для подальшої кластеризації, нечіткого аналізу та прогнозування стану взаємоблокування процесів.

3.2 Метод побудови бази нечітких правил

В розробленій архітектурі системи прогнозування стану взаємоблокування процесів метод побудови бази нечітких правил є безпосередньою алгоритмічною реалізацією методу M_2 у формулі (2.2), а у складі функції прогнозу – функції F_{M_2} у формулі (2.4). На рівні функціональних компонентів архітектури згідно з формулою (2.1) він охоплює послідовну роботу M_{S_3}MS3 – модуля навчання, M_{S_4} – модуля нечіткої кластеризації та M_{S_5} – модуля побудови бази правил. Результатом

є база правил M_{S_6}. Отже, метод 3.2 конкретизує передбачений архітектурою перехід від накопичених сигнатур і станів КС до сформованої бази нечітких правил.

У загальній послідовності прогнозування за (2.2) метод M_2 виконується після методу M_1 і формує інформаційний результат, необхідний для методу M_3. Аналогічно у формулі (2.4) функція F_{M_2} опрацьовує результати F_{M_1} та формує базу правил для подальшого використання F_{M_3}. Виділення цього етапу в окремий метод необхідне тому, що архітектура визначає склад компонентів та їх зв'язки, але не задає алгоритмічної послідовності підготовки навчальних даних, кластеризації, оцінювання кластерів і генерації правил.

Метод побудови бази нечітких правил призначений для формування набору правил підсистеми логічного висновку на основі накопичених станів комп'ютерної системи та ресурсної поведінки процесів. Результатом роботи цього методу є база правил, що відображає типові ресурсні стани процесів та їх зв'язок із переходом до взаємоблокування в навчальних даних.

Математичну основу методу сформовано так: стани КС, ресурсну сигнатуру, прогнозний маркер і навчальну пару визначено за формулами (2.11)–(2.14); формування вибірок і нормалізацію сигнатур – (2.15)–(2.20); матрицю ознак і FCM-кластеризацію – (2.21)–(2.27); оцінювання ризику та формування нечітких правил – (2.28)–(2.30). Ці співвідношення конкретизовано послідовністю операцій методу.

Вхідними даними методу є множина станів КС за період спостереження W, множина сигнатур процесів, яка отримана після виконання методу 3.1, множина ідентифікаторів ресурсів R_e, прогнозний горизонт Δt_p, а також параметри FCM-кластеризації. Множина ресурсів узгоджується з архітектурним поданням (2.1), множина активних сигнатур у момент часу – з (2.9), стан КС – з (2.10), а часова послідовність станів – з (2.11). Особливістю методу є те, що прогнозний маркер $y_i(t)$ формується тільки для навчальної вибірки і показує, чи призвів зафіксований у момент часу t ресурсний стан процесу до взаємоблокування протягом наступного інтервалу Δt_p відповідно до формули (2.13).

Навчальна пара є основною одиницею даних, що пов'язує спостережувану

ресурсну конфігурацію з фактичним наслідком її подальшого розвитку. У моделі з параграфа 2.4 вона задається формулою (2.14) як поєднання ресурсної сигнатури процесу за (2.12) і прогнозного маркера за (2.13). Ресурсна частина містить відомості про ресурси, які процес уже утримує, та ресурс, який він запитує, а маркер не описує поточний стан і не є вхідною ознакою FCM: він фіксує відомий для навчання наслідок, що настав у межах заданого прогнозного горизонту.

Перед кластеризацією ознакова частина навчальної пари нормалізується. У межах розробленої моделі нормалізація має структурно-кодовий характер, тобто усі ресурси зіставляються з єдиним упорядкуванням множини R_e, утримання ресурсів кодується бінарними компонентами за формулою (2.17), а запитуваний ресурс – окремим бінарним блоком за формулою (2.18). У результаті формується нормалізована (внормована) сигнатура $\tilde{a}_i(t)$ за формулою (2.19), розмірність якої є сталою для всіх процесів відповідно до формули (2.20). Такий спосіб не змішує PID, часову мітку або прогнозний маркер із простором ресурсних ознак і водночас зберігає відмінність між володінням ресурсом та очікуванням його надання.

Отже, формально навчальна пара утворюється до нормалізації за формулою (2.14), а для обчислювальної реалізації її ознакова частина замінюється еквівалентним векторним поданням $\tilde{a}_i(t)$. Тому підготовлену навчальну пару доцільно розглядати у вигляді $\langle \tilde{a}_i(t), y_i(t) \rangle$: нормалізована сигнатура надходить до матриці ознак X, тоді як $y_i(t)$ зберігається окремо і використовується лише після кластеризації для визначення небезпечності сформованих кластерів.

Структурне подання підготовленої навчальної пари задамо так (рисунок 3.4):

Ознакова частина нормалізована сигнатура $\tilde{a}_i(t)$ (2.17)–(2.20)	Цільова частина прогнозний маркер $y_i(t)$ (2.13)
Підготовлена навчальна пара: $\langle \tilde{a}_i(t), y_i(t) \rangle \rightarrow$ X + окремий вектор маркерів	

Рисунок 3.4 – Підготовлена навчальна пара

Таке розділення є принциповим для коректної роботи методу. Якщо включити $y_i(t)$ до матриці X, алгоритм FCM групуватиме приклади з урахуванням уже відомого результату і тим самим перестане виявляти природну структуру

ресурсних станів. Збереження маркера поза матрицею дає змогу спочатку виділити типові конфігурації процесів і ресурсів, а вже потім оцінити, наскільки часто кожна з них передувала взаємоблокуванню.

Вихідними даними методу є база нечітких правил B. До проміжних результатів належать модифікована множина навчальних сигнатур, навчальні пари, нормалізовані ресурсні вектори сталої розмірності, матриця X для FCM, центри кластерів, ступені належності сигнатур до кластерів та оцінки ризику кластерів. Представимо процес побудови бази нечітких правил такими кроками.

Крок 1. Формування часової послідовності станів. На вхід методу подаємо множину станів комп'ютерної системи W, яку сформовано за період спостереження. Вона відповідає формулі (2.11), кожний її елемент є станом КС у конкретний момент часу згідно формули (2.10) і містить множину активних сигнатур за формулою (2.9).

На цьому кроці зберігається не лише склад сигнатур, а й їх прив'язка до моментів часу. Без часової впорядкованості неможливо встановити, чи була конкретна ресурсна конфігурація попередником взаємоблокування. Час використовується для формування навчального зв'язку між станом і його наслідком, але не включається як числова ознака до FCM.

Крок 2. Підготовка навчальних записів. На основі W формуємо модифіковану множину сигнатур, придатну для навчання. Для кожного запису зберігаються PID, момент часу t, ресурсна частина та службовий зв'язок із вихідним станом КС. Перевіряється коректність ідентифікаторів та можливість однозначно зіставити ресурси з множиною R_e, визначеною в архітектурі системи.

На цьому етапі відокремлюються ідентифікаційні й часові дані, потрібні для відновлення зв'язку з конкретним процесом, від ресурсних ознак, які надалі братимуть участь у кластеризації. Це не змінює змісту вихідної сигнатури, сформованої методом 3.1, а лише готує її до навчального опрацювання.

Крок 3. Вибір неопрацьованої сигнатури. Перевіряється наявність неопрацьованих сигнатур у підготовленій множині. Якщо така сигнатура є, вона оби-

рається як поточна і виконується перехід до кроку 4. Якщо всі записи опрацьовано, здійснюється перехід до кроку 9, де формується матриця X для FCM-кластеризації.

Кожна сигнатура розглядається один раз у контексті свого моменту часу та прогнозного горизонту Δt_p. Повторне включення того самого запису без нової системної події штучно збільшило б вагу відповідного ресурсного стану у навчальній вибірці.

Крок 4. Виділення ресурсної сигнатури. Для поточного процесу визначаємо ресурсну сигнатуру відповідно до формули (2.12). До неї входять ідентифікатори ресурсів, які процес утримує в момент t, та ідентифікатор ресурсу, який він запитує. Всі ідентифікатори перевіряються на належність множині R_e. Некоректний або невідображуваний ресурс не може бути безпосередньо включений до навчального вектора, оскільки це порушило б сталість простору ознак.

Крок 5. Визначення прогнозного маркера. Для тієї самої сигнатури визначаємо маркер $y_i(t)$ відповідно до формули (2.13). Аналізується подальша послідовність станів цього процесу на інтервалі від t до $t + \Delta t_p$. Якщо протягом прогнозного горизонту зафіксовано перехід у стан взаємоблокування, то приймається $y_i(t) = 1$. Якщо такого переходу не зафіксовано, то $y_i(t) = 0$.

Маркер не включається до нормалізованої ресурсної сигнатури і не є координатою матриці X. Його роль полягає в оцінюванні кластерів після завершення FCM, тобто спочатку виявляється структура ресурсних станів, а потім ця структура інтерпретується через відомі навчальні наслідки.

Крок 6. Формування навчальної пари. Ресурсна сигнатура і визначений прогнозний маркер об’єднуються в навчальну пару відповідно до формули (2.14). Для одного моменту часу сукупність таких пар утворює навчальну вибірку за формулою (2.15), а послідовність вибірок за всіма станами W формує загальну навчальну вибірку за формулою (2.16).

Пара зберігає часовий причинно-наслідковий зв’язок Її ознакова частина

описує стан процесу до можливого взаємоблокування, а маркер характеризує фактичний результат подальшої ресурсної взаємодії. Саме це дозволяє після кластеризації визначати небезпечність не окремого наперед заданого шаблону, а групи схожих ресурсних станів.

Крок 7. Нормалізація сигнатури. Ознакова частина навчальної пари перетворюється на нормалізовану сигнатуру сталої розмірності. Утримувані ресурси кодуються відповідно до формули (2.17), запитувані – відповідно до формули (2.18), після чого формується вектор $\tilde{a}_i(t)$ за формулою (2.19). Його розмірність для всіх процесів є однаковою згідно з формулою (2.20).

Перший блок компонентів нормалізованої сигнатури відображає факт утримання кожного ресурсу з R_e, другий – факт запиту відповідного ресурсу. Тому процеси з різною кількістю фактично зайнятих об’єктів подаються в одному просторі ознак, а однакові набори утримуваних ресурсів із різними запитами не ототожнюються. PID і час зберігаються лише як службова прив’язка до вихідного запису.

Крок 8. Формування навчальної вибірки. Нормалізована сигнатура $\tilde{a}_i(t)$ зберігається разом із відповідним маркером $y_i(t)$ як підготовлене представлення навчальної пари. Після додавання запис позначається як опрацьований, і метод повертається до кроку 3. Для кожного елемента зберігається відповідність між нормалізованим вектором, вихідною сигнатурою, PID, моментом часу та прогнозним маркером.

Збереження цієї відповідності необхідне для змістовної інтерпретації кластерів: матриця FCM містить лише числові вектори, але після навчання потрібно встановити, які конкретні ресурсні конфігурації та процеси сформували той або інший типовий стан.

Крок 9. Формування матриці ознак. Після опрацювання всіх придатних сигнатур з нормалізованих векторів формується матриця X відповідно до формули (2.21). Її розмірність визначається за формулою (2.22), оскільки кожний рядок має однакову кількість ресурсних ознак. До X включається лише ознакова частина

навчальних пар; час і $y_i(t)$ зберігаються окремо.

Крок 10. Нечітка кластеризація Fuzzy C-Means. Матриця X подається на FCM-кластеризацію з кількістю кластерів $c = 4$. Результат кластеризації подається відповідно до формули (2.23) через центри v_j та ступені належності u_{ij}. На початку задаються початкові центри або початкова матриця належності. Далі для кожного навчального вектора обчислюються відстані до кластерних центрів за евклідовою нормою з формули (2.25).

На основі отриманих відстаней уточнюються ступені належності u_{ij}. Чим ближче $\tilde{a}_i(t)$ до центра v_j порівняно з іншими центрами, то тим більшим є його ступінь належності до j-го кластера. Після цього центри перераховуються як зважені середні навчальних векторів відповідно до формули (2.26), де вагами виступають ступені належності з урахуванням параметра нечіткості. Ітерації "відстані – належності – центри" повторюються до виконання критерію зупинки, тобто до стабілізації кластерних центрів і цільової функції FCM згідно з формулою (2.24).

Застосування нечіткої кластеризації принципово важливе для досліджуваної задачі, оскільки між звичайним очікуванням ресурсу, граничним станом і взаємоблокуванням немає різкої межі. FCM не виконує жорсткого віднесення сигнатури лише до одного класу. Один навчальний вектор може одночасно мати ненульові ступені належності до кількох кластерів. Кількість кластерів задано рівною чотирьом відповідно до прийнятої в моделі системи станів. Їх змістова інтерпретація уточнюється за центрами та прогнозними маркерами після завершення кластеризації.

Ступені належності u_{ij}, які обчислює FCM, слід відрізняти від функцій належності, що зберігаються в нечітких правилах. Для компактного подання передумов правил у межах методу використовуються трапецієподібні функції належності. Центральна ділянка трапеції відповідає найбільш характерній для кластера області нормалізованих ознак, а бічні ділянки описують поступовий перехід до суміжних кластерних станів. Параметри трапецій визначаються за

положенням центра v_j та розподілом навчальних векторів із найбільшими u_{ij}. Таким чином, трапецієподібне подання деталізує реалізацію передумови правила з формули (2.29), не вводячи нового компонента до навчальної пари.

Крок 11. Оцінювання ризику кластерів. Для кожного сформованого кластера обчислюється ризик P_j переходу до взаємоблокування за формулою (2.28). Це зважена частка навчальних прикладів із маркером $y_i(t) = 1$, причому вагами є ступені належності u_{ij}. Тому вектори, розташовані ближче до центра кластера, мають більший вплив на оцінку його небезпечності, ніж приклади на межі між кластерами.

Якщо P_j наближається до нуля, кластер описує переважно безпечні ресурсні стани; якщо до одиниці – ресурсні конфігурації, які часто передували або відповідали взаємоблокуванню. Проміжні значення характеризують стани з підвищеним, але не однозначним ризиком. До отримання P_j номер кластера не повинен трактуватися як наперед заданий клас небезпеки. Семантика встановлюється за поєднанням структури центра і фактичних навчальних наслідків.

Крок 12. Генерування бази нечітких правил. На основі центрів v_j, ступенів належності u_{ij}, параметрів трапецієподібних функцій та оцінених ризиків P_j формуються правила типу "якщо–то". Формат одного правила визначається за формулою (2.29). Передумова описує належність нормалізованої сигнатури до області відповідного кластерного стану, а висновок містить пов’язану з цим кластером оцінку ризику.

Сукупність сформованих правил утворює базу B відповідно до формули (2.30). Для прийнятих чотирьох кластерів база містить чотири узагальнені правила, що відображають послідовність станів від нормальної ресурсної поведінки до взаємоблокування. Після генерації правила зберігаються разом із параметрами функцій належ-ності та оцінками ризику, і метод завершує роботу.

Зобразимо кроки методу побудови бази нечітких правил графічно (рис. 3.5).

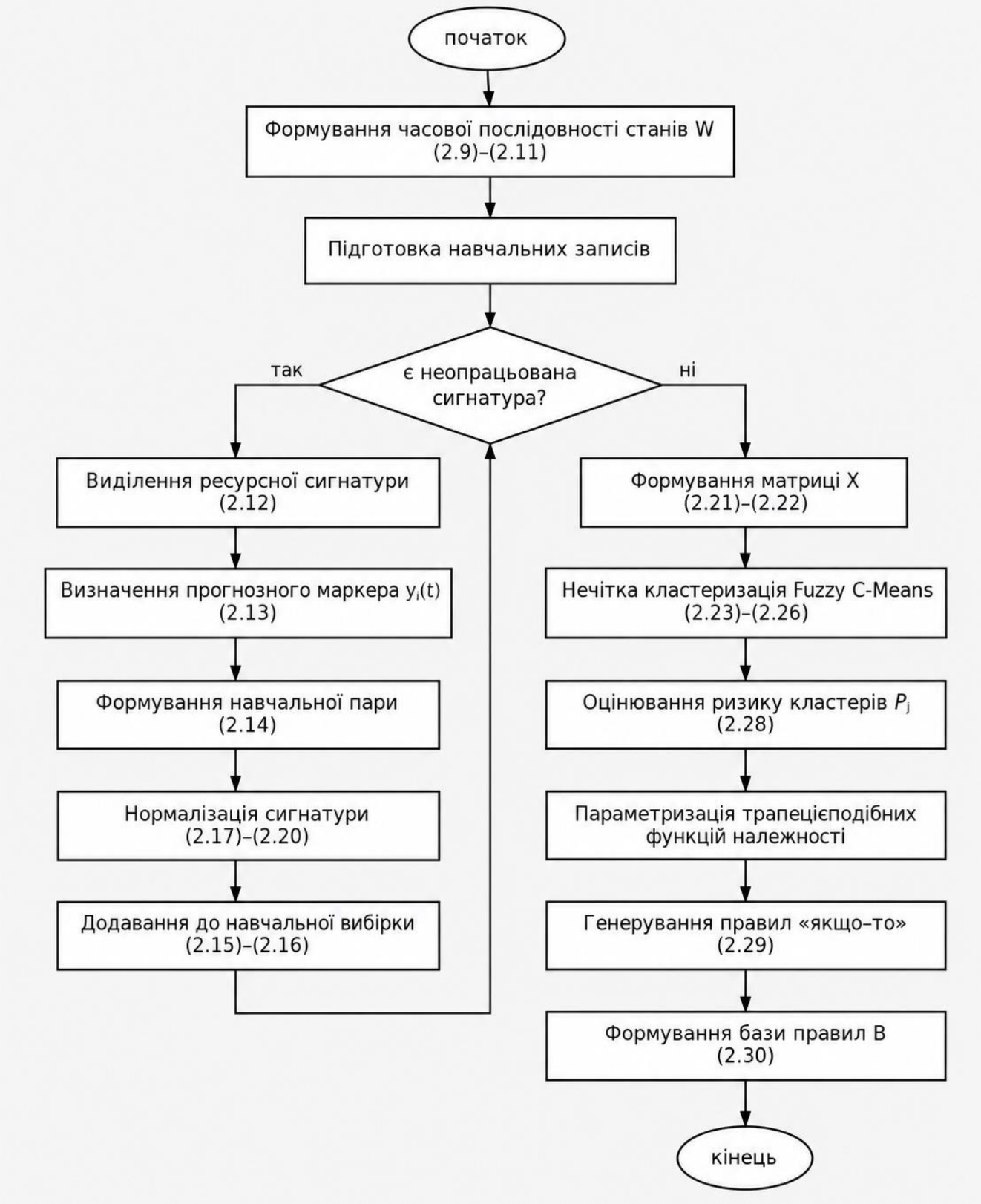


Рисунок 3.5 – Алгоритм побудови бази нечітких правил

Результатом виконання методу є база нечітких правил B, яка сформована відповідно до формули (2.30). Вона містить набір правил, кожне з яких пов’язане з одним із кластерів ресурсних станів, отриманих методом FCM. Така база не описує

окремий процес і не зберігає всі навчальні сигнатури, а узагальнює типові комбінації утримуваних і запитуваних ресурсів, що спостерігалися в історичних даних.

Кожне правило має дві змістові частини. Передумова правила визначає належність нормалізованої сигнатури до області певного кластера і реалізується відповідно до формули (2.29) з використанням параметризованої трапецієподібної функції належності. Висновок правила містить оцінку ризику P_j, обчислену за формулою (2.28). Тому база правил поєднує структуру ресурсного простору, отриману через FCM, і навчальну інформацію про те, які ресурсні стани в подальшому переходили до взаємоблокування.

До проміжних результатів методу належать ресурсні сигнатури за формулою (2.12), прогнозні маркери за формулою (2.13), навчальні пари за формулою (2.14), навчальні вибірки за формулами (2.15) і (2.16), нормалізовані сигнатури за формулою (2.17)–(2.20), матриця X за формулою (2.21) і (2.22), центри та ступені належності кластерів за формулами (2.23)–(2.26), а також оцінки ризику за формулою (2.28). Ці результати необхідні для побудови B, але самостійно не є кінцевим результатом застосування методу.

Отримана база правил може використовуватися багаторазово без повторного виконання FCM для кожної нової системної події. Це зменшує обчислювальне навантаження під час подальшого аналізу поточних станів і відповідає архітектурному розділенню ресурсомісткого навчання та застосування вже сформованих правил. За зміни складу ресурсів, характеру навантаження або політик доступу база B може бути повторно сформована на основі оновленої множини W.

Таким чином, метод побудови бази нечітких правил виконує перехід від накопичених сигнатур процесів до компактної та інтерпретованої бази знань. Він не приймає рішення про блокування чи завершення процесу і не визначає поточний прогноз для окремого PID; його призначення полягає у формуванні правил, які надалі можуть застосовуватися для нечіткого оцінювання нових або оновлених

ресурсних станів.

Розглянемо приклад функціонування запропонованого методу під час навчання системи прогнозування взаємоблокувань. Припустимо, що протягом періоду спостереження модуль побудови сигнатур (метод 3.1) сформував часово впорядковану множину станів W із сигнатурами активних процесів.

Нехай у момент часу t_1 у системі працювали три процеси: процес бази даних P_1, вебсервер P_2 та процес резервного копіювання P_3. Для них були сформовані сигнатури:

$a_1 = \{1240, r_2, r_5, 0\}$;

$a_2 = \{1384, r_7, 0\}$;

$a_3 = \{1512, r_1, r_4, 0\}$.

У цей момент жоден із процесів не перебував у стані взаємоблокування, тому після аналізу подальших станів у межах прогнозного горизонту для відповідних навчальних прикладів формується маркер $y = 0$.

У наступному стані процес P_1 продовжує утримувати ресурси r_2 та r_5, але вже очікує ресурс r_7:

$a_1 = \{1240, r_2, r_5, r_7\}$.

Одночасно процес P_2 утримує ресурс r_7 і очікує ресурс r_2:

$a_2 = \{1384, r_7, r_2\}$.

Після аналізу наступних станів системи встановлено, що в межах Δt_p обидва процеси перейшли у взаємоблокування. Відповідно для цих навчальних прикладів формується прогнозний маркер $y = 1$.

Таким чином, навчальна вибірка містить як приклади безпечної ресурсної поведінки, так і конфігурації, що передували взаємоблокуванню. Далі кожна ресурсна сигнатура нормалізується відповідно до формул (2.17)–(2.20). Якщо множина R_e містить сім ресурсів $r_1, \dots, r_7$, то для процесу P_1, який утримує r_2 і r_5 та запитує r_7, нормалізована сигнатура складається з двох семикомпонентних блоків:

$$\tilde{a}_1 = [0,1,0,0,1,0,0 \mid 0,0,0,0,0,0,1]$$

Перший блок відображає утримувані ресурси, другий – запитуваний ресурс.

Аналогічно формуються нормалізовані вектори для інших сигнатур. Сукупність таких векторів утворює матрицю X за формулою (2.21), яка подається на Fuzzy C-Means.

Під час FCM усі навчальні вектори отримують ступені належності до чотирьох кластерів. Після завершення ітерацій центри кластерів і склад найбільш характерних для них прикладів аналізуються разом із маркерами y. Наприклад, один кластер може об'єднувати переважно звичайні ресурсні конфігурації, інший – стани очікування, третій – граничні комбінації ресурсних залежностей, а кластер із найбільшою зваженою часткою $y = 1$ – конфігурації, що найчастіше передували взаємоблокуванню.

Припустимо, що для кластера, інтерпретованого як найбільш небезпечний, за формулою (2.28) отримано значення ризику $P_4 = 0{,}93$.

Це означає, що зважена за ступенями належності частка навчальних сигнатур цього кластера, для яких у подальшому було зафіксовано взаємоблокування, становить 0,93. Для кластера нормальної роботи аналогічний показник, наприклад, може становити $P_1 = 0{,}04$, тобто переважна частина близьких до нього навчальних станів є безпечною.

На основі центрів кластерів, трапецієподібних функцій належності та оцінок ризику автоматично формується база нечітких правил. Наприклад, для небезпечного кластера правило може мати такий зміст: якщо нормалізована ресурсна сигнатура процесу з високим ступенем належності відповідає кластеру, для якого характерне одночасне утримання декількох ресурсів і запит ресурсу, що бере участь у конфліктній залежності, то ризик переходу до взаємоблокування є високим.

Аналогічно формуються правила для інших кластерів. Після завершення навчання правила та параметри їх функцій належності зберігаються в базі B. Повторне виконання кластеризації потрібне лише під час оновлення навчальної вибірки або перебудови бази правил, а не для кожного нового поточного стану.

Таким чином, удосконалений метод реалізує навчальну частину архітектури

і конкретизує модель, забезпечуючи перехід від множини W та навчальних пар через нормалізацію і FCM-кластеризацію до оцінювання ризику та формування бази правил B за формулами (2.11)–(2.30). Особливістю методу є виконання нечіткої кластеризації за ресурсними ознаками процесів із подальшим оцінюванням ризику сформованих кластерів на основі навчальних даних. Це дає змогу формувати базу нечітких правил, що відображає типові ресурсні стани процесів та їх зв'язок із переходом у стан взаємоблокування, і зменшує залежність її побудови від експертного налаштування. Результатом роботи методу є база нечітких правил B для подальшого оцінювання станів процесів.

3.3 Метод прогнозування стану взаємоблокування процесів в КС

У розробленій архітектурі системи прогнозування стану взаємоблокування процесів метод прогнозування стану взаємоблокування процесів є безпосередньою алгоритмічною реалізацією методу M_3 у формулі (2.2), а у складі функції прогнозу – функції F_{M_3} у формулі (2.4). На рівні функціональних компонентів архітектури згідно з (2.1) він охоплює послідовну роботу M_{S_7} – модуля прогнозування, M_{S_8} – модуля оцінювання ризику та M_{S_9} – модуля формування прогнозного рішення; інформаційною основою для їх роботи є база правил M_{S_6}, сформована методом M_2, та сигнатури процесів, сформовані методом M_1 і об'єднані модулем M_{S_2} у поточне подання стану КС. Отже, метод 3.3 конкретизує передбачений архітектурою перехід від поточної ресурсної інформації про процеси та сформованої бази нечітких правил до оцінки ризику і прогнозного рішення.

У загальній послідовності прогнозування за (2.2) метод M_3 виконується після методів M_1 і M_2 та використовує сформовані ними результати: актуальні сигнатури процесів і базу нечітких правил. Аналогічно у формулі (2.4) функція F_{M_3} опрацьовує результати функцій F_{M_1} і F_{M_2} та формує підсумковий прогноз для процесів, відібраних із множини $M_{OS,pr,p}$. Сформоване модулем M_{S_9} рішення передається до передбаченої архітектурою множини функцій реакції M_{act}, де M_{act_1} відповідає

продовженню роботи процесу, а M_{act_2} – його перезапуску. Виділення прогнозування в окремий метод необхідне тому, що архітектура визначає склад компонентів, множини вхідних даних і їх функціональні зв'язки, однак не задає алгоритмічної послідовності відбору процесів, застосування бази правил, оцінювання ризику та формування рішення для поточної ресурсної події.

Метод прогнозування стану взаємоблокування процесів у КС призначений для оперативного визначення процесів, ресурсний стан яких може призвести до взаємоблокування. На відміну від методу побудови бази нечітких правил, цей метод не формує навчальну вибірку і не виконує повторну кластеризацію. Він використовує вже побудовану базу нечітких правил, поточну множину сигнатур процесів і події, пов'язані із запитом та ненаданням ресурсів.

Місце методу в загальній архітектурі системи визначається тим, що він працює після методу побудови сигнатури процесу та після методу побудови бази нечітких правил. Сигнатура процесу задає поточний формалізований опис процесу, а база правил відображає типові ресурсні стани, отримані на етапі навчання. Тому прогнозування не починається з аналізу сирих системних подій, а виконується над підготовленими об'єктами, структура яких уже узгоджена з моделями другого розділу.

Вхідними даними методу є оновлена множина сигнатур A^t відповідно до (2.31), поточні сигнатури процесів a_i^t, структура яких визначена моделлю (2.6)–(2.8), множина ідентифікаторів ресурсів R_e згідно (2.5), база нечітких правил B згідно (2.30), а також подія зміни ресурсної взаємодії. До таких подій належать створення процесу, зміна його параметрів, запит на ресурс, факт ненадання ресурсу, отримання ресурсу та звільнення ресурсу. Подальша послідовність прогнозного оцінювання безпосередньо відповідає співвідношенням (2.31)–(2.38).

Метод активується не для кожного процесу в системі, а для тих ситуацій, у яких зафіксовано зміну ресурсної взаємодії. Такий підхід узгоджується з моделлю прогнозування: якщо процес не має актуального запиту на ресурс, його поточний стан не є безпосереднім кандидатом на взаємоблокування. Якщо ж процес запитав

ресурс і не отримав його, то саме ця сигнатура повинна бути подана на процедуру оцінювання.

Вихідними даними методу є результат оцінювання поточного ресурсного стану процесів: відсутність ризикових процесів або сформована за умовами (2.36) і (2.37) множина D_θ процесів, що мають достатню належність до граничного або критичного кластерного стану. Якщо ця множина не порожня, за критерієм (2.38) визначається PID несистемного процесу з найменшим часом перебування в системі, для якого формується превентивне рішення про примусовий перезапуск. Після виконання дії актуалізуються множина сигнатур і стан ресурсів.

У межах цього методу база нечітких правил розглядається як готовий компонент. Якщо вона відсутня, прогнозування не може бути виконане коректно, оскільки немає правил, за якими поточна сигнатура зіставляється з типовими ресурсними станами. У такому випадку перед початком прогнозування потрібно виконати метод 3.2. Якщо база правил уже сформована, вона зчитується і використовується без повторного навчання.

Представимо процес прогнозування стану процесів в КС такими кроками.

Крок 1. Перевірка наявності бази правил. Перевіряється наявність бази нечітких правил. Якщо база правил відсутня, виконується перехід до кроку 2. Якщо база правил уже наявна, виконується перехід до кроку 3. Така перевірка потрібна для розмежування навчального й оперативного режимів роботи системи. Прогнозування не повинно щоразу запускати FCM-кластеризацію, оскільки це призвело б до надмірних обчислювальних витрат і змішування поточного аналізу з етапом навчання.

Крок 2. Формування відсутньої бази правил. Формується база нечітких правил за методом 3.2. На цьому кроці використовується попередньо накопичена множина станів КС, підготовлені ресурсні вектори, центри кластерів, ступені належності та оцінки ризику кластерів. Результатом є база правил B згідно (2.30). Після її формування метод повертається до кроку 3. У межах методу прогнозування цей крок є допоміжним: він виконується лише тоді, коли база правил ще не

створена або була втрачена.

Крок 3. Завантаження та перевірка бази правил. Зчитується база нечітких правил B. Під час зчитування перевіряється, чи містить вона правила для чотирьох кластерних станів, визначених на етапі навчання. Для подальшого прогнозування важливо, щоб порядок кластерів, центри кластерів і спосіб кодування ресурсів залишалися такими самими, як під час формування бази правил. Якщо структура бази правил не відповідає очікуваній, метод не повинен переходити до оцінювання поточних процесів, оскільки результат буде некоректним.

Крок 4. Реєстрація зміни ресурсного стану. Виявляється зміна в і-му процесі або в його ресурсній взаємодії. Зміною вважається така подія, яка може вплинути на сигнатуру процесу: поява процесу, зміна його параметрів, зміна набору зайнятих ресурсів, запит на ресурс, ненадання ресурсу або звільнення ресурсу. Якщо зміни не зафіксовано, виконується повернення до очікування наступної події. Якщо зміна зафіксована, виконується перехід до кроку 5.

Крок 5. Актуалізація сигнатури процесу. Оновлюється сигнатура процесу a_t. Оновлення виконується методом 3.1 та приводить поточні дані ОС до структури сигнатури згідно (2.6)–(2.8). При цьому зберігається зв’язок між PID процесу, його ідентифікаційними параметрами, ресурсами, які він уже займає, та ресурсом, який він запитує. Якщо процес новий, його сигнатура додається до A_t. Якщо процес уже присутній у A_t, попередній запис замінюється актуальним. Оновлення множини сигнатур після системної події відповідає (2.31).

Крок 6. Відбір процесу для прогнозування. Перевіряється, чи запитав процес ресурс і чи не отримав його. Якщо процес не має поточного запиту на ресурс або запит був успішно задоволений, його сигнатура не передається на прогнозне оцінювання і виконується перехід до кроку 12. Якщо процес запитав ресурс, але ресурс не був наданий, виконується перехід до кроку 7. Цей крок є ключовим фільтром, оскільки він відсікає процеси, що не беруть участі в потенційно небезпечній ресурсній залежності. Умова такого відбору формалізована в (2.32).

Крок 7. Оцінювання кластерної належності. Для сигнатури процесу, що

очікує ресурс, виконується оцінювання належності до кластерних станів. Поточна сигнатура приводиться до того самого виду, який використовувався під час навчання, згідно (2.33), після чого подається на базу правил B. Спрацювання правил визначає, наскільки поточний ресурсний стан процесу близький до відповідних кластерів. Основна увага приділяється третьому і четвертому кластерним станам, оскільки вони відповідають граничному стану та взаємоблокуванню. Результат спрацювання бази правил подається вектором ступенів належності за (2.34).

Крок 8. Формування оцінки небезпечного стану. Формується оцінка належності поточної сигнатури до небезпечної області станів. Для цього використовується результат роботи бази правил згідно (2.34) та об'єднання оцінок для третього і четвертого кластерів згідно (2.35). Якщо отримане значення не перевищує встановлений поріг, сигнатура не включається до множини D_θ і виконується перехід до кроку 12. Якщо поріг перевищено, сигнатура включається до D_θ і виконується перехід до кроку 9. Порогові умови формування множини ризикових процесів задаються (2.36) і (2.37).

Крок 9. Перевірка множини ризикових процесів. Перевіряється, чи є множина D_θ порожньою. Якщо D_θ порожня, у поточному стані не виявлено процесів, які потребують керуючої дії, тому виконується перехід до кроку 12. Якщо D_θ не порожня, це означає, що принаймні один процес має достатній ступінь належності до граничного або взаємозаблокованого стану. У такому випадку виконується перехід до кроку 10.

Крок 10. Вибір процесу-кандидата. Визначається PID процесу, який може бути обраний для керуючої дії. Вибір здійснюється серед процесів із множини D_θ з урахуванням обмеження, що процес не повинен бути системним. Додатково використовується критерій найменшого часу перебування процесу в системі, що задано у (2.38). Такий підхід дозволяє мінімізувати втрати від втручання та уникнути завершення критичних процесів ОС.

Крок 11. Примусовий перезапуск процесу. Для PID, визначеного за (2.38),

виконується примусовий перезапуск несистемного процесу з найменшим часом перебування в системі. Під час перезапуску поточний екземпляр процесу завершується, зайняті ним ресурси звільняються, після чого процес запускається повторно. Після цієї дії актуалізується множина сигнатур A^t і стан доступності ресурсів, щоб наступне прогнозне оцінювання виконувалося за поточними даними.

Крок 12. Контроль продовження роботи методу. Перевіряється умова завершення роботи методу. Якщо роботу системи прогнозування потрібно продовжити, виконується перехід до кроку 4 і метод очікує наступну зміну в процесах або ресурсах. Якщо роботу потрібно завершити, метод закінчує виконання. Завдяки цьому прогнозування працює циклічно та може застосовуватися до послідовності подій у реальному часі.

Графічне подання послідовності кроків методу зображено на рисунку 3.6.

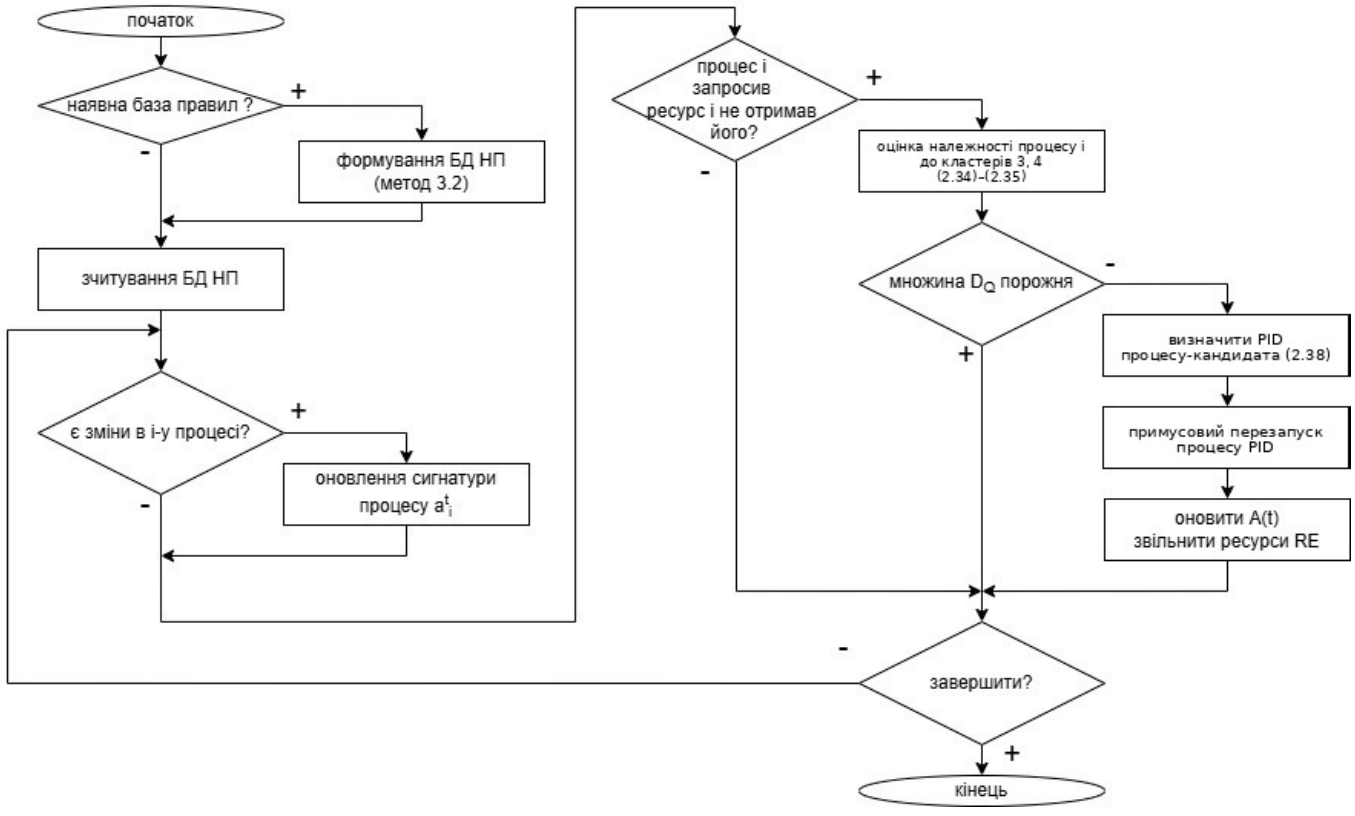


Рисунок 3.6 – Алгоритм прогнозування стану взаємоблокування процесів у КС

Подана схема відображає два режими роботи методу. Перший режим пов’язаний із підготовкою бази нечітких правил. Він виконується лише за відсутності готової бази правил і завершується формуванням B за методом 3.2.

Другий режим є основним і виконує прогнозування для поточних подій ОС. Саме цей режим забезпечує оперативну реакцію на ситуації, коли процес не отримав запитуваний ресурс.

Важливо, що метод не аналізує всі процеси однаково. Якщо процес змінив параметри, але не має факту очікування ресурсу, його сигнатура лише оновлюється в A_t. Це дозволяє підтримувати актуальний стан системи без зайвого запуску логічного висновку. Прогнозна процедура запускається лише тоді, коли у сигнатурі з'являється ненульова потреба в ресурсі, яка не була задоволена менеджером ресурсів.

Оцінювання належності до третього та четвертого кластерів використовується як механізм раннього виявлення небезпечного стану. Третій кластер відображає граничний стан, у якому взаємоблокування ще може не бути фактично зафіксованим. Четвертий кластер відповідає стану взаємоблокування або максимальному наближенню до нього за структурою ресурсних залежностей. Тому спільний аналіз цих двох кластерів дозволяє реагувати не тільки на вже сформоване взаємоблокування, а й на стан, що йому передує.

Множина D_θ є проміжним результатом методу. Вона не є повним списком усіх процесів у системі і не замінює A_t. Її зміст полягає у відборі тільки тих процесів, для яких належність до небезпечних кластерних станів перевищує заданий поріг. Якщо D_θ порожня, система продовжує спостереження без втручання. Якщо D_θ не порожня, виникає потреба у виборі процесу, до якого може бути застосована керуюча дія.

Вибір процесу для керуючої дії не повинен виконуватися лише за максимальним ризиком, оскільки серед ризикових процесів можуть бути системні процеси. Примусовий перезапуск такого процесу може спричинити порушення роботи ОС, втрату служб або нестабільність менеджера ресурсів. Тому в методі використовується обмеження на несистемність процесу. Лише після такого відбору застосовується критерій найменшого часу перебування в системі.

Для передблокувального або критичного стану в межах запропонованого

методу передбачено одну керуючу дію – примусовий перезапуск несистемного процесу, обраного за критерієм найменшого часу перебування в системі відповідно до (2.38). Такий вибір обмежує втручання лише множиною процесів, що пройшли прогнозний відбір, і виключає автоматичний вплив на системні процеси. Після перезапуску актуалізуються множина сигнатур A^t та стан ресурсів.

Результатом виконання методу є прогнозне оцінювання поточного ресурсного стану та, за наявності небезпечної конфігурації, сформована множина D_θ і PID процесу-кандидата, визначеного за (2.38). Якщо D_θ порожня, система продовжує спостереження без втручання. Якщо D_θ не порожня, виконується примусовий перезапуск обраного несистемного процесу, після чого оновлюються множина сигнатур A^t і відомості про доступні ресурси.

Таким чином, метод прогнозування стану процесів в КС забезпечує перехід від пасивного спостереження за ресурсними подіями до активного попередження взаємоблокування.

Розглянемо роботу запропонованого методу в реальному режимі функціонування комп'ютерної системи. Припустимо, що на попередньому етапі навчання вже була сформована база нечітких правил B, яка містить чотири правила, що відповідають типовим ресурсним станам процесів. Після завершення навчання система переходить у режим оперативного прогнозування, тому повторне виконання FCM-кластеризації вже не здійснюється.

У момент часу t_1 у комп'ютерній системі виконуються декілька процесів. Процес бази даних P_1 PID = 1240 утримує ресурси r_2 та r_5, процес вебсервера P_2 PID = 1384 використовує ресурс r_7, а процес резервного копіювання P_3 працює незалежно від інших процесів.

Під час роботи процес P_1 надсилає запит на отримання ресурсу r_7. Менеджер ресурсів встановлює, що цей ресурс уже зайнятий процесом P_2, тому запит не може бути виконаний. Саме ця подія є підставою для запуску методу прогнозування. Якби ресурс був успішно наданий, прогнозування не виконувалося б, оскільки потенційно небезпечна ресурсна залежність була б відсутня.

Метод отримує повідомлення про зміну стану процесу та викликає метод 3.1 для оновлення сигнатури процесу. У результаті формується актуальна сигнатура $a_1 = \{1240, r_2, r_5, r_7\}$, що означає, що процес утримує ресурси r_2 і r_5 та очікує отримання ресурсу r_7.

Далі сигнатура автоматично перетворюється у ресурсний вектор тієї самої структури, яка використовувалася під час навчання. Таке перетворення відповідає (2.33). Наприклад, якщо в системі використовується сім ресурсів, формується вектор $x_1 = [0,1,0,0,1,0,0 \mid 0,0,0,0,0,0,1]$. Отриманий вектор подається не на алгоритм FCM, а безпосередньо до бази нечітких правил B. Кожне правило визначає ступінь належності поточної сигнатури до одного із чотирьох кластерних станів.

Припустимо, що після виконання нечіткого логічного висновку були отримані такі результати: $\gamma_1 = 0.05,\ \gamma_2 = 0.18,\ \gamma_3 = 0.61,\ \gamma_4 = 0.84$. Отримані значення означають, що поточний стан процесу має найвищий ступінь належності до четвертого кластерного стану, який під час навчання був пов'язаний із високою ймовірністю виникнення взаємоблокування. Крім того, досить велике значення належності до третього кластера свідчить, що процес уже перебуває у граничному ресурсному стані.

Далі метод об'єднує оцінки третього та четвертого кластерів відповідно до формули (2.35) та порівнює отримане значення із заданим порогом спрацювання згідно (2.36)–(2.37). Нехай поріг становить $\theta = 0.60$. Оскільки об’єднана за (2.35) оцінка небезпечного стану перевищує встановлений поріг, процес P_1 включається до множини кандидатів для керуючої дії.

Одночасно система аналізує інші процеси, які також очікують ресурси. Наприклад, процес P_2 має оцінку небезпечності 0.42, а процес P_3 лише 0.08. Оскільки ці значення не перевищують встановлений поріг, вони не включаються до множини ризикових процесів.

Після формування множини кандидатів виконується вибір процесу, щодо якого може бути застосована керуюча дія. Спочатку перевіряється, чи не є процес

системним. Оскільки процес P_1 належить користувацькому застосунку, він може бути розглянутий як кандидат для втручання. Якщо декілька процесів мають високий рівень ризику, додатково використовується критерій найменшого часу перебування процесу в системі. Припустимо, що саме процес P_1 був створений найпізніше, тому він обирається для виконання керуючої дії.

Оскільки процес P_1 є несистемним і, за умовами прикладу, має найменший час перебування в системі серед процесів-кандидатів, для нього виконується примусовий перезапуск. Поточний екземпляр P_1 завершується, менеджер ресурсів звільняє ресурси r_2 та r_5, після чого процес запускається повторно, а множина сигнатур активних процесів актуалізується. Звільнення утримуваних ресурсів усуває сформовану конфліктну залежність і дає змогу продовжити подальше спостереження за оновленим станом КС.

Отже, розроблений метод прогнозування стану взаємоблокування процесів реалізує метод M_3 з формули (2.2) та функцію F_{M_3} з формули (2.4), конкретизуючи послідовність прогнозного оцінювання за формулами (2.31)–(2.38): від опрацювання поточних сигнатур процесів сформованою базою нечітких правил B до оцінювання ризику, формування множини ризикових процесів і прогнозного рішення. Особливістю методу є завчасне оцінювання ризику переходу процесів у стан взаємоблокування шляхом обробки їх поточних сигнатур базою нечітких правил, сформованою за результатами кластеризації, що дає змогу враховувати поточний ресурсний стан комп’ютерної системи та особливості взаємодії процесів із ресурсами. Результатом роботи методу є формування підстав для прийняття превентивних рішень щодо керування ресурсами до фактичного виникнення взаємоблокування.

3.4 Висновки до третього розділу

Таким чином, розроблений метод побудови сигнатури забезпечує перетворення даних операційної системи про процес та його ресурсну взаємодію у

структуроване подання, яке актуалізується в процесі функціонування системи та використовується як єдина інформаційна основа для подальшого аналізу. Відокремлення ознакової частини сигнатури від навчального маркера дає змогу використовувати однакове подання даних як під час навчання, так і в режимі поточного прогнозування.

Метод побудови бази нечітких правил реалізує перехід від накопичених сигнатур до узагальненого опису характерних ресурсних станів процесів. Підготовлені ресурсні вектори кластеризуються методом Fuzzy C-Means, після чого сформовані кластери оцінюються з урахуванням фактичних наслідків ресурсної взаємодії. Прогнозний маркер при цьому не впливає на процес кластеризації, а використовується на наступному етапі для визначення небезпечності отриманих кластерів і формування бази нечітких правил. Така організація дає змогу будувати правила на основі накопичених даних про функціонування системи без необхідності задавати належність кожної ресурсної конфігурації до певного класу до виконання кластеризації.

Метод прогнозування використовує сформовані сигнатури та базу нечітких правил у режимі поточного функціонування КС. Після зміни ресурсної взаємодії процесів виконується оцінювання їхнього стану, визначається множина процесів із підвищеним ризиком взаємоблокування та, за необхідності, формується рішення щодо превентивного впливу. Таким чином, розроблені методи утворюють завершену послідовність перетворень від отримання та формалізації системних даних до оцінювання ризику й формування прогнозного рішення до фактичного виникнення взаємоблокування.

Основні результати розділу опубліковані у працях [117 – 119; 121; 122].

РОЗДІЛ 4
ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ОЦІНЮВАННЯ МЕТОДІВ

4.1 Архітектура програмного засобу прогнозування стану взаємоблокування процесів

Для практичного використання розроблених моделей і методів сформовано архітектуру засобу прогнозування стану взаємоблокування процесів, що забезпечує отримання даних від ОС, спостереження за змінами ресурсної поведінки процесів, прогнозне оцінювання небезпечних станів і передавання керуючих дій до середовища виконання. Архітектурне рішення орієнтовано на роботу з поточними подіями ОС, тому в ньому відокремлено рівень доступу до системної інформації, рівень аналітичного опрацювання та рівень превентивного впливу на процеси.

Загальна архітектура засобу прогнозування стану взаємоблокування процесів зображена на рисунку 4.1. У ній окремо виділено ядро ОС як джерело подій і виконавець керуючих дій, а також власне засіб прогнозування, межі якого позначено пунктирним контуром. Усередині засобу відображено не лише основні функціональні блоки, а й послідовність перетворення системних подій у керуюче рішення щодо недопущення взаємоблокування.

Первинним джерелом даних є ядро операційної системи, зокрема планувальник процесів, менеджер ресурсів, підсистема обліку процесів і механізми синхронізації. Від цих компонентів до засобу прогнозування надходять відомості про створення, виконання, блокування та завершення процесів, а також події виділення, запиту й звільнення ресурсів. До складу таких даних входять ідентифікатор процесу, поточний стан процесу, множина утримуваних ресурсів, запитуваний ресурс, час очікування, кількість повторних звернень до ресурсу, пріоритет процесу та тип ресурсу.

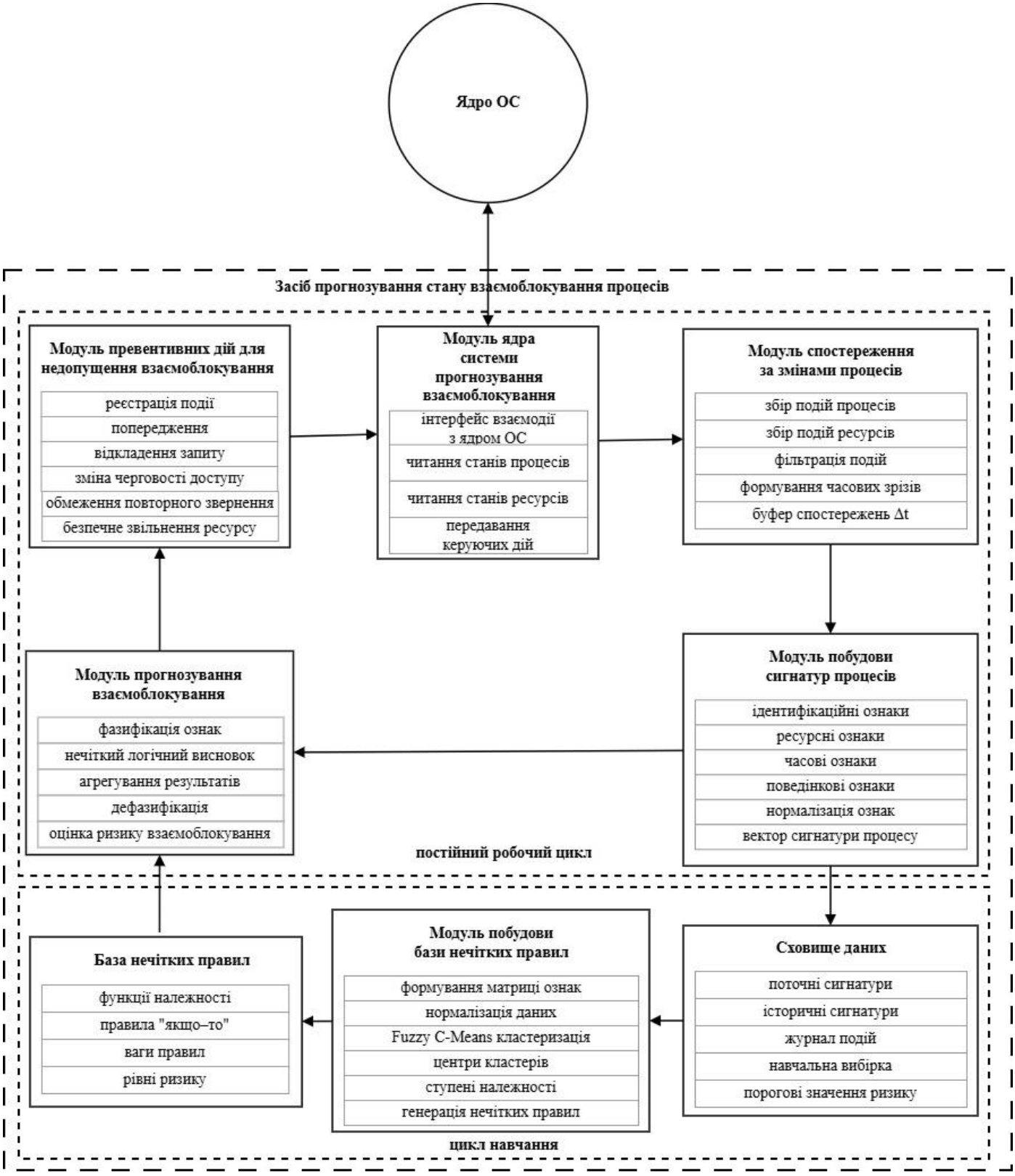


Рисунок 4.1 - Архітектура програмного засобу прогнозування стану взаємоблокування процесів

Модуль ядра системи прогнозування взаємоблокування виконує роль узгоджувального компонента між ядром ОС і засобом прогнозування. Він

забезпечує отримання системної інформації про стани процесів і ресурсів, передавання цих даних до модуля спостереження, а також повернення керуючих дій до ядра ОС або менеджера ресурсів. За функціональним призначенням цей модуль забезпечує реалізацію методу побудови сигнатури процесу, оскільки саме через нього отримуються первинні параметри процесу, необхідні для формування його сигнатури. Крім того, у частині передавання керуючих дій цей модуль забезпечує виконання результатів методу прогнозування стану процесів в КС. Алгоритм роботи модуля наступний.

Крок 1. Модуль ініціалізує інтерфейс взаємодії з ядром ОС та отримує доступ до джерел інформації про активні процеси, їхні стани, ресурси та події синхронізації.

Крок 2. Виконується читання поточного переліку активних процесів. Для кожного процесу визначається його ідентифікатор, стан виконання, пріоритет, час існування та ознаки перебування у стані очікування.

Крок 3. Для кожного активного процесу модуль зчитує інформацію про ресурси, які вже виділені процесу.

Крок 4. Модуль перевіряє наявність для процесу поточного запиту на додатковий ресурс. Якщо запит відсутній, то інформація про процес передається до модуля спостереження як звичайний стан. Якщо запит на ресурс зафіксовано, то процес позначається як кандидат для подальшого аналізу ризику.

Крок 5. Якщо процес є новим або його стан змінився порівняно з попереднім циклом спостереження, то модуль передає відповідні дані до модуля спостереження за змінами процесів. Якщо змін не виявлено, то виконується перехід до кроку 7.

Крок 6. Модуль приймає від модуля превентивних дій керуючу команду. Якщо команда відсутня, то робота продовжується в режимі моніторингу. Якщо команда наявна, то модуль перевіряє її допустимість для відповідного процесу.

Крок 7. Якщо керуюча дія є допустимою, вона передається до ядра ОС або менеджера ресурсів. Якщо дія недопустима, наприклад, стосується системного

процесу або критичного ресурсу, то вона блокується.

Крок 8. Після завершення обробки поточних процесів модуль переходить до наступного циклу спостереження.

Модуль спостереження за змінами процесів виконує збір і попередню обробку подій, що надходять від ядра ОС. Його функціями є формування поточного стану комп'ютерної системи та передавання підготовлених даних до модуля побудови сигнатур. Цей модуль реалізує початковий етап методу побудови сигнатури процесу, оскільки забезпечує накопичення та упорядкування відомостей про процеси, утримувані ресурси, запитувані ресурси та часові характеристики їх очікування. Алгоритм роботи модуля наступний.

Крок 1. Модуль приймає від модуля ядра системи прогнозування потік подій про процеси та ресурси.

Крок 2. Виконується класифікація отриманих подій. Події поділяються на події створення процесу, зміни стану процесу, виділення ресурсу, запиту ресурсу, звільнення ресурсу та завершення процесу.

Крок 3. Модуль виконує фільтрацію службових і нерелевантних подій. Якщо подія не впливає на ресурсну поведінку процесу, то вона не передається на подальший аналіз, але може бути зафіксована в журналі.

Крок 4. Якщо подія пов'язана зі зміною стану процесу або ресурсною взаємодією, то вона включається до поточного буфера спостережень за інтервал часу Δt.

Крок 5. Модуль перевіряє стан завершення поточного часового інтервалу спостереження. Якщо інтервал Δt ще не завершено, то виконується повернення до кроку 1. Якщо інтервал завершено, то виконується перехід до кроку 6.

Крок 6. Формується часовий зріз стану системи, який містить множину активних процесів, множину зайнятих ресурсів, множину запитуваних ресурсів і поточні залежності між процесами та ресурсами.

Крок 7. Для кожного процесу з часового зрізу перевіряється зміна його параметрів порівняно з попереднім зрізом. Якщо зміни відсутні, то процес не

передається на повторну побудову сигнатури. Якщо зміни наявні, дані процесу передаються до модуля побудови сигнатур процесів.

Крок 8. Після передавання даних буфер поточного інтервалу очищується, після чого модуль переходить до нового циклу спостереження.

Модуль побудови сигнатур процесів формує впорядкований вектор ознак, що описує поточну ресурсну поведінку процесу. Цей модуль безпосередньо реалізує метод побудови сигнатури процесу, оскільки його основним результатом є формалізований опис процесу у вигляді вектора ознак. Алгоритм роботи модуля наступний.

Крок 1. На вхід модуля надходить ідентифікатор процесу PID та набір параметрів, отриманих від модуля спостереження.

Крок 2. Модуль перевіряє наявність в поточній множині сигнатур запис для процесу з цим PID. Якщо сигнатура існує, то виконується перехід до кроку 5. Якщо сигнатура відсутня, то виконується перехід до кроку 3.

Крок 3. Для нового процесу формується ідентифікаційна частина сигнатури. До неї вносяться PID, тип процесу, пріоритет, час створення або час перебування у системі.

Крок 4. Створюється новий запис сигнатури процесу в множині поточних сигнатур.

Крок 5. Модуль визначає ресурсні ознаки процесу.

Крок 6. Виконується кодування утримуваних і запитуваних ресурсів у вигляді числових ознак.

Крок 7. Формується сигнатура процесу. Якщо процес не має запиту на ресурс, то сигнатура передається до сховища даних лише для збереження поточного стану. Якщо процес має запит на ресурс, то сигнатура додатково передається до модуля прогнозування взаємоблокування.

Крок 8. Оновлена сигнатура записується до сховища даних.

Сховище даних містить історичні сигнатури процесів. Наявність такого сховища дає змогу розділити навчальну фазу, під час якої формується база нечітких

правил, і фазу поточного прогнозування, під час якої система застосовує вже сформовані правила до нових сигнатур процесів. Цей блок забезпечує інформаційну підтримку методу побудови бази нечітких правил. Алгоритм роботи модуля наступний.

Крок 1. Сховище приймає від модуля побудови сигнатур поточну сигнатуру процесу.

Крок 2. Перевіряється наявність в сховищі запису для цього процесу. Якщо запис існує, то виконується його оновлення. Якщо запису немає, то створюється новий запис.

Крок 3. Поточна сигнатура записується до множини актуальних сигнатур активних процесів.

Крок 4. Перевіряється необхідність щодо зберігання сигнатури як навчального прикладу. Якщо сигнатура належить до завершеного сценарію або має відомий фактичний результат, то вона включається до навчальної вибірки. Якщо результат ще невідомий, то сигнатура залишається лише в поточному сховищі.

Крок 5. Сховище перевіряє обсяг навчальної вибірки. Якщо кількість накопичених прикладів менша за заданий поріг, то формування бази нечітких правил не запускається. Якщо поріг досягнуто, то дані передаються до модуля побудови бази нечітких правил.

Крок 6. Якщо процес завершився, то його поточна сигнатура переноситься до історичних даних.

Модуль побудови бази нечітких правил реалізує навчальну частину архітектури. На його вхід надходить множина історичних сигнатур процесів і відповідні стани комп’ютерної системи за період спостереження. На основі цих даних формується матриця ознак, виконується нормалізація та нечітка кластеризація Fuzzy C-Means. У результаті визначаються типові групи станів процесів, центри кластерів, ступені належності сигнатур до кластерів і рівні ризику, пов’язані з кожним кластером. На основі отриманих кластерів генерується база нечітких правил, яка описує залежність між ознаками процесу та ризиком переходу

до стану взаємоблокування. Цей модуль безпосередньо реалізує метод побудови бази нечітких правил. Алгоритм роботи модуля наступний.

Крок 1. Модуль отримує зі сховища навчальну вибірку, що містить історичні сигнатури процесів і відомі стани системи.

Крок 2. Виконується перевірка повноти навчальних даних. Якщо вибірка порожня або містить недостатню кількість записів, то побудова бази правил не виконується, а модуль повертається до очікування нових даних.

Крок 3. З навчальних сигнатур формується матриця ознак. Кожен рядок матриці відповідає одному спостереженню, а кожен стовпець – окремій ознаці процесу або ресурсної взаємодії.

Крок 4. Виконується нормалізація ознак.

Крок 5. Запускається процедура нечіткої кластеризації Fuzzy C-Means. Для кожної сигнатури обчислюються ступені належності до кожного кластера.

Крок 6. Модуль перевіряє умову збіжності кластеризації. Якщо зміна центрів кластерів перевищує заданий поріг, то виконується повторна ітерація Fuzzy C-Means, тобто перехід до кроку 5. Якщо умова збіжності виконана, виконується перехід до кроку 7.

Крок 7. Для кожного кластера визначається рівень ризику взаємоблокування на основі складу сигнатур, що мають найбільший ступінь належності до цього кластера.

Крок 8. На основі центрів кластерів і ступенів належності формуються нечіткі правила типу “якщо–то”.

Крок 9. Сформована база нечітких правил передається до сховища правил і стає доступною для модуля прогнозування.

База нечітких правил є окремим інформаційним ресурсом засобу прогнозування. Вона містить функції належності, центри кластерів, правила нечіткого логічного висновку, оцінки ризику. Цей блок є результатом виконання методу побудови бази нечітких правил і водночас є вхідним інформаційним ресурсом для методу прогнозування стану процесів в КС. Алгоритм роботи модуля

наступний.

Крок 1. База приймає сформований набір нечітких правил від модуля побудови бази нечітких правил.

Крок 2. Для кожного правила зберігаються умови, висновок, пов’язаний рівень ризику та параметри функцій належності.

Крок 3. Під час надходження запиту від модуля прогнозування база передає набір правил і параметри функцій належності, необхідні для нечіткого логічного висновку.

Крок 4. Якщо база правил відсутня або неактуальна, модуль прогнозування отримує повідомлення про неможливість виконати повноцінне прогнозування.

Модуль прогнозування взаємоблокування приймає поточну сигнатуру процесу, зіставляє отримані значення з базою нечітких правил і обчислює ступені спрацювання правил. За результатами спрацювання нечітких правил визначається ступінь відповідності поточного стану процесу сформованим класам станів та обирається клас із найбільшим ступенем відповідності. Залежно від отриманого результату стан процесу може бути віднесений до одного з рівнів: нормальний, потенційно ризиковий, передблокувальний або критичний. Цей модуль безпосередньо реалізує метод прогнозування стану процесів в КС. Алгоритм роботи модуля наступний.

Крок 1. Модуль отримує поточну сигнатуру процесу від модуля побудови сигнатур або зі сховища даних.

Крок 2. Перевіряється наявність у процесу поточного запиту на ресурс. Якщо запит відсутній, то прогнозування для цього процесу не виконується. Якщо запит на ресурс зафіксовано, то виконується перехід до кроку 3.

Крок 3. Модуль отримує з бази нечітких правил функції належності, правила “якщо–то” та рівні ризику.

Крок 4. Для кожного правила бази визначається ступінь його спрацювання. Якщо умови правила не виконуються, то ступінь спрацювання дорівнює нулю або є незначним.

Крок 5. Виконується агрегування результатів усіх правил. Для кожного рівня ризику обчислюється узагальнений ступінь належності процесу до відповідного стану.

Крок 6. На основі отриманих ступенів відповідності визначається клас стану, для якого сформовано найбільший результуючий ступінь відповідності поточної сигнатури процесу.

Крок 7. Якщо поточний стан процесу віднесено до нормального класу, то результат прогнозування фіксується і виконується перехід до кроку 10..

Крок 8. Якщо стан процесу віднесено до потенційно ризикового, передблокувального або критичного класу, то формується відповідний результат прогнозування. Для передблокувального та критичного станів інформація передається до модуля превентивних дій для вибору керуючого впливу.

Крок 9. Якщо декілька процесів одночасно віднесено до передблокувального або критичного стану та вони пов’язані спільними ресурсами, то модуль формує множину граничних процесів для подальшого вибору об’єкта керуючого впливу.

Крок 10. Результат прогнозування, визначений клас стану процесу, відповідні ступені належності та ідентифікатори спрацьованих нечітких правил записуються до сховища даних.

Модуль превентивних дій перетворює результат прогнозування на керуюче рішення. Для процесів, віднесених до нормального стану та потенційно ризикового стану, передбачено реєстрацію результату прогнозування без формування керуючого впливу. Для передблокувального та критичного станів як превентивну керуючу дію передбачено примусовий перезапуск одного з несистемних процесів, залучених до потенційного взаємоблокування. Вибір процесу для перезапуску здійснено за критерієм найменшого часу його перебування в системі. Системні процеси вилучено з множини кандидатів на примусовий перезапуск. Такий підхід забезпечив розрив потенційної послідовності взаємного очікування ресурсів із мінімізацією втрат результатів виконання процесів, оскільки для перезапуску обрано процес із найменшим часом виконання на момент формування керуючого

рішення. Таким чином, модулем реалізовано заключний етап методу прогнозування стану процесів у КС, у межах якого за результатом визначення прогнозованого класу стану сформовано превентивний керуючий вплив. Алгоритм роботи модуля наступний.

Крок 1. Від модуля прогнозування отримано визначені класи станів процесів,

Крок 2. Перевірено визначений клас стану процесу. Якщо процес віднесено до нормального або до потенційно ризикового стану, то результат прогнозування зареєстровано без формування керуючої дії та здійснено перехід до кроку 7.

Крок 3. Якщо один або декілька процесів віднесено до передблокувального або критичного стану, то сформовано множину процесів-кандидатів на застосування превентивної керуючої дії.

Крок 4. Із сформованої множини кандидатів вилучено системні процеси, примусовий перезапуск яких може порушити функціонування операційної системи.

Крок 5. Серед решти несистемних процесів визначено процес із найменшим часом перебування в системі. Цей процес обрано як об'єкт превентивного керуючого впливу.

Крок 6. Для обраного процесу сформовано керуючу дію примусового перезапуску та передано її до модуля ядра системи прогнозування взаємоблокування для виконання засобами операційної системи.

Крок 7. Визначений клас стану процесу, ідентифікатор обраного для перезапуску процесу та результат виконання керуючої дії записано до журналу подій і сховища історичних даних.

Таким чином, розширена архітектура забезпечує повний цикл функціонування засобу прогнозування. Він включає процес від збору системних подій і побудови сигнатур процесів до нечіткого оцінювання ризику та формування превентивних рішень. Її особливістю є поєднання трьох розроблених методів: методу побудови сигнатури процесу, методу побудови бази нечітких правил та методу прогнозування стану процесів в КС.

4.2 Оцінка достовірності методу прогнозування взаємоблокувань процесів

Для програмної реалізації середовища імітаційного моделювання обрано мову Python. Її використання дає змогу подати всі компоненти моделі у вигляді однозначно визначених програмних процедур, винести параметри сценарію до зовнішніх файлів і автоматизувати послідовні запуски без ручної перебудови моделі. На відміну від спеціалізованих графічних пакетів, така реалізація не обмежує спосіб включення власних алгоритмів, спрощує контроль проміжних станів і забезпечує відтворюваність обробки однакових вхідних даних.

Подієве ядро середовища реалізовано засобами бібліотеки SimPy. Її процесний підхід відповідає структурі досліджуваної системи: модельовані процеси описуються генераторними функціями, обчислювальні ядра та інші обмежені об’єкти подаються ресурсами SimPy, а затримки виконання задаються подіями модельного часу. На відміну від використання реальних потоків операційної системи, послідовність подій не залежить від поточного завантаження комп’ютера, а очікування не потребує фактичної затримки на відповідну кількість секунд. Це дає змогу відокремити перебіг моделі від реального часу виконання програми та обробляти великі набори сценаріїв у керованому подієвому середовищі.

Середовище побудовано за конфігураційно-подієвим принципом. Параметри процесів, ресурсів і досліджуваного методу надходять із зовнішніх файлів, після чого проходять завантаження та перевірку. Підготовлені дані використовуються для ініціалізації подієвого ядра, генерації екземплярів процесів, відтворення їх виконання та реєстрації результатів. Структуру середовища і напрямки передавання інформації між його блоками зображено на рисунку 4.2.

Вхідний рівень містить чотири групи даних. Файл сценарію процесів задає ідентифікатор процесу, момент його надходження, тривалості окремих етапів обчислення, ідентифікатори запитуваних ресурсів і затримки між операціями їх звільнення. Конфігурація ресурсів визначає доступні ідентифікатори та місткість кожного модельованого ресурсу. Окремі файли зберігають параметри, потрібні реа-

лізації досліджуваного методу, зокрема координати центрів і числові характеристики правил. Додаткові константи програми задають кількість процесорних ядер, квант процесорного часу, граничну тривалість існування екземпляра процесу, інтервал періодичного контролю та режим детального журналювання.



Рисунок 4.2 – Структура середовища імітаційного моделювання та інформаційні потоки

Блок завантаження та валідації перетворює текстові записи CSV у внутрішні структури Python і не допускає запуску моделі з неузгодженими даними. Перевіряються наявність обов’язкових стовпців, унікальність ідентифікаторів процесів і ресурсів, невід’ємність часових параметрів, коректність посилань процесів на ресурси та допустимість значень місткості. Для параметрів методу додатково контролюються унікальність правил, діапазон їх вихідних значень, наявність відповідного центра і збіг розмірності вектору центра з розмірністю стану, сформованого для заданої кількості ресурсів.

Після успішної перевірки створюється об’єкт SimPy Environment, який зберігає модельний час і чергу майбутніх подій. Процесорна підсистема подається ресурсом із місткістю, що дорівнює кількості модельованих ядер. Виконання процесу на ядрі розбивається на інтервали, довжина яких не перевищує заданого кванта.

Після завершення кванта ядро звільняється, а за наявності невиконаної частини обчислення процес формує новий запит до процесорного ресурсу. Така організація відтворює циклічне обслуговування процесів без використання реального паралельного виконання.

Модельовані ресурси формуються як словник об’єктів SimPy Resource, доступ до яких здійснюється за ідентифікаторами з конфігураційного файла. Генератор процесів впорядковує записи сценарію за часом надходження, переводить модельний годинник до чергової часової позначки та створює відповідний екземпляр процесу. Кожен екземпляр виконує визначену сценарієм послідовність: обчислювальний етап, запит першого ресурсу, наступний обчислювальний етап, запит другого ресурсу, обчислення перед звільненням ресурсів і завершальний етап. Тривалість кожної частини береться безпосередньо з рядка вхідного файла.

Стан середовища підтримується спільними структурами обліку. У них зберігаються активні екземпляри процесів, зайняті ресурси, поточні запити, часові позначки, накопичені інтервали очікування, кількість звернень до процесорних ядер і службові ознаки завершення. Модуль реалізації досліджуваного методу отримує лише підготовлене подієвим ядром подання поточного стану та повертає результат обробки через визначений програмний інтерфейс. Завдяки цьому сценарій навантаження, механізм відтворення процесів і код методу залишаються логічно розділеними.

Блок реєстрації результатів працює під час усього прогону. Модельний час використовується для визначення моментів надходження, завершення та тривалості очікувань, тоді як реальний час окремо вимірюється високоточною функцією таймера Python. Це виключає змішування часових характеристик модельованої системи з обчислювальними витратами самої програми. Після завершення черги подій формуються детальні записи для кожного екземпляра процесу, агрегована статистика прогону, журнали подій і показники часу виконання програмних процедур. Назви вихідних файлів доповнюються ідентифікатором поточної конфігурації, що дає змогу зберігати результати серії запусків окремо.

Загальний алгоритм роботи середовища імітаційного моделювання має таку послідовність.

Крок 1. Задаються загальні параметри прогону: кількість процесорних ядер, квант процесорного часу, гранична тривалість існування процесу, інтервал періодичних подій, режим журналювання та імена вхідних файлів.

Крок 2. Із файлу сценарію зчитуються конфігурації процесів, а з файлу ресурсів – перелік доступних ресурсів та їх місткість. Паралельно завантажуються зовнішні параметри реалізації досліджуваного методу.

Крок 3. Виконується структурна й логічна перевірка вхідних даних. У разі відсутності обов’язкового поля, дублювання ідентифікатора, недопустимого часового значення або некоректного посилання запуск припиняється з повідомленням про виявлену помилку.

Крок 4. Створюється середовище SimPy, процесорний ресурс із заданою кількістю ядер, набір модельованих ресурсів і внутрішні структури для обліку поточного стану та статистики.

Крок 5. Запускаються службові процеси моделі: генератор екземплярів процесів, періодичний реєстратор стану та засоби контролю тривалості виконання окремого екземпляра.

Крок 6. Генератор упорядковує конфігурації за часом надходження. Після настання відповідної часової позначки для кожного запису створюється окремий процес SimPy і формується його початковий статистичний запис.

Крок 7. Екземпляр процесу послідовно відтворює задані у сценарії обчислювальні етапи. Кожен етап виконується квантами: процес запитує вільне ядро, використовує його протягом не більше одного кванта та, за потреби, повторно стає в чергу.

Крок 8. У визначених сценарієм точках процес формує запити до модельованих ресурсів. Подієве ядро фіксує момент запиту, тривалість очікування, момент надання ресурсу та подальше його звільнення.

Крок 9. Після подій, для яких передбачено аналіз, формується актуальне подання стану і передається модулю досліджуваного методу. Результат його роботи повертається до модельного контуру та реєструється разом із часовою позначкою і використаними параметрами.

Крок 10. Після кожної зміни стану оновлюються структури зайнятих ресурсів, активних запитів і статистики процесу. Для завершеного екземпляра обчислюються тривалість перебування в системі, сумарний час очікування та кількість отриманих квантів.

Крок 11. Подієве виконання продовжується, доки не завершено генерацію сценарію та в черзі SimPy не залишиться активних або запланованих процесів. Окремо визначається реальна тривалість усього прогону і сумарний час виконання програмних процедур аналізу.

Крок 12. Накопичені дані записуються до вихідних файлів: детальної таблиці станів процесів, узагальненого текстового звіту, журналів подій і допоміжних наборів, потрібних для подальшої статистичної обробки.

Описана організація середовища забезпечує єдиний порядок підготовки вхідних даних, відтворення сценарію та формування вихідних показників для всіх досліджуваних методів. Зміна набору процесів, кількості ресурсів або параметрів методу виконується через конфігураційні файли й константи запуску, тому програмне ядро моделі не потребує перебудови перед кожним окремим прогоном.

За результатами аналізу можливості реалізації методів вирішення задачі взаємоблокування, представлених у таблиці 1.3, для імітаційного моделювання обрано такі: ігнорування взаємоблокувань (ІГН), примусове вилучення ресурсу або відкат (ПВР), заборона на переривання (ЗП), алгоритм банкіра (АБ), граф розподілу ресурсів (ГПР), аналіз графа очікування (ГО), запропонований метод прогнозування взаємоблокувань (МП). Решта відомих методів мають переважно теоретичний характер, обмежено застосовуються на практиці або потребують спеціалізованої архітектури апаратних і програмних засобів.

Для наочного порівняння часової логіки роботи обраних для імітаційного моделювання методів на рисунку 4.3 зображено послідовність їх реагування на зміну ресурсного стану процесу. На часовій осі відображено обчислювальний етап, формування запиту на ресурс, очікування його надання та подальше отримання ресурсу або виникнення взаємоблокування. Для ІГН, ПВР, ЗП, АБ, ГПР, ГО та МП відображено моменти виконання характерних для кожного методу дій відносно моменту фактичного виникнення взаємоблокування.

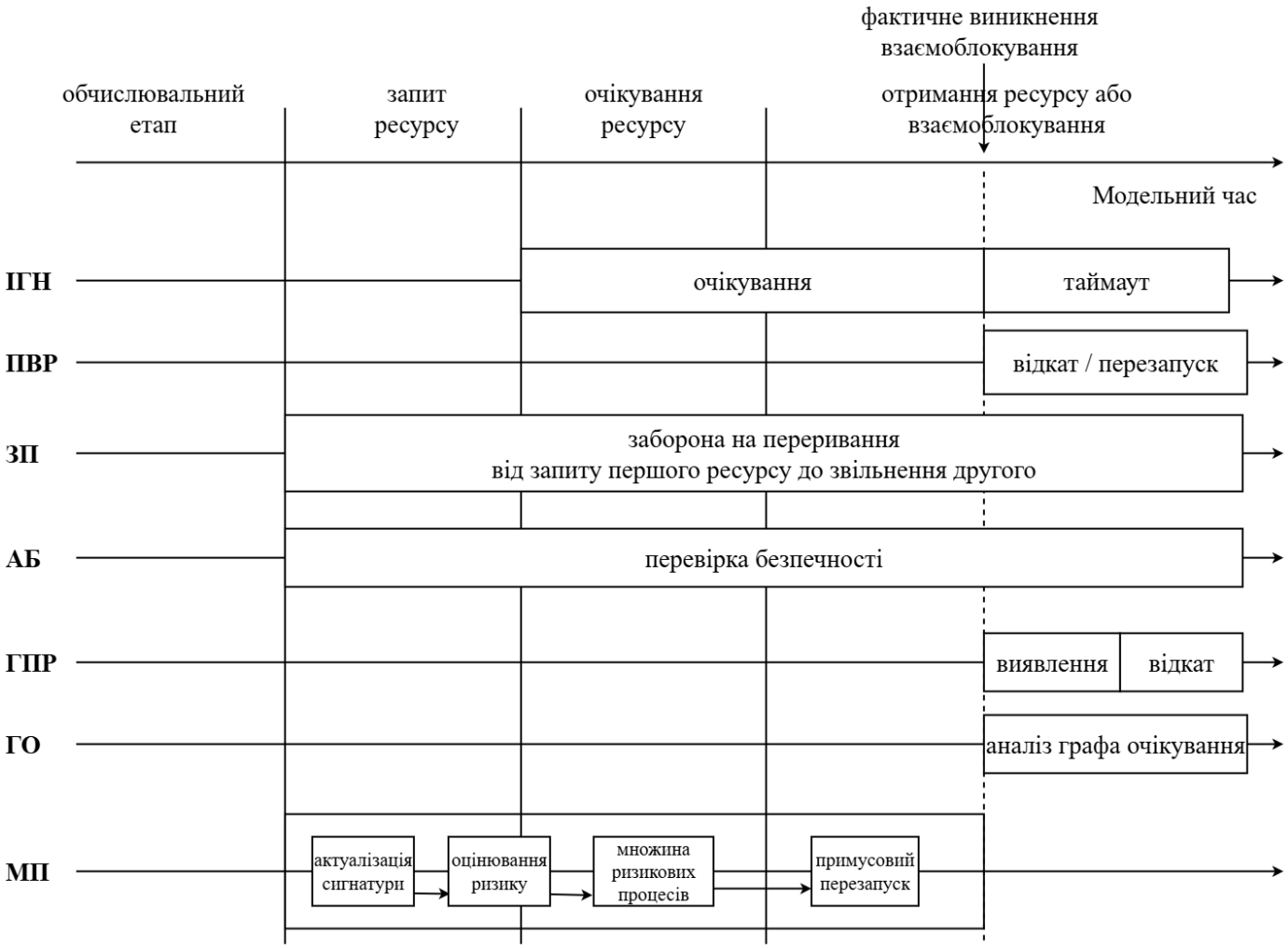


Рисунок 4.3 – Часова діаграма реагування методів на змінуресурсного стану процесів

Згідно рисунку 4.3, досліджувані методи відрізняються насамперед моментом втручання у перебіг ресурсної взаємодії процесів. ІГН не передбачає активного реагування до завершення граничного часу очікування, а ПВР, ГПР і ГО виконують

відповідні дії після фактичного виникнення взаємоблокування. ЗП та АБ впливають на порядок використання ресурсів до формування критичного стану. Запропонований МП виконує аналіз уже на етапі очікування ресурсу: актуалізує сигнатуру процесу, оцінює ризик, формує множину ризикових процесів і, за необхідності, застосовує превентивну керуючу дію до фактичного виникнення взаємоблокування. Для деталізації роботи запропонованого методу розглянемо часову послідовність обробки події ненадання процесу запитуваного ресурсу. На рисунку 4.4 зображено перехід від фіксації цієї події та актуалізації сигнатури процесу до формування ресурсного вектора, його обробки базою нечітких правил, оцінювання ризику та формування множини ризикових процесів. Завершальним етапом є вибір несистемного процесу з найменшим часом перебування в системі та виконання керуючої дії.

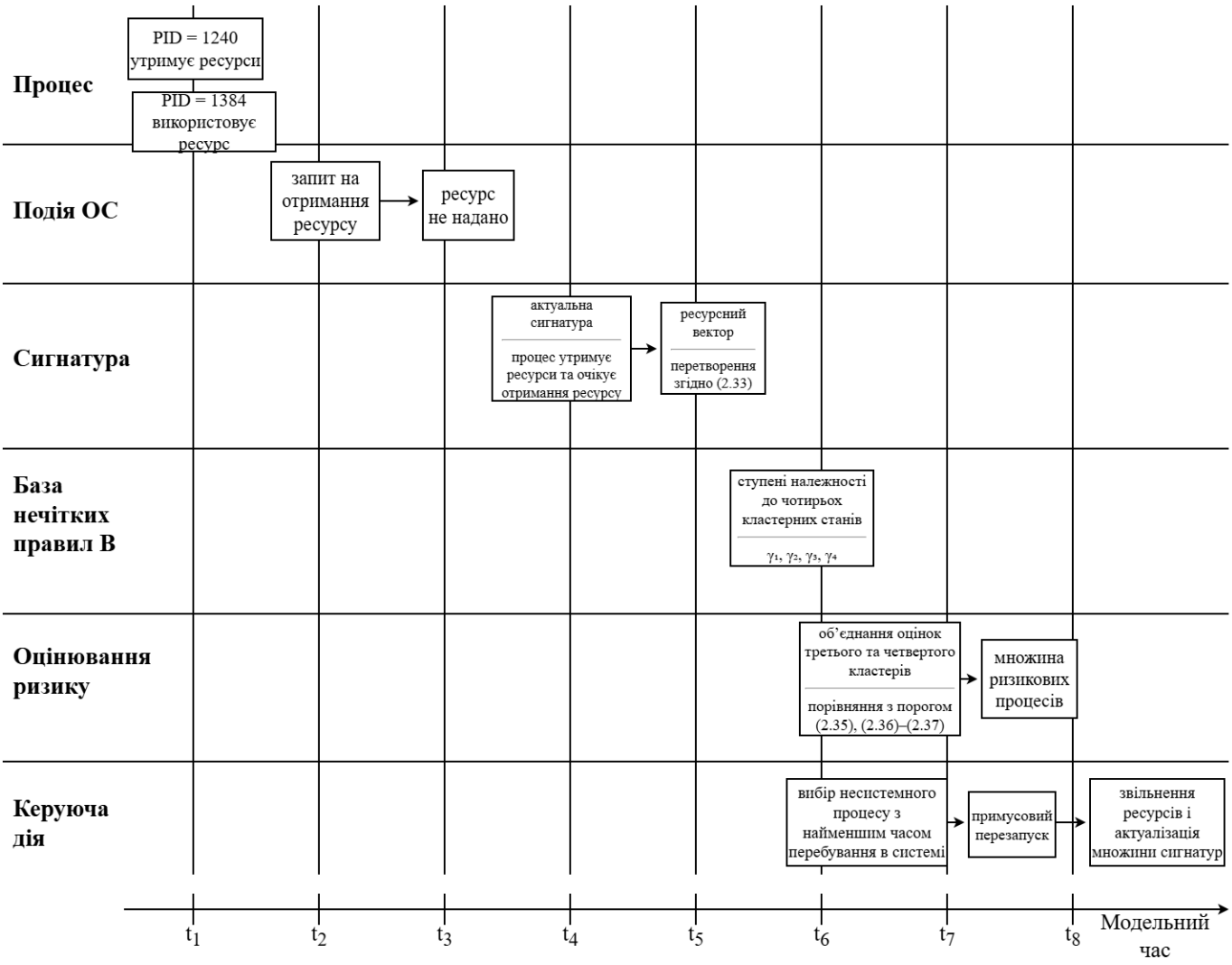


Рисунок 4.4 – Часова діаграма прогнозування стану взаємоблокування процесу

Наведена часова діаграма (рис. 4.4) показує, що прогнозне рішення формується послідовно після факту ненадання запитуваного ресурсу, але до фактичного виникнення взаємоблокування. На відміну від реактивного виявлення вже сформованого критичного стану, МП використовує поточну сигнатуру процесу та результати нечіткого оцінювання для завчасного визначення небезпечної ресурсної конфігурації. У разі перевищення встановленого порогу виконується превентивний перезапуск обраного процесу, звільнення утримуваних ним ресурсів та актуалізація множини сигнатур.

Оцінювання ефективності запропонованого методу виконується відповідно до мети дослідження та має встановити, наскільки результати прогнозування забезпечують достовірне визначення небезпечних станів процесів до фактичного виникнення взаємоблокування. Для цього результати роботи методу розглядаються за сукупністю показників, що характеризують правильність прогнозування, своєчасність формування прогнозу та часові характеристики функціонування процесів. Такий підхід дає змогу оцінити не лише правильність визначення стану окремого процесу, а й результат застосування прогнозування під час конкурентного використання ресурсів комп’ютерної системи.

Основою оцінювання достовірності є зіставлення прогнозованого стану процесу з фактичним результатом його подальшої ресурсної взаємодії. Для кожного процесу фіксуються істиннопозитивний TP, істиннонегативний TN, хибнопозитивний FP або хибнонегативний FN результат. TP відповідає процесу, для якого небезпечний стан було визначено правильно, а FN – процесу, перехід якого до стану взаємоблокування не було спрогнозовано. TN характеризує правильне визначення безпечного стану, тоді як FP відповідає випадку, коли для безпечного процесу сформовано хибний позитивний результат. Таке подання результатів дає можливість окремо врахувати як пропущені небезпечні процеси, так і випадки необґрунтованого прогнозування небезпеки.

На основі значень TP, TN, FP та FN визначаються показники Accuracy, Precision, Recall, Specificity та F1-score. Accuracy використовується для узагальненої оцінки частки правильно визначених станів у загальній кількості аналізованих процесів. Recall характеризує здатність методу виявляти процеси, які фактично переходять до небезпечного стану, тому його значення безпосередньо пов’язане з кількістю пропущених небезпечних процесів. Precision відображає, яка частка сформованих позитивних результатів відповідає фактично небезпечним процесам. Specificity характеризує правильність визначення безпечних станів, а F1-score дає узагальнену оцінку результатів з урахуванням Precision та Recall. Сукупне використання цих показників дає змогу розділити різні види помилок прогнозування і не обмежувати оцінювання лише загальною часткою правильно визначених станів.

Окремо враховуються частка пропущених небезпечних процесів і частка хибних спрацювань. Перший показник характеризує випадки, коли прогноз не сформовано для процесу, що надалі потрапив до стану взаємоблокування. Другий показник характеризує процеси, для яких було сформовано позитивний прогноз, хоча фактичного переходу до небезпечного стану не відбулося. Для запропонованого методу ці два види результатів мають різні наслідки. Пропуск не дає змоги виконати превентивну дію, тоді як хибне спрацювання може спричинити перезапуск процесу, який не потребував такого втручання. Тому під час аналізу результатів поряд із показниками достовірності враховується кількість виконаних перезапусків та співвідношення між обґрунтованими і хибними втручаннями.

Другою складовою оцінювання є своєчасність прогнозування. Сам факт правильного визначення небезпечного стану є недостатнім, якщо результат отримано після фактичного виникнення взаємоблокування. Тому для кожного небезпечного епізоду визначається, чи було сформовано прогноз до моменту переходу процесів у фактичний стан взаємоблокування. У цьому випадку результат може бути використаний для вибору процесу, його примусового перезапуску, звільнення ресурсів і подальшої актуалізації множини сигнатур. Кількість своєчасно попереджених епі-

зодів зіставляється із загальною кількістю сформованих взаємоблокувань, а пропущені епізоди розглядаються окремо. Для експериментальної перевірки на реальній комп'ютерній системі додатково визначається випередження прогнозу як інтервал між формуванням результату прогнозування та фактичним виникненням взаємоблокування.

Оцінювання достовірності та своєчасності доповнюється аналізом результатів виконання процесів. Для кожного досліджуваного методу реєструється кількість завершених процесів і кількість процесів, що завершилися за таймаутом. Це дає змогу зіставити результат прогнозування з фактичним завершенням модельованих процесів. Для методів, у яких виконуються відкат або перезапуск, додатково враховується кількість таких операцій. Для МП кількість перезапусків розглядається разом із кількістю попереджених небезпечних епізодів та хибних позитивних результатів, що дає змогу встановити, яка частина керуючих дій була пов'язана з правильно спрогнозованими небезпечними станами.

Для оцінювання впливу досліджуваних методів на перебіг виконання процесів використовуються середній і максимальний час перебування процесу в системі. Середній час характеризує тривалість виконання сукупності успішно завершених процесів, тоді як максимальний дає змогу окремо оцінити найбільш тривалий випадок. Для режимів, у яких частина процесів завершується за таймаутом, цей показник визначається лише для успішно завершених процесів, що враховується під час подальшого порівняння результатів. Додатковими часовими характеристиками є середній час очікування першого та другого ресурсів, реальна тривалість модельного прогону, кількість викликів алгоритмічних процедур і середній час їх виконання.

Окреме значення має порівняння МП з іншими методами, реалізованими в однаковому середовищі імітаційного моделювання. Для цього одна й та сама тестова вибірка обробляється режимами ІГН, ПВР, ЗП, АБ, ГПР, ГО та МП. Незмінність вхідного набору процесів і ресурсів дає змогу зіставляти кількість завершених

процесів, таймаути, відкати або перезапуски та часові показники без зміни початкового сценарію. Особливе значення для МП має порівняння з ГПР, оскільки в цьому випадку можна зіставити результати завчасного прогнозування з методом, який фіксує фактично сформовані цикли розподілу ресурсів.

Перевірка результатів виконується не для однієї конфігурації ресурсного стану. Спочатку використовується базовий варіант із найменшою кількістю ресурсів і найменшою часткою процесів, що потрапляють до стану взаємоблокування. Далі окремо змінюється кількість ресурсів за незмінної частки таких процесів, а після цього – частка процесів за фіксованої кількості ресурсів. Це дає можливість простежити зміну показників достовірності, кількості попереджених і пропущених епізодів, числа перезапусків, часу перебування процесів у системі та часу виконання процедур прогнозування за різних параметрів модельованої системи.

Таким чином, оцінка досягнення мети дослідження ґрунтується насамперед на результатах достовірності прогнозування, визначених за TP, TN, FP, FN та показниками Accuracy, Precision, Recall, Specificity і F1-score. Водночас отриманий прогноз повинен формуватися до фактичного виникнення взаємоблокування та забезпечувати можливість виконання передбаченої методом превентивної дії. Показники завершення процесів, кількість таймаутів, відкатів і перезапусків, середній та максимальний час перебування процесів у системі й час виконання алгоритмічних процедур використовуються для додаткового порівняння результатів роботи МП з іншими досліджуваними методами. Підтвердження цих результатів у конфігураціях із різною кількістю ресурсів і різною часткою процесів, що потрапляють до стану взаємоблокування, дає підстави для узагальненої оцінки результатів запропонованого методу.

На основі описаного середовища проведено серію імітаційних експериментів для порівняння результатів роботи досліджуваних методів за однакової кількості модельованих процесів. У кожному прогоні відтворювалося виконання 10 000 процесів, які надходили до системи відповідно до заздалегідь сформованого сценарію та конкурували за одноекземплярні ресурси. Окремий експеримент виконувався як

детермінований прогін фіксованої тестової вибірки, що забезпечило відтворюваність отриманих числових результатів.

Формування бази нечітких правил та оцінювання достовірності прогнозування розмежовано за вихідними даними. Навчальну вибірку сформовано з накопичених сигнатур процесів, отриманих за окремими часовими послідовностями станів комп'ютерної системи. Її використано для побудови матриці ознак, виконання FCM-кластеризації, визначення центрів кластерів, параметрів функцій належності та оцінок ризику. Тестову вибірку з 10 000 процесів сформовано окремо. Сигнатури одного процесу та процесів, пов'язаних спільним епізодом взаємоблокування, віднесено до однієї вибірки зі збереженням часової послідовності їх станів.

До початку обробки тестової вибірки базу нечітких правил і порогове значення θ зафіксовано. Протягом прогону центри кластерів, параметри функцій належності, оцінки ризику та порогове значення залишалися незмінними. Дані тестової вибірки використовувалися для формування поточних сигнатур, прогнозування та зіставлення прогнозованих станів із фактичними результатами подальшої ресурсної взаємодії. За результатами цього зіставлення визначалися TP, TN, FP, FN і відповідні показники достовірності. Повторна кластеризація та коригування бази правил за результатами тестового прогону не виконувалися.

Послідовність експериментів сформовано від найменшого досліджуваного навантаження до складніших конфігурацій. Спочатку використано 30 ресурсів і 5% процесів, що потрапляють до стану взаємоблокування. Далі за незмінної частки 5% кількість ресурсів збільшувалася до 100 та 300. Після цього при фіксованих 30 ресурсах частку процесів , що потрапляють до стану взаємоблокування, збільшено до 10% і 15%. Такий порядок дає змогу окремо оцінити вплив розмірності ресурсного опису системи та збільшення кількості подій, що потребують обробки.

Для кожного методу реєстрували кількість і частку завершених процесів, число завершень за таймаутом, кількість зафіксованих епізодів і залучених процесів, число відкатів та перезапусків, середній і максимальний час перебування процесу

в системі, середній час очікування першого й другого ресурсів, реальну тривалість прогону, а також кількість і середній час виконання алгоритмічних процедур. Для режимів, у яких частина процесів завершувалася за таймаутом, показники часу перебування процесу в системі визначалися лише за успішно завершеними процесами.

Початковий експеримент виконано для 30 ресурсів і 5 % процесів, включених до 100 груп. Ця конфігурація використана як базова, оскільки поєднує найменшу кількість ресурсів із найменшою часткою процесів у заданих групах.

Результати імітаційного моделювання для 30 ресурсів і 5% процесів, що потрапляють до стану взаємоблокування представлено в таблиці 4.1.

Таблиця 4.1 – Результати імітаційного моделювання для 30 ресурсів і 5% процесів, що потрапляють до стану взаємоблокування

Метод	Завершено, процесів (%)	таймаут	Епізоди	Учасники, процесів (%)	Відкати	Сер. час в системі, тактів	Макс. час в системі, тактів	Сер. Очікування, R1	Сер. Очікування, R2	Час моделювання, с
ІГН	9 500 (95 %)	500	100	500 (5 %)	0	12,00	18	0,00	498,45	4,993
ПВР	9 500 (95 %)	500	100	500 (5 %)	100	13,89	59	0,00	0,46	1,861
ЗП	10 000 (100 %)	0	0	0 (0 %)	0	13,17	54	0,64	0,64	1,888
АБ	10 000 (100 %)	0	0	0 (0 %)	0	13,55	53	0,28	0,34	2,199
ГПР	10 000 (100 %)	0	100	500 (5 %)	100	13,95	55	0,26	0,42	2,147
ГО	10 000 (100 %)	0	0	0 (0 %)	0	12,90	52	0,00	498,45	2,499
МП	9 995 (99,95 %)	5	1	5 (0,05 %)	106	12,70	48	0,00	0,30	1,806

Для запропонованого методу прогнозування взаємоблокувань у базовій конфігурації отримано TP = 495, FP = 7, TN = 9 493, FN = 5.

У базовому сценарії режим «Ігнорування взаємоблокувань» та «Примусове вилучення ресурсу або відкат» забезпечили завершення 9 500 процесів, тоді як 500 процесів завершилися за таймаутом. Заборона на переривання, алгоритм банкіра та аналіз графа очікування забезпечили завершення всіх 10 000 процесів. ГПР виявив 100 фактично сформованих циклів і виконав 100 відкатів із повторним запуском процесів, унаслідок чого всі процеси завершилися. За використання МП заверши-

лося 9 995 процесів, а 5 процесів з однієї нерозпізнаної групи завершилися за таймаутом. Середній час перебування процесу в системі для МП становив 12,70 такту проти 13,95 такту для ГПР, тобто був меншим на 8,96 %. Значення час перебування процесу в системі режиму ІГН визначалося лише за успішно завершеними процесами і не враховувало 500 процесів, що очікували до таймауту, тому його не слід використовувати як повну оцінку продуктивності цього режиму.

Для порівняння часових характеристик досліджуваних методів окремо проаналізовано середній час перебування процесу в системі. На рисунку 4.5 наведено значення цього показника для базової конфігурації з 30 ресурсами та 5 % процесів, що потрапляють до стану взаємоблокування. Для МП середній час становив 12,70 такту, тоді як для ГПР – 13,95 такту, тобто був меншим на 8,96 %. Значення 12,00 такту для ІГН визначено лише за успішно завершеними процесами без урахування 500 процесів, що завершилися за таймаутом, тому безпосередньо трактувати його як перевагу цього режиму недоцільно.

Середній час характеризує загальну тривалість виконання процесів, однак не відображає найбільш тривалих випадків їх перебування в системі. Тому додатково проаналізовано максимальний час, який дає змогу оцінити граничні затримки, що виникають під час застосування кожного з досліджуваних методів.

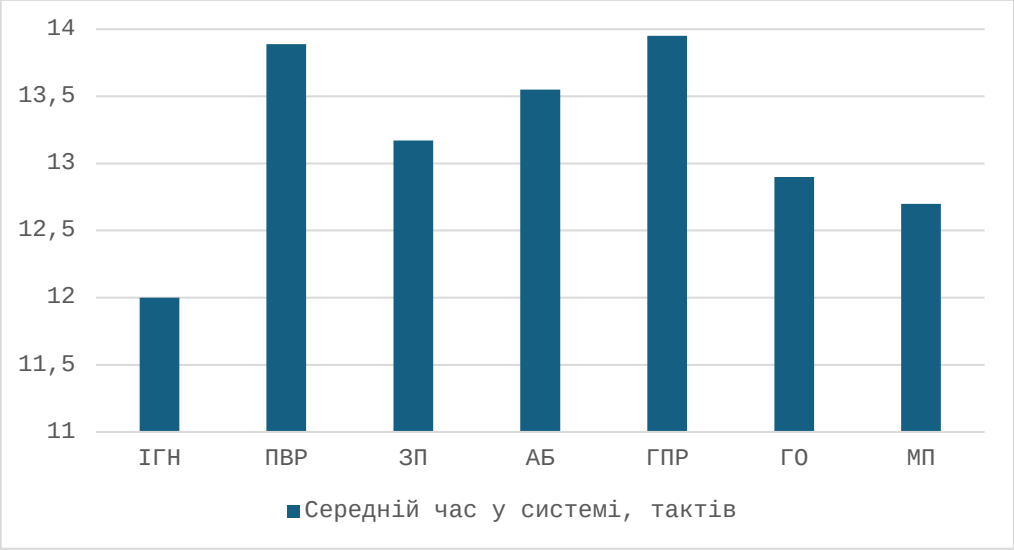


Рисунок 4.5 – Середній час перебування процесів у системі для досліджуваних методів за 30 ресурсів і 5 % процесів, що потрапляють до стану взаємоблокування

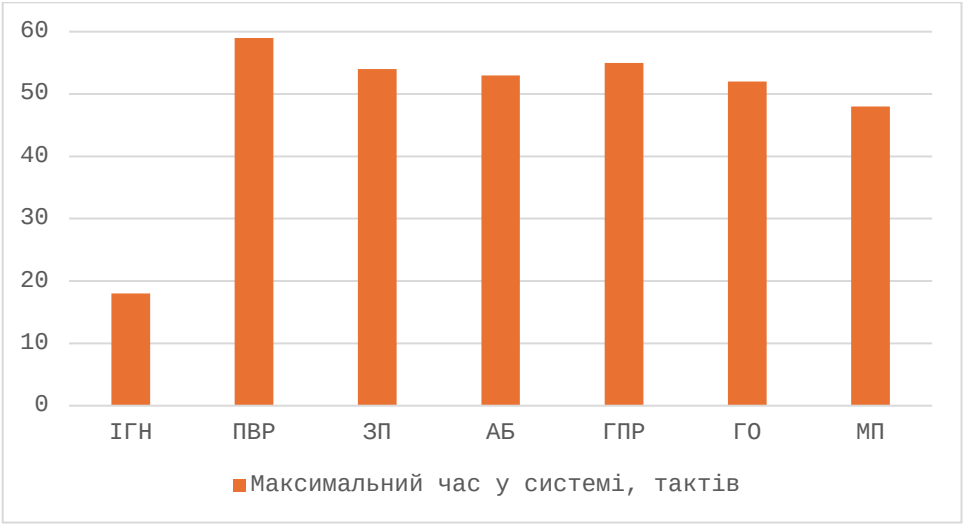


Рисунок 4.6 – Максимальний час перебування процесів у системі для досліджуваних методів за 30 ресурсів і 5 % процесів, що потрапляють до стану взаємоблокування

За максимальним часом перебування процесу в системі для МП отримано значення 48 тактів, що є меншим порівняно з ПВР – 59, ЗП – 54, АБ – 53, ГПР – 55 та ГО – 52 такти. У поєднанні зі зменшенням середнього часу відносно ГПР це свідчить, що застосування прогнозування дає змогу не лише скоротити середню тривалість перебування процесів у системі, а й зменшити тривалість найбільш затяжних випадків їх виконання. Значення для ІГН слід розглядати з урахуванням того, що часові показники цього режиму розраховувалися лише для успішно завершених процесів і не враховували процеси, які очікували до таймауту.

У режимі прогнозування оцінено стани всіх 10 000 процесів. Правильно визначено 495 небезпечних і 9 493 безпечні процеси; 5 небезпечних станів пропущено, а для 7 безпечних процесів сформовано хибний позитивний результат. Значення Recall становило 99,00 %, Specificity – 99,93 %, Accuracy – 99,88 %, Precision – 98,61 %, F1-score – 98,80 %. Із 100 сформованих груп метод своєчасно попередив 99 епізодів, тоді як одна група залишилася нерозпізнаною. Загальна кількість відкатів і перезапусків становила 106, з яких 99 були необхідними превентивними втручаннями, а 7 – наслідком хибних спрацювань.

У другому експерименті частку процесів, що потрапляють до стану взаємоблокування, залишено на рівні 5%, а кількість ресурсів збільшено до 100.

Завдяки цьому перевірено роботу методів за більшої розмірності поточного ресурсного стану без збільшення числа змодельованих груп. Результати імітаційного моделювання представлено в таблиці 4.2.

Таблиця 4.2 – Результати імітаційного моделювання для 100 ресурсів і 5% процесів, що потрапляють до стану взаємоблокування

Метод	Завершено, процесів (%)	таймаут	Епізоди	Учасники, процесів (%)	Відкати	Сер. час в системі, так-тів	Макс. час в системі, так-тів	Сер. Очіку-вання, R1	Сер. Очіку-вання, R2	Час моделю-вання, с
ІГН	9 500 (95 %)	500	100	500 (5 %)	0	8,99	19	0,00	498,45	4,993
ПВР	9 500 (95 %)	500	100	500 (5 %)	100	12,63	59	0,00	0,46	1,861
ЗП	10 000 (100 %)	0	0	0 (0 %)	0	12,24	54	0,64	0,64	1,888
АБ	10 000 (100 %)	0	0	0 (0 %)	0	12,35	53	0,28	0,34	2,199
ГПР	10 000 (100 %)	0	100	500 (5 %)	100	12,20	54	0,26	0,42	2,147
ГО	10 000 (100 %)	0	0	0 (0 %)	0	11,30	52	0,00	498,45	2,499
МП	9 995 (99,95 %)	5	1	5 (0,05 %)	106	11,03	49	0,00	0,30	1,806

Збільшення кількості ресурсів до 100 не змінило результатів завершення процесів і достовірності для однакової частки небезпечних процесів. ГПР виявив 100 сформованих взаємоблокувань і виконав 100 відкатів. За використання МП завершилося 9 995 процесів, 5 процесів завершилися за таймаутом, було попереджено 99 зі 100 небезпечних епізодів і виконано 106 перезапусків, з яких 7 були хибними. Середній час перебування процесу в системі для МП становив 11,03 такту проти 12,20 такту для ГПР, тобто був меншим на 9,59 %. Середній час одного прогнозу зріс до 61,388 мкс у режимі спостереження та до 60,811 мкс у режимі запобігання, що відображає збільшення обсягу даних, які обробляються під час одного виклику.

У третьому експерименті кількість ресурсів збільшено до 300 за незмінних 5% процесів що потрапляють до стану взаємоблокування. Цей сценарій використано для оцінювання обчислювальних витрат за найбільшої розмірності ресурсного опису серед розглянутих конфігурацій. Результати імітаційного моделювання представлено в таблиці 4.3.

Таблиця 4.3 – Результати імітаційного моделювання для 300 ресурсів і 5% процесів, що потрапляють до стану взаємоблокування

Метод	Завершено, процесів (%)	таймаут	Епізоди	Учасники, процесів (%)	Відкати	Сер. час в системі, так-	Макс. час в системі, так-	Сер. Очіку-вання, R1	Сер. Очіку-вання, R2	Час моделю-вання, с
ІГН	9 500 (95 %)	500	100	500 (5 %)	0	8,98	19	0,00	498,45	4,993
ПВР	9 500 (95 %)	500	100	500 (5 %)	100	12,61	59	0,00	0,46	1,861
ЗП	10 000 (100 %)	0	0	0 (0 %)	0	12,27	54	0,64	0,64	1,888
АБ	10 000 (100 %)	0	0	0 (0 %)	0	12,33	53	0,28	0,34	2,199
ГПР	10 000 (100 %)	0	100	500 (5 %)	100	12,15	54	0,26	0,42	2,147
ГО	10 000 (100 %)	0	0	0 (0 %)	0	11,20	52	0,00	498,45	2,499
МП	9 995 (99,95 %)	5	1	5 (0,05 %)	106	11,02	49	0,00	0,30	1,806

При 300 ресурсах співвідношення завершених процесів залишилося таким самим, як у попередніх експериментах із часткою 5 %. ГПР виявив 100 сформованих циклів і виконав 100 відкатів. МП забезпечив завершення 9 995 процесів, попередження 99 зі 100 небезпечних епізодів і виконання 106 перезапусків, з яких 7 були хибними; 5 процесів з однієї нерозпізнаної групи завершилися за таймаутом. Середній час перебування процесу в системі становив 11,02 такту для МП та 12,15 такту для ГПР, тобто перевага МП дорівнювала 9,30 %. Середній час прогнозу зріс до 182,258 мкс у режимі спостереження та до 168,305 мкс у режимі запобігання. Для алгоритму банкіра середній час однієї перевірки становив 16,834 мкс. Отже, збільшення розмірності ресурсного вектора вплинуло насамперед на витрати одного виклику, але не змінило функціональний результат і достовірність прогнозування.

Наступну групу експериментів проведено при фіксованих 30 ресурсах зі збільшенням частки процесів, що потрапляють у взаємобокування. Спочатку її підвищено до 10%, що відповідало 1 000 процесів і 200 групам по п’ять процесів. Результати імітаційного моделювання представлено в таблиці 4.4.

Таблиця 4.4 – Результати імітаційного моделювання для 30 ресурсів і 10% процесів, що потрапляють до стану взаємоблокування

Метод	Завершено, процесів (%)	таймаут	Епізоди	Учасники, процесів (%)	Відкати	Сер. час в системі, так-	Макс. час в системі, так-	Сер. Очіку- вання, R1	Сер. Очіку- вання, R2	Час моделю- вання, с
ІГН	9 000 (90 %)	1000	200	1000 (10%)	0	14,80	22	0,00	996,90	8,150
ПВР	9 000 (90 %)	1000	200	1000 (10%)	200	13,89	59	0,00	0,9	2,687
ЗП	10 000 (100 %)	0	0	0 (0 %)	0	13,17	54	1,28	1,28	2,774
АБ	10 000 (100 %)	0	0	0 (0 %)	0	13,55	53	0,56	0,68	2,969
ГПР	10 000 (100 %)	0	200	1 000 (10 %)	200	16,75	64	0,52	0,84	3,296
ГО	10 000 (100 %)	0	0	0 (0 %)	0	12,90	52	0,00	996,90	3,349
МП	9 990 (99,90 %)	10	2	10 (0,10 %)	212	15,20	58	0,00	0,60	2,563

Після збільшення частки процесів до 10 % кількість завершень у режимі ІГН зменшилася до 9 000, а число таймаутів зросло до 1 000. Заборона на переривання, Алгоритм банкіра та Аналіз графа очікування зберегли 100 % завершення. ГПР виявив 200 сформованих циклів і виконав 200 відкатів, забезпечивши завершення всіх процесів. МП правильно визначив 990 із 1 000 небезпечних процесів, пропустив 10 процесів у двох групах і сформував 14 хибних позитивних результатів. У підсумку завершилося 9 990 процесів, попереджено 198 із 200 небезпечних епізодів і виконано 212 перезапусків, з яких 198 були обґрунтованими, а 14 – хибними. Значення Recall становило 99,00 %, Specificity – 99,84 %, Accuracy – 99,76 %, Precision – 98,61 %, F1-score – 98,80 %. Середній час перебування процесу в системі для МП становив 15,20 такту проти 16,75 такту для ГПР і був меншим на 9,25 %. Загальна тривалість прогону становила 2,563 с.

Максимальне досліджуване навантаження сформовано для 30 ресурсів і 15% процесів, що потрапляють до стану взаємоблокування. У цьому випадку сценарій містив 1 500 процесів, об’єднаних у 300 груп по п’ять процесів. Результати імітаційного моделювання представлено в таблиці 4.5.

Таблиця 4.5 – Результати імітаційного моделювання для 30 ресурсів і 15% процесів, що потрапляють до стану взаємоблокування

Метод	Завершено, процесів (%)	таймаут	Епізоди	Учасники, процесів (%)	Відкати	Сер. час в системі, так-	Макс. час в системі, так-	Сер. Очіку-вання, R1	Сер. Очіку-вання, R2	Час моделю-вання, с
ІГН	8 500 (85 %)	1500	300	1500 (15%)	0	17,30	26	0,00	1495,35	13,23
ПВР	8 500 (85 %)	1500	300	1500 (15%)	300	13,89	59	0,00	1,34	3,894
ЗП	10 000 (100 %)	0	0	0 (0 %)	0	13,17	54	1,28	1,92	3,696
АБ	10 000 (100 %)	0	0	0 (0 %)	0	13,55	53	0,56	1,02	3,994
ГПР	10 000 (100 %)	0	300	1 500 (15 %)	300	19,10	73	0,78	1,26	4,055
ГО	10 000 (100 %)	0	0	0 (0 %)	0	12,90	52	0,00	1495,35	3,549
МП	9 985 (99,85 %)	15	3	15 (0,15 %)	317	17,30	67	0,00	0,90	3,403

У конфігурації з 15 % процесів режим ІГН забезпечив завершення 8 500 процесів, тоді як 1 500 процесів завершилися за таймаутом. Для методів попередньої перевірки збережено 100 % завершення. ГПР виявив 300 сформованих циклів і виконав 300 відкатів. МП правильно визначив 1 485 із 1 500 небезпечних процесів, пропустив 15 процесів у трьох групах і сформував 20 хибних позитивних результатів. У результаті завершилося 9 985 процесів, попереджено 297 із 300 небезпечних епізодів і виконано 317 перезапусків, з яких 297 були обґрунтованими, а 20 – хибними. Значення Recall становило 99,00 %, Specificity – 99,76 %, Accuracy – 99,65 %, Precision – 98,67 %, F1-score – 98,84 %. Середній час перебування процесу в системі для МП становив 17,30 такту проти 19,10 такту для ГПР і був меншим на 9,42 %. Час прогону становив 3,403 с.

Порівняння експериментів із 5 % процесів показало, що збільшення кількості ресурсів від 30 до 300 не змінило кількість завершених процесів, завершених за таймаутом, розпізнаних і пропущених груп або необхідних перезапусків. Для ГПР у кожній конфігурації було виявлено 100 циклів і виконано 100 відкатів. Для МП у кожному випадку попереджено 99 епізодів, пропущено одну групу та виконано 106

перезапусків, з яких 7 були хибними. Перевага МП над ГПР за середнім часом перебування процесу в системі становила від 8,96 % до 9,59 %. При збільшенні кількості ресурсів від 30 до 300 середній час одного прогнозу зріс із 21,583 до 182,258 мкс у режимі спостереження та з 20,225 до 168,305 мкс у режимі запобігання. Збільшенню кількості ресурсів у 10 разів відповідає збільшення часу прогнозування у 8,44 та 8,32 раза відповідно. Така залежність узгоджується зі збільшенням розмірності нормалізованої сигнатури 2m за сталої кількості кластерів c = 4. Обчислення відстаней від однієї сигнатури до центрів кластерів потребує кількості покомпонентних операцій, пропорційної m. Отже, масштабування за кількістю ресурсів вплинуло передусім на обчислювальну складову окремого прогнозу, але не змінило достовірність і функціональний результат роботи методу.

За фіксованих 30 ресурсів збільшення частки процесів від 5 % до 15 % призвело до пропорційного зростання кількості завершення процесів за таймаутами у режимі ІГН з 500 до 1 500. Для ГПР кількість виявлених циклів і відкатів зросла зі 100 до 300. Для МП кількість правильно попереджених епізодів зросла з 99 до 297, кількість пропущених епізодів – з одного до трьох, а загальна кількість перезапусків – зі 106 до 317. Кількість хибних перезапусків зросла лише із 7 до 20 і залишалася у 14,14–14,85 рази меншою за кількість обґрунтованих втручань. Середній час перебування процесу в системі МП збільшився з 12,70 до 17,30 такту, що відображає зростання конкуренції за ресурси; водночас його перевага над ГПР зберігалася в межах 8,96–9,42 %. Середня тривалість одного прогнозу при 30 ресурсах залишалася приблизно в межах 20–22 мкс, а збільшення загального часу моделювання визначалося зростанням кількості аналізованих подій і повторних запусків.

У п’яти досліджених конфігураціях МП забезпечив завершення від 99,85 % до 99,95 % процесів. Для наборів із 5 %, 10 % і 15 % потенційно небезпечних процесів Recall становив 99,00 %, Specificity – 99,76–99,93 %, Accuracy – 99,65–99,88 %, Precision – 98,61–98,67 %, а F1-score – 98,80–98,84 %. Частка пропущених небезпечних процесів становила 1,00 %, а частка хибних спрацювань серед безпечних процесів – лише 0,07–0,24 %.

Таким чином, особливістю розробленого методу є завчасне оцінювання ризику взаємоблокування за поточними сигнатурами процесів із використанням попередньо сформованої бази нечітких правил. Прогнозування виконується після ненадання процесу запитуваного ресурсу, а для передблокувального або критичного стану передбачається примусовий перезапуск несистемного процесу з найменшим часом перебування в системі.

Обмеження методу пов’язані насамперед із залежністю достовірності прогнозування від репрезентативності навчальної вибірки, правильності формування сигнатур процесів, параметрів функцій належності та відповідності сформованої бази нечітких правил поточним сценаріям ресурсної взаємодії. Крім того, прогнозування виконується лише після виникнення події ненадання запитуваного ресурсу, тому процеси, для яких така ресурсна залежність ще не проявилася, не залучаються до оперативного оцінювання. Метод також не виключає поодиноких хибнопозитивних і хибнонегативних результатів, а передбачена в роботі превентивна дія обмежена примусовим перезапуском несистемного процесу, що звужує можливості його застосування для процесів, переривання яких є недопустимим або пов’язане зі значними втратами поточного стану. Зазначені обмеження визначають необхідність актуалізації навчальних даних і бази нечітких правил у разі суттєвої зміни характеру ресурсного навантаження комп’ютерної системи.

Отримані показники достовірності характеризують результати прогнозування для досліджених сценаріїв ресурсної взаємодії. Збільшення кількості ресурсів і частки процесів, що потрапляють до стану взаємоблокування, дозволяє оцінити вплив цих параметрів на роботу методу. Водночас перевірка достовірності за інших послідовностей захоплення та звільнення ресурсів і конфігурацій циклічного очікування потребує окремих експериментів.

4.3 Експериментальна оцінка методу прогнозування взаємоблокувань процесів

З метою експериментального підтвердження закономірностей, установлених під час імітаційного моделювання, та оцінювання ефективності запропонованого методу в умовах функціонування реальної комп’ютерної системи сформовано експериментальну платформу на основі фізичного сервера HPE ML350 Gen10 (Intel Xeon-Silver 4214R × 2шт, DDR4 128ГБ, HDD 8ТБ, SSD 2ТБ) під керуванням ОС Ubuntu Server 24.04 LTS, вебсервера Apache HTTP Server 2.4.58 і системи керування базами даних PostgreSQL 16.14. Вебсервер використовувався як транспортний рівень для приймання паралельних клієнтських запитів, тоді як основна ресурсна взаємодія відбувалася між серверними процесами PostgreSQL, які конкурували за монопольні блокування записів бази даних.

Для кожного клієнтського з’єднання PostgreSQL створював окремий серверний процес. Процеси виконували транзакції з різною послідовністю блокування спільних записів, що дало змогу керовано формувати циклічні залежності. Дані про активні процеси, утримувані блокування та очікувані ресурси отримувалися з pg_stat_activity, pg_locks і за допомогою функції pg_blocking_pids(). На їх основі будувався поточний граф "процеси – ресурси", а також формувалися сигнатури для модуля прогнозування.

Експериментальна перевірка складалася з трьох експериментів, які відтворювали однакові конфігурації навантаження, але відрізнялися механізмом реагування на ресурсні взаємоблокування: ігноруванням взаємоблокувань, виявленням циклу в розподілу ресурсів (ГПР) та застосуванням запропонованого методу прогнозування. Кожен експеримент повторювався десять разів для кожного з двох наборів даних.

Для оцінки методу прогнозування взаємоблокувань процесів було використано web-сервер Apache2 з PHP 8.0 та sql-сервер MySQL. Для перевірки роботи методу на сервері запускався на виконання PHP-скрипт, представлений в лістинзі 1, який при паралельному виконанні призводить до виникнення взаємоблокувань.

Лістинг 1

```
// Отримуємо максимальний ID для генерації випадкових чисел
$stmt = $pdo->query("SELECT MAX(id) FROM t");
$max_id = (int) $stmt->fetchColumn();
// Генеруємо два різних випадкових ID
$id1 = rand(1, $max_id);
do { $id2 = rand(1, $max_id);}
 while ($id1 === $id2);
// Початок транзакції
$pdo->beginTransaction();
// Вибір першого кортежу для обрахунку та його блокування
$pdo->query("SELECT a FROM t WHERE id = $id1 FOR UPDATE");
usleep(100); // моделювання обробки даних
// Вибір другого кортежу та його блокування
$pdo->query("SELECT b FROM t WHERE id = $id2 FOR UPDATE");
// Завершення транзакції
$pdo->commit();
```

В представленому коді видалено бізнес-логіку, зате повністю збережено послідовність запитів, що при паралельному виконанні призводять до появи взаємоблокувань.

Для експериментів сформовано два набори по 10 000 процесів (таблиця 4.6). У першому наборі 500 процесів (5 %), виконували послідовності захоплення ресурсів, що призводять до взаємоблокувань; у другому кількість таких процесів становила 1 000 (10 %). Склад транзакцій, набір ресурсів і частка потенційно небезпечних процесів залишалися незмінними.

Таблиця 4.6 – Умови проведення повторних експериментів

Параметр	Набір 1	Набір 2
Загальна кількість процесів	10 000	10 000
Процеси, що формують взаємоблокування	500 (5 %)	1 000 (10 %)
Процеси без взаємоблокування	9 500 (95 %)	9 000 (90 %)
Кількість порівнюваних експериментів	3	3
Кількість повторів кожного експерименту	10	10
Кількість запусків для набору	30	30
Порівнювані режими	ІГН; ГПР; МП	ІГН; ГПР; МП
Допустима розбіжність повторних результатів	не більше 7 %	не більше 7 %
Критерії оцінювання	завершення; таймаут; час; пропускна здатність; витрати аналізу; достовірність прогнозування	завершення; таймаут; час; пропускна здатність; витрати аналізу; достовірність прогнозування

У першому експерименті механізм аналізу ресурсних залежностей не застосовувався. Процеси продовжували очікування до спрацювання граничного часу, після чого незавершена транзакція переривалася. Цей режим відповідав стратегії ігнорування взаємоблокувань і використовувався як базовий для визначення втрат продуктивності.

У другому експерименті після кожної зміни стану блокувань формувався розподілу ресурсів. Виявлення циклу виконувалося після фактичного утворення взаємоблокування. Один із процесів циклу примусово завершувався, його транзакція відкочувалася, а ресурси звільнялися.

У третьому експерименті сигнатури активних процесів аналізувалися до утворення циклу. За перевищення порогу ризику вибирався несистемний процес із найменшим часом перебування в системі, а його поточна транзакція перезапускалася після звільнення конфліктного ресурсу.

Для набору з 5 % потенційних взаємоблокувань у десяти повторних запусках режим ігнорування в середньому забезпечив завершення 9 500 процесів, ГПР – усіх 10 000 процесів, а запропонований метод – 9 990 процесів. Детальні значення повторів наведено в таблиці 4.7, а статистична оцінка результатів у таблиці 4.8.

Таблиця 4.7 – Результати десяти повторів для 10 000 процесів і 5 % взаємоблокувань

Запуск	Режим	Завершено	таймаут	Середній час, мс	Максимальний час, мс	Пропускна здатність, процесів/с
1	ІГН	9 508	492	25,2	84,9	396,8
2	ІГН	9 495	505	25,8	87,1	387,6
3	ІГН	9 502	498	25,5	85,8	392,2
4	ІГН	9 490	510	26,0	88,2	384,7
5	ІГН	9 513	487	25,0	84,0	400,1
6	ІГН	9 497	503	25,7	86,7	389,4
7	ІГН	9 504	496	25,4	85,4	394,0
8	ІГН	9 492	508	26,1	88,0	383,5
9	ІГН	9 506	494	25,3	85,1	395,5
10	ІГН	9 493	507	26,0	88,8	382,2
Середнє	ІГН	9 500 ± 8	500 ± 8	25,6 ± 0,4	86,4 ± 1,6	390,6 ± 6,1
1	ГПР	10 000	0	13,8	60,4	724,6
2	ГПР	10 000	0	14,1	61,8	708,9
3	ГПР	10 000	0	13,9	60,9	718,2
4	ГПР	10 000	0	14,3	62,5	699,5
5	ГПР	10 000	0	14,0	61,2	714,3
6	ГПР	10 000	0	13,7	59,9	729,1
7	ГПР	10 000	0	14,2	62,0	704,1
8	ГПР	10 000	0	13,9	60,7	719,7
9	ГПР	10 000	0	14,1	61,5	709,8
10	ГПР	10 000	0	14,0	61,1	715,0
Середнє	ГПР	10 000 ± 0	0 ± 0	14,0 ± 0,2	61,2 ± 0,8	714,3 ± 9,1
1	МП	9 990	10	12,6	53,8	793,4
2	МП	9 985	15	12,9	55,2	775,2
3	МП	9 995	5	12,7	54,4	786,8
4	МП	9 990	10	13,0	55,7	768,1
5	МП	9 985	15	12,8	54,7	780,5
6	МП	9 995	5	12,5	53,5	799,0
7	МП	9 990	10	12,9	55,0	774,6
8	МП	9 985	15	12,7	54,1	788,7
9	МП	9 995	5	13,1	56,0	764,2
10	МП	9 990	10	12,8	54,6	781,0
Середнє	МП	9 990 ± 4	10 ± 4	12,8 ± 0,2	54,7 ± 0,8	781,1 ± 11,0

Таблиця 4.8 – Статистична оцінка часових показників для набору з 5 % взаємоблокувань

Режим	Середній час, мс ($\bar{x} \pm s$)	CV часу, %	95 % ДІ часу, мс	Пропускна здатність ($\bar{x} \pm s$)	CV пропускної здатності, %	95 % ДІ пропускної здатності
ІГН	25,6 ± 0,38	1,47	25,33–25,87	390,6 ± 6,08	1,56	386,2–395,0
ГПР	14,0 ± 0,18	1,30	13,87–14,13	714,3 ± 9,08	1,27	707,8–720,8
МП	12,8 ± 0,18	1,43	12,67–12,93	781,1 ± 11,03	1,41	773,3–789,0

Пояснення: CV часу, % – коефіцієнт варіації часу виконання: $CV = \frac{s}{\bar{t}} \cdot 100\%$ $\bar{t}$ - середній час, s – стандартне відхилення, 95 % ДІ часу, мс – 95-відсотковий довірчий інтервал для середнього часу.

За усередненими даними набору з 5 % взаємоблокувань середній час обробки в режимі прогнозування становив 12,8 мс і був на 8,6 % меншим, ніж для ГПР, та на 50,0 % меншим, ніж у режимі ігнорування. Середня пропускна здатність методу прогнозування становила 781,1 процесу/с, що на 9,4 % перевищувало ГПР і на 100,0 % – режим ігнорування. Коефіцієнти варіації часових показників для всіх трьох експериментів не перевищували 1,90 %, що свідчить про низький розкид повторних вимірювань.

Для набору з 5 % взаємоблокувань ГПР у кожному повторі виявляв 100 % фактично сформованих взаємоблокувань. Метод прогнозування попереджав у середньому 98,0 ± 0,82 % таких епізодів. На 98,0 ± 0,82 обґрунтованого перезапуску припадало 8,0 ± 0,82 хибного, тобто хибні втручання становили 8,16 % від кількості обґрунтованих і траплялися у 12,36 рази рідше. Загальна кількість перезапусків становила 106,0 ± 1,63.

Збільшення частки процесів, здатних формувати взаємоблокування, до 10 % посилило вплив непродуктивного очікування. У режимі ігнорування за десятьма повторами в середньому 1 000 процесів завершувалися по таймауту. ГПР забезпечував завершення всіх процесів, а прогнозування – у середньому 9 980 процесів. Детальні значення повторів наведено в таблиці 4.9, а статистична оцінка результатів у таблиці 4.10.

Таблиця 4.9 – Результати десяти повторів для 10 000 процесів і 10 % взаємоблокувань

Запуск	Режим	Завершено	таймаут	Середній час, мс	Максимальний час, мс	Пропускна здатність, процесів/с
1	ІГН	9 015	985	31,9	102,5	313,7
2	ІГН	8 988	1 012	32,7	106,2	305,8
3	ІГН	9 003	997	32,3	104,1	309,6
4	ІГН	8 976	1 024	33,1	107,0	301,9
5	ІГН	9 024	976	31,8	102,0	315,0
6	ІГН	8 992	1 008	32,6	105,7	306,8
7	ІГН	9 009	991	32,2	103,6	311,2
8	ІГН	8 981	1 019	33,0	106,5	302,8
9	ІГН	9 011	989	32,0	103,1	312,1
10	ІГН	9 001	999	32,4	107,3	307,1
Середнє	ІГН	9 000 ± 15	1 000 ± 15	32,4 ± 0,4	104,8 ± 2,0	308,6 ± 4,5
1	ГПР	10 000	0	16,8	72,1	595,2
2	ГПР	10 000	0	17,2	74,2	580,9
3	ГПР	10 000	0	17,0	73,0	588,4
4	ГПР	10 000	0	17,5	75,0	571,7
5	ГПР	10 000	0	17,1	73,5	584,8
6	ГПР	10 000	0	16,7	71,8	598,1
7	ГПР	10 000	0	17,4	74,6	575,6
8	ГПР	10 000	0	16,9	72,8	591,0
9	ГПР	10 000	0	17,3	74,0	579,8
10	ГПР	10 000	0	17,1	74,0	582,5
Середнє	ГПР	10 000 ± 0	0 ± 0	17,1 ± 0,3	73,5 ± 1,1	584,8 ± 8,4
1	МП	9 980	20	15,3	64,7	652,4
2	МП	9 975	25	15,7	66,5	635,1
3	МП	9 985	15	15,5	65,4	644,8
4	МП	9 980	20	16,0	67,0	625,0
5	МП	9 970	30	15,6	65,9	639,7
6	МП	9 990	10	15,2	64,3	657,9
7	МП	9 980	20	15,9	66,7	628,9
8	МП	9 975	25	15,4	65,1	648,5
9	МП	9 985	15	15,8	66,2	631,6
10	МП	9 980	20	15,6	67,2	633,1
Середнє	МП	9 980 ± 6	20 ± 6	15,6 ± 0,3	65,9 ± 1,0	639,7 ± 10,8

Таблиця 4.10 – Статистична оцінка часових показників для набору з 10 % взаємоблокувань

Режим	Середній час, мс ($\bar{x} \pm s$)	CV часу, %	95 % ДІ часу, мс	Пропускна здатність ($\bar{x} \pm s$)	CV пропускної здатності, %	95 % ДІ пропускної здатності
ІГН	32,4 ± 0,45	1,38	32,08–32,72	308,6 ± 4,46	1,44	305,4–311,8
ГПР	17,1 ± 0,26	1,51	16,92–17,28	584,8 ± 8,42	1,44	578,8–590,8
МП	15,6 ± 0,26	1,66	15,42–15,78	639,7 ± 10,85	1,70	631,9–647,5

Для набору з 10 % взаємоблокувань усереднений час обробки при прогнозуванні становив 15,6 мс, що на 8,8 % менше, ніж для ГПР, і на 51,9 % менше, ніж при ігноруванні. Усереднена пропускна здатність прогнозного режиму становила

639,7 процесу/с і перевищувала ГПР на 9,4 %, а режим ігнорування – на 107,3 %. Коефіцієнт варіації основних часових показників не перевищував 1,93 %.

Середній час аналізу стану для ГПР становив 39,6 ± 0,69 мкс, тоді як для методу прогнозування – 25,0 ± 0,48 мкс. Отже, обчислювальні витрати одного виклику прогнозування були на 36,9 % меншими.

Для оцінювання достовірності у кожному повторі результат прогнозу зіставлявся з фактичним станом, підтвердженим ресурсними залежностями PostgreSQL. Узагальнення виконано за середніми значеннями десяти повторів; разом із середнім наведено стандартне відхилення та 95-відсотковий довірчий інтервал (табл. 4.11).

Таблиця 4.11 – Усереднені показники достовірності прогнозування за десятьма повторами

Показник	5 %: $\bar{x} \pm s$	95 % ДІ	10 %: $\bar{x} \pm s$	95 % ДІ
TP	490,0 ± 4,1	487,1–492,9	980,0 ± 4,1	977,1–982,9
FN	10,0 ± 4,1	7,1–12,9	20,0 ± 4,1	17,1–22,9
FP	8,0 ± 0,8	7,4–8,6	15,0 ± 0,8	14,4–15,6
TN	9 492,0 ± 0,8	9 491,4–9 492,6	8 985,0 ± 0,8	8 984,4–8 985,6
Accuracy, %	99,82 ± 0,03	99,80–99,84	99,65 ± 0,03	99,63–99,67
Recall, %	98,00 ± 0,82	97,42–98,58	98,00 ± 0,41	97,71–98,29
Specificity, %	99,92 ± 0,01	99,91–99,92	99,83 ± 0,01	99,83–99,84
Precision, %	98,39 ± 0,15	98,29–98,50	98,49 ± 0,07	98,44–98,55
F1-score, %	98,19 ± 0,34	97,95–98,44	98,25 ± 0,17	98,13–98,37
Частка пропусків, %	2,00 ± 0,82	1,42–2,58	2,00 ± 0,41	1,71–2,29
Частка хибних спрацювань, %	0,08 ± 0,01	0,08–0,09	0,17 ± 0,01	0,16–0,17
Випередження прогнозу, мс	3,10 ± 0,08	3,04–3,16	3,60 ± 0,08	3,54–3,66

Для набору з 5 % взаємоблокувань середні значення Accuracy, Recall, Specificity, Precision і F1-score становили відповідно 99,82 %, 98,00 %, 99,92 %, 98,39 % та 98,19 %. Для набору з 10 % відповідні значення дорівнювали 99,65 %, 98,00 %, 99,83 %, 98,49 % та 98,25 %. Частка пропущених небезпечних процесів становила 2,00 % в обох наборах, тоді як частка хибних спрацювань дорівнювала лише 0,08 % для набору з 5 % і 0,17 % для набору з 10 %. У другому наборі на 196,0 ± 0,82 обґрунтованого перезапуску припадало 15,0 ± 0,82 хибного, тобто хибні втручання траплялися у 13,10 раза рідше.

Прогноз формувався в середньому на 3,1 ± 0,08 мс раніше для набору з 5 % і на 3,6 ± 0,08 мс раніше для набору з 10 %. Цей інтервал використовувався для превентивного перезапуску процесу та звільнення ресурсу до переходу системи у фактичний стан взаємоблокування.

Порівняння трьох експериментів виконано за усередненими значеннями десяти повторів. Ігнорування не створювало додаткових витрат на аналіз, але призводило в середньому до завершення за таймаутом 500 процесів для набору з 5 % і 1 000 процесів для набору з 10 %. ГПР забезпечував завершення всіх процесів, однак реагував лише після формування циклу. Запропонований метод забезпечував у середньому завершення 9 990 і 9 980 процесів відповідно та формував рішення до фактичного утворення більшості циклів.

За двома конфігураціями навантаження прогнозування скорочувало середній час обробки відносно ГПР на 8,6–8,8 % і підвищувало пропускну здатність на 9,4 %. Порівняно з ігноруванням скорочення середнього часу становило 50,0–51,9 %, а приріст пропускної здатності – 100,0–107,3 %. Усереднена Accuracy перебувала в межах 99,65–99,82 %, Precision – 98,39–98,49 %, а F1-score – 98,19–98,25 %. Кількість хибних перезапусків була у 12,36–13,10 раза меншою за кількість обґрунтованих. Низькі коефіцієнти варіації та вузькі довірчі інтервали підтверджують стабільність відносної переваги методу в межах повторних експериментів.

Отримані усереднені результати показують, що використання прогнозування дає змогу поєднати високу достовірність із меншою затримкою реагування. Водночас відсутність резервного механізму означає, що пропущені прогнозом взаємоблокування безпосередньо зменшують частку успішно завершених процесів.

Для детального аналізу механізму формування взаємоблокування розглянемо зміну ресурсного стану окремої пари серверних процесів під час виконання транзакцій. У межах експерименту кожен процес послідовно отримував монопольний доступ до двох різних записів бази даних. За паралельного виконання різний порядок захоплення однакових ресурсів створював умови, за яких один процес утримував перший ресурс і очікував другий, зайнятий іншим процесом. Така послідовність

забезпечувала кероване формування циклічних ресурсних залежностей.

Для компактного подання позначимо два процеси як P1 і P2, а два записи бази даних, до яких вони отримують монопольний доступ, – як R1 і R2. Значення 1 у таблицях означає, що відповідний ресурс утримується процесом, значення 0 – що ресурс процесу не належить. Окремо вказується ресурс, на отримання якого сформовано поточний запит. Такий спосіб подання дає змогу одночасно відобразити фактичний розподіл ресурсів і напрямок подальшої ресурсної взаємодії процесів.

Після надання процесу P1 ресурсу R1, а процесу P2 ресурсу R2 обидва ресурси стають зайнятими. Наступний запит P1 на отримання R2 не може бути виконаний, оскільки R2 утримується процесом P2. У цей момент P1 переходить до очікування, а для запропонованого методу виникає подія, за якою актуалізується сигнатура процесу та виконується прогнозування. Розподіл ресурсів у цей момент наведено в таблиці 4.12.

Таблиця 4.12 – Розподіл ресурсів перед формуванням циклічної залежності

Процес	R1	R2	Запитуваний ресурс	Стан процесу
P1	1	0	R2	очікування R2
P2	0	1	–	виконання
Доступно	0	0	–	–

Стан, наведений у таблиці 4.12, ще не містить замкненого циклу: P1 очікує R2, однак P2 у цей момент не очікує ресурс, утримуваний P1. Для графа розподілу ресурсів цього недостатньо для фіксації фактичного взаємоблокування. Якщо подальше виконання продовжується без превентивного втручання і P2 формує запит на R1, ресурсна залежність замикається. Розподіл ресурсів у момент формування такого циклу наведено в таблиці 4.13.

Таблиця 4.13 – Розподіл ресурсів у момент формування циклічної залежності

Процес	R1	R2	Запитуваний ресурс	Стан процесу
P1	1	0	R2	очікування R2
P2	0	1	R1	очікування R1
Доступно	0	0	–	–

У таблиці 4.13 процес P1 утримує R1 та очікує R2, тоді як P2 утримує R2 та очікує R1. Отже, формується замкнена ресурсна залежність P1 – R2 – P2 – R1 – P1. Саме після формування такого стану ГПР може зафіксувати цикл і виконати відкат одного з процесів. Запропонований МП відрізняється тим, що обробка починається раніше – після факту ненадання запитуваного ресурсу, коли стан ще відповідає таблиці 4.12. Для правильно розпізнаного небезпечного епізоду це дає можливість сформувати превентивну керуючу дію до переходу ресурсної взаємодії у стан, наведений у таблиці 4.13.

Послідовність зміни ресурсного стану та відмінність моментів реагування ГПР і МП відобразимо на часовій діаграмі (рисунок 4.7). Для подання порядку подій введемо послідовні моменти t0–t5. У момент t0 процеси отримують перші ресурси, у t1 процес P1 формує запит на R2, а у t2 фіксується факт ненадання ресурсу. Для ГПР подальше реагування можливе лише після формування циклу ресурсних залежностей. Для МП подія ненадання ресурсу є підставою для актуалізації сигнатури, оцінювання ризику, формування множини ризикових процесів та вибору процесу для керуючої дії.

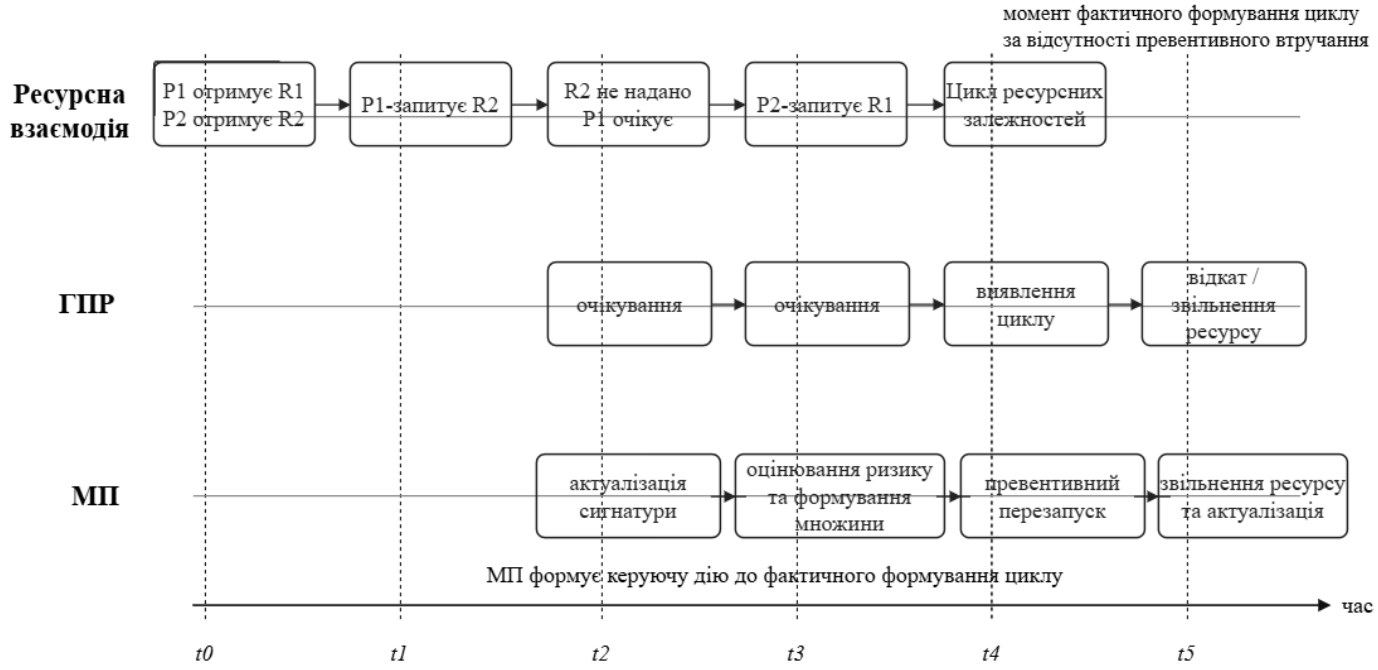


Рисунок 4.7 – Часова діаграма зміни ресурсного стану та реагування ГПР і МП під час експерименту

Як видно з рисунка 4.7, до моменту ненадання R2 обидва методи спостерігають однакову послідовність ресурсних подій. Після переходу P1 до очікування їхня логіка відрізняється. ГПР очікує появи замкненого циклу, тобто фактичного взаємоблокування, після чого виконується його виявлення та відкат. МП використовує проміжний ресурсний стан: після ненадання R2 актуалізується сигнатура P1 і виконується оцінювання ризику. Якщо процес включено до множини ризикових, керуюча дія виконується до формування циклу, а звільнення утримуваного ресурсу змінює подальшу послідовність ресурсної взаємодії.

Отримані часові результати повторних експериментів підтверджують таку послідовність. Для набору з 5 % процесів, що формують взаємоблокування, прогноз формувався в середньому на 3,1 ± 0,08 мс раніше за фактичне виникнення взаємоблокування, а для набору з 10 % – на 3,6 ± 0,08 мс. Цей інтервал використовувався для перезапуску обраного процесу та звільнення конфліктного ресурсу до формування фактичного циклу.

Отже, таблиці 4.12, 4.13 та рисунок 4.7 показують, що ГПР фіксує вже сформований цикл, а МП виконує аналіз після ненадання ресурсу та може сформувати керуючу дію до фактичного взаємоблокування.

Часові результати параграфів 4.2 і 4.3 мають різну фізичну природу: в імітаційній моделі середній час перебування процесу в системі визначався в тактах модельного часу, а в експериментальній системі середній час обробки вимірювався в мілісекундах реального виконання. Під час імітаційного моделювання скорочення середнього часу перебування процесу в системі запропонованого методу відносно ГПР становило 6,6–11,8 %, тоді як за усередненими даними повторних експериментів на реальній платформі – 8,6–8,8 %. Отже, встановлена закономірність зберігається.

Експериментальна перевірка на фізичній серверній платформі із застосуванням реальної операційної системи та СКБД підтверджує працездатність засобу за контрольовано сформованих сценаріїв ресурсної конкуренції. Водночас отримані значення достовірності й часових характеристик не є основою для безпосереднього

поширення їх результатів на довготривалі неконтрольовані виробничі навантаження.

Для використання методу прогнозування взаємоблокувань операційна система повинна забезпечувати можливість отримання інформації про поточний стан процесів та їх взаємодію з ресурсами. Необхідним є доступ до ідентифікаторів процесів і ресурсів, відомостей про ресурси, які вже утримуються процесом, а також про активні запити на отримання додаткових ресурсів. Критичною вимогою є можливість фіксації події відмови або затримки у виділенні ресурсу, оскільки саме така подія ініціює виконання процедури прогнозування. Операційна система також повинна надавати засобу прогнозування достатні привілеї для спостереження за станами процесів без порушення механізмів їх виконання та синхронізації.

Метод може застосовуватися в операційних системах, у яких використовується конкурентний доступ процесів або потоків до обмежених ресурсів і існує можливість однозначно визначити факт володіння ресурсом та запит на його отримання. При цьому не висувається вимога щодо конкретної архітектури або типу ОС, однак необхідна можливість інтеграції засобу прогнозування з механізмами керування процесами та ресурсами.

4.4 Висновки до четвертого розділу

Розроблено архітектуру програмного засобу прогнозування стану взаємоблокування процесів, що забезпечує отримання даних від операційної системи, побудову сигнатур процесів, використання бази нечітких правил, прогнозне оцінювання небезпечних станів і передавання керуючих дій до середовища виконання. Відокремлення рівнів доступу до системної інформації, аналітичного опрацювання та превентивного впливу забезпечує розділення процедур навчання й оперативного прогнозування.

Результати імітаційного моделювання для різної кількості ресурсів і частки процесів, що потрапляють до стану взаємоблокування, підтвердили працездатність запропонованого методу. У досліджених конфігураціях забезпечено завершення

99,85–99,95 % процесів, Accuracy становила 99,65–99,88 %, Recall – 99,00 %, Precision – 98,61–98,67 %, а F1-score – 98,80–98,84 %. Частка пропущених небезпечних процесів не перевищувала 1,00 %, а хибних спрацювань – 0,24 %.

Експериментальна перевірка на реальній комп’ютерній системі підтвердила високу достовірність і стабільність прогнозування. За десятьма повторами Accuracy перебувала в межах 99,65–99,82 %, Precision – 98,39–98,49 %, F1-score – 98,19–98,25 %, а прогноз формувався в середньому на 3,1–3,6 мс до фактичного виникнення взаємоблокування. Порівняно з графом розподілу ресурсів середній час обробки скоротився на 8,6–8,8 %, а пропускна здатність зросла на 9,4 %, що підтверджує доцільність використання розробленого методу для завчасного формування превентивних керуючих дій.

Основні результати розділу опубліковані у працях [116; 119].

ВИСНОВКИ

У дисертаційній роботі розв’язано актуальну науково-прикладну задачу підвищення достовірності та забезпечення своєчасності прогнозування стану взаємоблокування процесів у комп’ютерних системах шляхом розроблення та удосконалення моделей, методів і засобів аналізу сигнатур процесів, автоматизованого формування бази нечітких правил та завчасного оцінювання ризику переходу процесів у стан взаємоблокування. Основні наукові та практичні результати роботи полягають у такому.

1. Проведено системний аналіз причин та умов виникнення взаємоблокувань процесів, моделей подання станів процесів і ресурсних залежностей, а також відомих методів і алгоритмів вирішення задачі взаємоблокування в комп’ютерних системах. Встановлено, що методи ігнорування не усувають проблему на рівні операційної системи, методи запобігання та уникнення накладають обмеження на архітектуру апаратних та програмних засобів, на розподіл ресурсів і потребують попередньої інформації про ресурсні потреби процесів, а методи виявлення та усунення мають переважно реактивний характер. Визначено недостатнє врахування відомими підходами сукупності ідентифікаційних, ресурсних, часових і поведінкових характеристик процесів. Обґрунтовано доцільність переходу від виявлення сформованого взаємоблокування до завчасного прогнозування небезпечних ресурсних станів із використанням сигнатур процесів, нечіткої кластеризації та нечіткого логічного висновку.

2. Удосконалено архітектуру засобів прогнозування стану взаємоблокування процесів, яка поєднує модулі отримання даних про процеси та ресурси, побудови сигнатур, формування стану комп’ютерної системи, навчання, нечіткої кластеризації, побудови бази правил, прогнозування, оцінювання ризику та формування керуючого рішення. Для її функціонування розроблено моделі сигнатури процесу, стану комп’ютерної системи, процесу побудови бази нечітких правил та прогнозування стану взаємоблокування. Відокремлення процедур

навчання від оперативного прогнозування дало змогу винести ресурсомісткі процедури кластеризації та формування бази правил за межі оперативного циклу розподілу ресурсів і використовувати в робочому режимі попередньо сформовану базу правил.

3. Набув подальшого розвитку метод побудови сигнатури процесу, який забезпечує формалізоване перетворення різнорідних даних операційної системи про процес та його ресурсну взаємодію у впорядкований вектор ознак сталої структури. У сигнатурі враховано ідентифікаційні, ресурсні, часові та поведінкові характеристики процесу, зокрема відомості про утримувані та запитувані ресурси. Розмежовано інформаційну частину сигнатури та навчальний маркер, завдяки чому однакове подання даних використовується під час навчання й оперативного прогнозування, а формування навчальних ознак винесено за межі оперативного циклу. Це забезпечило отримання узгодженого інформативного опису ресурсної поведінки процесів, придатного для подальшої кластеризації, нечіткого аналізу та прогнозування.

4. Удосконалено метод побудови бази нечітких правил для прогнозування стану взаємоблокування процесів. Метод передбачає формування навчальної вибірки з накопичених сигнатур, виділення ресурсної складової, нормалізацію ознак, нечітку кластеризацію методом Fuzzy C-Means, визначення рівня ризику сформованих кластерів та автоматизоване генерування правил типу «якщо–то». Сформована база правил узагальнює типові комбінації утримуваних і запитуваних ресурсів та їх зв’язок із подальшим переходом процесів до небезпечних станів. Це зменшило залежність процедури формування правил від експертного налаштування та створило формалізовану інформаційну основу для оперативного оцінювання поточних сигнатур процесів.

5. Розроблено метод прогнозування стану взаємоблокування процесів, який використовує поточні сигнатури процесів і сформовану базу нечітких правил для завчасного оцінювання небезпечних змін ресурсної взаємодії. Для процесів, що запитують та не отримують ресурс, виконуються фазифікація ознак, активізація

нечітких правил, агрегування результатів і формування оцінки належності до небезпечних станів. Для нормального та потенційно ризикового станів результат прогнозування реєструється без керуючого впливу. Для передблокувального та критичного станів формується множина кандидатів, із якої вилучаються системні процеси, після чого для несистемного процесу з найменшим часом перебування в системі формується рішення про примусовий перезапуск. Це забезпечує можливість розриву потенційної послідовності взаємного очікування ресурсів до фактичного виникнення взаємоблокування з обмеженням втрат результатів виконання процесів.

6. Розроблено програмні засоби та проведено експериментальну перевірку запропонованого методу прогнозування. Програмна реалізація забезпечує повний цикл від отримання подій операційної системи та формування сигнатур до нечіткого оцінювання ризику й формування превентивного керуючого рішення. За результатами імітаційного моделювання забезпечено завершення 99,85–99,95 % процесів, Accuracy становила 99,65–99,88 %, Recall – 99,00 %, Precision – 98,61–98,67 %, F1-score – 98,80–98,84 %, а частка пропущених небезпечних процесів не перевищувала 1,00 %. Експериментальна перевірка на реальній комп’ютерній системі підтвердила стабільність результатів: Accuracy становила 99,65–99,82 %, Precision – 98,39–98,49 %, F1-score – 98,19–98,25 %, при цьому прогноз формувався в середньому за 3,1–3,6 мс до фактичного виникнення взаємоблокування. Порівняно з методом на основі графа розподілу ресурсів середній час обробки скорочено на 8,6–8,8 %, а пропускну здатність підвищено на 9,4 %, що підтвердило достовірність, своєчасність та практичну придатність розробленого методу і засобів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pester A., Sulema Y., Dychka I., Sulema O. Temporal multimodal data-processing algorithms based on Algebraic System of Aggregates. *Algorithms*. 2023. Vol. 16, No. 4. Article 186. DOI: https://doi.org/10.3390/a16040186.

2. Dychka I., Oleshchenko L., Vovchenko D., Hu Z. Intelligent load balancing framework for distributed big data processing system. *International Journal of Wireless and Microwave Technologies*. 2026. Vol. 16, No. 3. P. 72–87. DOI: https://doi.org/10.5815/ijwmt.2026.03.05.

3. Mukhin V., Hu Z., Kornaga Y., Lavrenko Y., Herasymenko O. Distributed computer system resources control mechanism based on network-centric approach. *International Journal of Intelligent Systems and Applications*. 2017. Vol. 9, No. 7. P. 41–51. DOI: https://doi.org/10.5815/ijisa.2017.07.05.

4. Mukhin V., Steshyn V., Kornaga Y., Herasymenko O., Bazaka Y. Study of the functioning of the distributed computer system with a resource control mechanism based on a network-centric approach. *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*. Kyiv, 2019. P. 122–126. URL: https://ieeexplore.ieee.org/document/8924331/

5. Mukhin V., Kornaga Y., Bazaka Y., Barabash A., Mukhin O. Improved method of testing distributed system software. *TASK Quarterly*. 2021. Vol. 25, No. 4. P. 459–474. URL: https://journal.mostwiedzy.pl/TASKQuarterly/article/view/1755

6. Mukhin V., Volokyta A., Heriatovych Y., Rehida P. Method for efficiency increasing of distributed computer systems based on the proactive parallel computing approach. *Advances in Electrical and Computer Engineering*. 2018. Vol. 18, No. 2. P. 83–90. URL: https://www.aece.ro/abstractplus.php?article=15&number=2&year=2018

7. Kovalenko A., Miroshnychenko R., Martyntsov A. Distributed computing systems based on the use of Grid technologies. *Системи управління, навігації та зв'язку*. 2023. Вип. 1. С. 85–90. DOI: https://doi.org/10.26906/SUNZ.2023.1.101

8. Giebas D., Wojszczyk R. Deadlocks Detection in Multithreaded Applications Based on Source Code Analysis. *Applied Sciences*. 2020. Vol. 10(2). Art. 532. DOI: https://doi.org/10.3390/app10020532.

9. Wang Y., Kelly T., Kudlur M., Lafortune S., Mahlke S. Gadara: Dynamic Deadlock Avoidance for Multithreaded Programs. *OSDI*, 2008. P. 281–294. URL: https://www.usenix.org/conference/osdi-08/gadara-dynamic-deadlock-avoidance-multithreaded-programs

10. Ярошевич Р. О., Коваленко А. А., Ляшенко О. С. Порівняльний аналіз організації хмарної інфраструктури. *Сучасні інформаційні системи*. 2021. Т. 5, № 2. С. 108–113. DOI: https://doi.org/10.20998/2522-9052.2021.2.15

11. Melnyk A., Al Khatib Ahmad. Orthogonal transform processor cores based on discrete cosine transform. *Contemporary Computing in Ukraine*. New York: ACM, 2000. P. 45–52. DOI: https://doi.org/10.1145/352491.352498.

12. Мельник А. О. Архітектура комп'ютера. Луцьк: Волинська обласна друкарня, 2008. 470 с. URL: https://librarygo.lpnu.ua/?elbook=arhitektura-kompyutera

13. Habermann A. N. Prevention of system deadlocks. *Communications of the ACM*. 1969. Vol. 12, No. 7. P. 373–377, 385. DOI: https://doi.org/10.1145/363156.363160.

14. Obermarck R. Distributed deadlock detection algorithm. *ACM Transactions on Database Systems*. 1982. Vol. 7, No. 2. P. 187–208. DOI: https://doi.org/10.1145/319702.319717.

15. Coffman E. G., Elphick M. J., Shoshani A. System deadlocks. *ACM Computing Surveys*. 1971. Vol. 3, No. 2. P. 67–78. DOI: https://doi.org/10.1145/356586.356588.

16. Holt R. C. Some deadlock properties of computer systems. *ACM Computing Surveys*. 1972. Vol. 4, No. 3. P. 179–196. DOI: https://doi.org/10.1145/356603.356607.

17. Knapp E. Deadlock detection in distributed databases. *ACM Computing Surveys*. 1987. Vol. 19, No. 4. P. 303–328. DOI: https://doi.org/10.1145/45075.46163.

18. Chandy K. M., Misra J., Haas L. M. Distributed deadlock detection. *ACM Transactions on Computer Systems*. 1983. Vol. 1, No. 2. P. 144–156. DOI: https://doi.org/10.1145/357360.357365.

19. Peterson G. L. Myths about the mutual exclusion problem. *Information Processing Letters*. 1981. Vol. 12, No. 3. P. 115–116. DOI: https://doi.org/10.1016/0020-0190(81)90106-X.

20. Lamport L. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*. 1978. Vol. 21, No. 7. P. 558–565. DOI: https://doi.org/10.1145/359545.359563.

21. Lynch N. A. Distributed Algorithms. *San Francisco: Morgan Kaufmann*, 1996. 872 p. URL: https://shop.elsevier.com/books/distributed-algorithms/lynch/978-1-55860-348-6

22. Tanenbaum A. S., Bos H. Modern Operating Systems. *4th ed. Boston: Pearson*, 2015. 1136 p. URL: https://research.vu.nl/en/publications/modern-operating-systems-4th-edition/

23. Silberschatz A., Galvin P. B., Gagne G. Operating System Concepts. *10th ed. Hoboken: Wiley*, 2018. 976 p. URL: https://www.wiley.com/en-ca/operating-system-concepts-10th-edition-p-9781119320913R150

24. Stallings W. Operating Systems: Internals and Design Principles. *9th ed. Pearson*, 2018. 800 p. URL: https://www.pearson.com/en-gb/subject-catalog/p/operating-systems-internals-and-design-principles-global-edition/P200000005477/9781292214306

25. Tanenbaum A. S., Van Steen M. Distributed Systems. *3rd ed. CreateSpace Independent Publishing Platform*, 2017. 705 p. URL: https://www.distributed-systems.net/index.php/books/ds3/

26. Sikora L., Lysa N., Fedevych O., Khyliak N. Information and laser technologies for assessing the level of risks from harmful emissions from man-made objects. *Computer Systems and Information Technologies*. 2025. No. 1. P. 6–15. DOI: https://doi.org/10.31891/csit-2025-1-1.

27. Kryzhanyvskyi D., Drozd A., Besedovskyi O. Decision-making method in interdependent computing systems. *Computer Systems and Information Technologies*. 2025. No. 1. P. 54–65. DOI: https://doi.org/10.31891/csit-2025-1-7.

28. Pikh I., Bilyk O. Workload balancing in the test case scheduling: a mathematical approach. *Computer Systems and Information Technologies*. 2025. No. 2. P. 81–86. DOI: https://doi.org/10.31891/csit-2025-2-9.

29. Hassan M. H., Darwish S. M., Elkaffas S. M. An efficient deadlock handling model based on neutrosophic logic: case study on real time healthcare database systems. *IEEE Access*. 2022. Vol. 10. P. 76607–76621. DOI: https://doi.org/10.1109/ACCESS.2022.3192414.

30. Talaat M., Saleh A., Moawad M., Zaki J. Fog computing effective load balancing and strategy for deadlock prediction management. *Ain Shams Engineering Journal*. 2023. Vol. 14, No. 12. Article 102561. DOI: https://doi.org/10.1016/j.asej.2023.102561.

31. Lu Y., Chen Y., Li Z., Wu N. An efficient method of deadlock detection and recovery for flexible manufacturing systems by resource flow graphs. *IEEE Transactions on Automation Science and Engineering*. 2022. Vol. 19, No. 3. P. 1707–1718. DOI: https://doi.org/10.1109/TASE.2021.3114498.

32. Li Z., Bian J., Li W., Zhang L. Safety measures in automotive operating system: a comprehensive review of trends and defense frameworks. *The Journal of Supercomputing*. 2025. Vol. 81. Article 1205. DOI: https://doi.org/10.1007/s11227-025-07635-7.

33. Aldossary D., Aldahasi E., Balharith T., Helmy T. A systematic literature review on load-balancing techniques in fog computing: architectures, strategies, and emerging trends. *Computers*. 2025. Vol. 14, No. 6. Article 217. DOI: https://doi.org/10.3390/computers14060217.

34. Kashani M. H., Mahdipour E. Load balancing algorithms in fog computing. *IEEE Transactions on Services Computing*. 2022. Vol. 16, No. 2. P. 1505–1521. DOI: https://doi.org/10.1109/TSC.2022.3174475.

35. Sadashiv N. Load balancing in fog computing: a detailed survey. *International Journal of Computing and Digital Systems*. 2023. Vol. 13. P. 729–750. DOI: https://doi.org/10.12785/ijcds/130158

36. Shakeel H., Alam M. Load balancing approaches in cloud and fog computing environments: a framework, classification, and systematic review. *International Journal of Cloud Applications and Computing*. 2022. Vol. 12, No. 1. P. 1–24. DOI: https://doi.org/10.4018/IJCAC.311503

37. Ebneyousef S., Shirmarz A. A taxonomy of load balancing algorithms and approaches in fog computing: a survey. *Cluster Computing*. 2023. Vol. 26. P. 3187–3208. DOI: https://doi.org/10.1007/s10586-023-03982-3

38. Singh J., Singh P., Amhoud E. M., Hedabou M. Energy-efficient and secure load balancing technique for SDN-enabled fog computing. *Sustainability*. 2022. Vol. 14. Article 12951. DOI: https://doi.org/10.3390/su141912951

39. Javanmardi S., Shojafar M., Mohammadi R., Persico V., Pescapè A. S-FoS: a secure workflow scheduling approach for performance optimization in SDN-based IoT-Fog networks. *Journal of Information Security and Applications*. 2023. Vol. 72. Article 103404. DOI: https://doi.org/10.1016/j.jisa.2022.103404

40. Liu W., Li C., Zheng A., Zheng Z., Zhang Z., Xiao Y. Fog computing resource-scheduling strategy in IoT based on artificial bee colony algorithm. *Electronics*. 2023. Vol. 12. Article 1511. DOI: https://doi.org/10.3390/electronics12071511

41. Chaudhry R., Rishiwal V. An efficient task allocation with fuzzy reptile search algorithm for disaster management in urban and rural area. *Sustainable Computing: Informatics and Systems*. 2023. Vol. 39. Article 100893. DOI: https://doi.org/10.1016/j.suscom.2023.100893

42. Cai Y., Chan W. K. MagicLock: Scalable Detection of Potential Deadlocks in Large-Scale Multithreaded Programs. *IEEE Transactions on Software Engineering*. 2014. Vol. 40(3). P. 266–281. DOI: https://doi.org/10.1109/TSE.2014.2301725

43. Kazemi S. M., Ghanbari S., Kazemi M., Othman M. Optimum scheduling in fog computing using the Divisible Load Theory (DLT) with linear and nonlinear loads.

Computer Networks. 2023. Vol. 220. Article 109483. DOI: https://doi.org/10.1016/j.comnet.2022.109483

44. Cai Y., Wu S., Chan W. K. ConLock: A Constraint-Based Approach to Dynamic Checking on Deadlocks in Multithreaded Programs. *ICSE*, 2014. P. 491–502. DOI: https://doi.org/10.1145/2568225.2568312.

45. Lin Y., Kulkarni S. S. Automatic Repair for Multi-Threaded Programs with Deadlock/Livelock Using Maximum Satisfiability. *ISSTA*, 2014. P. 237–247. DOI: https://doi.org/10.1145/2610384.2610398.

46. Kavi K., Moshtaghi A., Chen D. J. Modeling Multithreaded Applications Using Petri Nets. *International Journal of Parallel Programming*. 2002. Vol. 30(5). P. 353–371. DOI: https://doi.org/10.1023/A:1019917329895.

47. Ameena B., Ramasamy L. Drawer cosine optimization enabled task offloading in fog computing. *Expert Systems with Applications*. 2025. Vol. 259. Article 125212. DOI: https://doi.org/10.1016/j.eswa.2024.125212

48. Singh S. P. Effective load balancing strategy using fuzzy golden eagle optimization in fog computing environment. *Sustainable Computing: Informatics and Systems*. 2022. Vol. 35. Article 100766. DOI: https://doi.org/10.1016/j.suscom.2022.100766

49. Liao H., Wang Y., Stanley J., et al. Eliminating Concurrency Bugs in Multithreaded Software: A New Approach Based on Discrete-Event Control. *IEEE Transactions on Control Systems Technology*. 2013. Vol. 21(6). P. 2067–2082. DOI: https://doi.org/10.1109/TCST.2012.2226034.

50. Liu Z., Dai P., Xing H., Yu Z., Zhang W. A distributed algorithm for task offloading in vehicular networks with hybrid fog/cloud computing. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. 2022. Vol. 52. P. 4388–4401.

51. Bensalem S., Havelund K. Dynamic Deadlock Analysis of Multi-threaded Programs. *LNCS 3875*. 2006. P. 208–223. DOI: https://doi.org/10.1007/11678779_15.

52. Gordon C. S., Ernst M. D., Grossman D. Static Lock Capabilities for Deadlock Freedom. *TLDI*, 2012. P. 67–78. DOI: https://doi.org/10.1145/2103786.2103796.

53. Gerakios P., Papaspyrou N., Sagonas K. A Type System for Unstructured Locking that Guarantees Deadlock Freedom without Imposing a Lock Ordering. *EPTCS*. 2011. Vol. 69. P. 44–58. DOI: https://doi.org/10.4204/EPTCS.69.4.

54. Lu F., Tao R., Du Y., Zeng Q., Bao Y. Deadlock Detection-Oriented Unfolding of Unbounded Petri Nets. *Information Sciences*. 2019. Vol. 497. P. 1–22. DOI: https://doi.org/10.1016/j.ins.2019.05.021.

55. Wu B., Lv X., Shamsi W. D., Gholami Dizicheh E. Optimal deploying IoT services on the fog computing: a metaheuristic-based multi-objective approach. *Journal of King Saud University - Computer and Information Sciences*. 2022. Vol. 34. P. 10010–10027. DOI: https://doi.org/10.1016/j.jksuci.2022.10.002

56. Cheng Y., Vijayaraj A., Sree Pokkuluri K., Salehnia T., Montazerolghaem A., Rateb R. Vehicular fog resource allocation approach for VANETs based on deep adaptive reinforcement learning combined with heuristic information. *IEEE Access*. 2024. Vol. 12. P. 139056–139075. DOI: https://doi.org/10.1109/ACCESS.2024.3455168

57. Wu W., Xing Z., Yue H., Su H., Pang S. Petri-Net-Based Deadlock Detection and Recovery for Control of Interacting Equipment in Automated Container Terminals. *IET Intelligent Transport Systems*. 2022. Vol. 16(6). P. 739–753. DOI: https://doi.org/10.1049/itr2.12168.

58. Aldağ M., Kırsal Y., Ülker S. An analytical modelling and QoS evaluation of fault-tolerant load balancer and web servers in fog computing. *The Journal of Supercomputing*. 2022. Vol. 78. P. 12136–12158. DOI: https://doi.org/10.1007/s11227-022-04345-2

59. Bukhari A. A., Hussain F. K. Fuzzy logic trust-based fog node selection. *Internet of Things*. 2024. Vol. 27. Article 101293. DOI: https://doi.org/10.1016/j.iot.2024.101293

60. Lu F., Wang X., Yuan G., Zheng J., Bao Y. Deadlock Detection and Replay of Multi-Thread Programs Based on Refined Segmentation and Lock Graphs. *Tsinghua Science and Technology*. 2026. DOI: https://doi.org/10.26599/TST.2025.9010085.

61. Bianchi F. A., Margara A., Pezzè M. A Survey of Recent Trends in Testing Concurrent Software Systems. *IEEE Transactions on Software Engineering*. 2018. Vol. 44(8). P. 747–783. DOI: https://doi.org/10.1109/TSE.2017.2707089.

62. Premkumar N., Santhosh R. Pelican optimization algorithm with blockchain for secure load balancing in fog computing. *Multimedia Tools and Applications*. 2023. Vol. 83. P. 53417–53439.

63. Wang Y., Liao H., Reveliotis S., Kelly T., Mahlke S., Lafortune S. Gadara Nets: Modeling and Analyzing Lock Allocation for Deadlock Avoidance in Multithreaded Software. *IEEE CDC*, 2009. P. 4971–4976. DOI: https://doi.org/10.1109/CDC.2009.5399950.

64. Azizi S., Shojafar M., Farzin P., Dogani J. DCSP: a delay and cost-aware service placement and load distribution algorithm for IoT-based fog networks. *Computer Communications*. 2024. Vol. 215. P. 9–20. DOI: https://doi.org/10.1016/j.comcom.2023.12.016

65. Ebrahim M., Hafid A. Resilience and load balancing in fog networks: a multi-criteria decision analysis approach. *Microprocessors and Microsystems*. 2023. Vol. 101. Article 104893. DOI: https://doi.org/10.1016/j.micpro.2023.104893

66. Abdulazeez D. H., Askar S. K. A novel offloading mechanism leveraging fuzzy logic and deep reinforcement learning to improve IoT application performance in a three-layer architecture within the fog-cloud environment. *IEEE Access*. 2024. Vol. 12. P. 39936–39952. DOI: https://doi.org/10.1109/ACCESS.2024.3376670

67. Al Maruf M., Singh A., Azim A., Auluck N. Faster fog computing based over-the-air vehicular updates: a transfer learning approach. *IEEE Transactions on Services Computing*. 2022. Vol. 15. P. 3245–3259. DOI: https://doi.org/10.1109/TSC.2021.3099897

68. Savage S., Burrows M., Nelson G., Sobalvarro P., Anderson T. Eraser: A Dynamic Data Race Detector for Multithreaded Programs. *ACM Transactions on Computer Systems*. 1997. Vol. 15(4). P. 391–411. DOI: https://doi.org/10.1145/265924.265927

69. Khang T. D., Vuong N. D., Tran M.-K., Fowler M. Fuzzy C-Means Clustering Algorithm with Multiple Fuzzification Coefficients. *Algorithms*. 2020. Vol. 13(7). Art. 158. DOI: https://doi.org/10.3390/a13070158.

70. Kaur A., Auluck N. Scheduling algorithms for truly heterogeneous hierarchical fog networks. Software: *Practice and Experience*. 2022. Vol. 52. P. 2411–2438. DOI: https://doi.org/10.1002/spe.3132

71. Askari S. Fuzzy C-Means Clustering Algorithm for Data with Unequal Cluster Sizes and Contaminated with Noise and Outliers: Review and Development. *Expert Systems with Applications*. 2021. Vol. 165. Art. 113856. DOI: https://doi.org/10.1016/j.eswa.2020.113856.

72. Wang C., Pedrycz W., Li Z., Zhou M. Residual-Driven Fuzzy C-Means Clustering for Image Segmentation. *IEEE/CAA Journal of Automatica Sinica*. 2021. Vol. 8(4). P. 876–889. DOI: https://doi.org/10.1109/JAS.2020.1003420.

73. Ren C., Sun L. A Bi-directional Fuzzy C-Means Clustering Ensemble Algorithm Considering Local Information. *International Journal of Computational Intelligence Systems*. 2021. Vol. 14. Art. 171. DOI: https://doi.org/10.1007/s44196-021-00014-z

74. Rustam E. Modified Possibilistic Fuzzy C-Means Algorithm for Clustering Incomplete Data Sets. *Acta Polytechnica*. 2021. Vol. 61(2). P. 364–377. DOI: https://doi.org/10.14311/AP.2021.61.0364.

75. Sarker I. H. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*. 2021. Vol. 2. Art. 160. DOI: https://doi.org/10.1007/s42979-021-00592-x.

76. Lone K., Sofi S. A. e-TOALB: an efficient task offloading in IoT-fog networks. *Concurrency and Computation: Practice and Experience*. 2024. Vol. 36. Article e7951. DOI: https://doi.org/10.1002/cpe.7951

77. Jain A. K. Data Clustering: 50 Years Beyond K-Means. *Pattern Recognition Letters*. 2010. Vol. 31(8). P. 651–666. DOI: https://doi.org/10.1016/j.patrec.2009.09.011.

78. Xu D., Tian Y. A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*. 2015. Vol. 2. P. 165–193. DOI: https://doi.org/10.1007/s40745-015-0040-1.

79. Hayawi K., Anwar Z., Malik A. W., Trabelsi Z. Airborne computing: a toolkit for UAV-assisted federated computing for sustainable smart cities. *IEEE Internet of Things Journal*. 2023. Vol. 10. P. 18941–18950. DOI: https://doi.org/10.1109/JIOT.2023.3292308

80. Zhang T., Yue D., Yu L., Dou C., Xie X. Joint energy and workload scheduling for fog-assisted multimicrogrid systems: a deep reinforcement learning approach. *IEEE Systems Journal*. 2023. Vol. 17. P. 164–175. DOI: https://doi.org/10.1109/JSYST.2022.3171534

81. Keshri R., Vidyarthi D. P. An ML-based task clustering and placement using hybrid Jaya-gray wolf optimization in fog-cloud ecosystem. *Concurrency and Computation: Practice and Experience*. 2024. Vol. 36. Article e8109. DOI: https://doi.org/10.1002/cpe.8109

82. Ezugwu A. E., Ikotun A. M., Oyelade O. O., et al. A Comprehensive Survey of Clustering Algorithms: State-of-the-Art Machine Learning Applications, Taxonomy, Challenges, and Future Research Prospects. *Engineering Applications of Artificial Intelligence*. 2022. Vol. 110. Art. 104743. DOI: https://doi.org/10.1016/j.engappai.2022.104743.

83. Khan S., Shah I. A., Nadeem M. F., Jan S., Whangbo T., Ahmad S. Optimal resource allocation and task scheduling in fog computing for Internet of Medical Things applications. *Human-Centric Computing and Information Sciences*. 2023. Vol. 13. DOI: https://doi.org/10.22967/HCIS.2023.13.056

84. Arbelaitz O., Gurrutxaga I., Muguerza J., Pérez J. M., Perona I. An Extensive Comparative Study of Cluster Validity Indices. *Pattern Recognition*. 2013. Vol. 46(1). P. 243–256. DOI: https://doi.org/10.1016/j.patcog.2012.07.021.

85. Hüllermeier E. Fuzzy Methods in Machine Learning and Data Mining: Status and Prospects. *Fuzzy Sets and Systems*. 2005. Vol. 156(3). P. 387–406. DOI: https://doi.org/10.1016/j.fss.2005.05.036.

86. Yakubu I. Z., Murali M. An efficient meta-heuristic resource allocation with load balancing in IoT-Fog-cloud computing environment. *Journal of Ambient Intelligence and Humanized Computing*. 2023. Vol. 14. P. 2981–2992. DOI: https://doi.org/10.1007/s12652-023-04544-6

87. Mendel J. M. Fuzzy Logic Systems for Engineering: A Tutorial. *Proceedings of the IEEE*. 1995. Vol. 83(3). P. 345–377. DOI: https://doi.org/10.1109/5.364485.

88. Guillaume S. Designing Fuzzy Inference Systems from Data: An Interpretability-Oriented Review. *IEEE Transactions on Fuzzy Systems*. 2001. Vol. 9(3). P. 426–443. DOI: https://doi.org/10.1109/91.928739.

89. Gacto M. J., Alcalá R., Herrera F. Interpretability of Linguistic Fuzzy Rule-Based Systems: An Overview of Interpretability Measures. *Information Sciences*. 2011. Vol. 181(20). P. 4340–4360. DOI: https://doi.org/10.1016/j.ins.2011.02.021.

90. Xu R., Wunsch D. Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks*. 2005. Vol. 16(3). P. 645–678. DOI: https://doi.org/10.1109/TNN.2005.845141.

91. Halkidi M., Batistakis Y., Vazirgiannis M. On Clustering Validation Techniques. *Journal of Intelligent Information Systems*. 2001. Vol. 17(2–3). P. 107–145. DOI: https://doi.org/10.1023/A:1012801612483.

92. Potu N., Bhukya S., Jatoth C., Parvataneni P. Quality-aware energy efficient scheduling model for fog computing comprised IoT network. *Computers & Electrical Engineering*. 2022. Vol. 97. Article 107603. DOI: https://doi.org/10.1016/j.compeleceng.2021.107603

93. Sabireen H., Venkataraman N. A hybrid and light weight metaheuristic approach with clustering for multi-objective resource scheduling and application placement in fog environment. *Expert Systems with Applications*. 2023. Vol. 223. Article 119895. DOI: https://doi.org/10.1016/j.eswa.2023.119895

94. Ramezani Shahidani F., Ghasemi A., Toroghi Haghighat A., Keshavarzi A. Task scheduling in edge-fog-cloud architecture: a multi-objective load balancing approach using reinforcement learning algorithm. *Computing*. 2023. Vol. 105. P. 1337–1359. DOI: https://doi.org/10.1007/s00607-022-01147-5

95. Pakhira M. K., Bandyopadhyay S., Maulik U. Validity Index for Crisp and Fuzzy Clusters. *Pattern Recognition*. 2004. Vol. 37(3). P. 487–501. DOI: https://doi.org/10.1016/j.patcog.2003.06.005.

96. Chiu S. L. Fuzzy Model Identification Based on Cluster Estimation. *Journal of Intelligent & Fuzzy Systems*. 1994. Vol. 2(3). P. 267–278. DOI: https://doi.org/10.3233/IFS-1994-2306.

97. Ishibuchi H., Nozaki K., Yamamoto N., Tanaka H. Selecting Fuzzy If-Then Rules for Classification Problems Using Genetic Algorithms. *IEEE Transactions on Fuzzy Systems*. 1995. Vol. 3(3). P. 260–270. DOI: https://doi.org/10.1109/91.413232.

98. Deng W., Zhu L., Shen Y., Zhou C., Guo J., Cheng Y. A novel multi-objective optimized DAG task scheduling strategy for fog computing based on container migration mechanism. *Wireless Networks*. 2024. Vol. 31. P. 1005–1019. DOI: https://doi.org/10.1007/s11276-024-03811-4

99. Oprea S.-V., Bâra A. An edge-fog-cloud computing architecture for IoT and smart metering data. *Peer-to-Peer Networking and Applications*. 2023. Vol. 16. P. 818–845. DOI: https://doi.org/10.1007/s12083-022-01436-y

100. Ali H. S., Sridevi R. Mobility and security aware real-time task scheduling in fog-cloud computing for IoT devices: a fuzzy-logic approach. *The Computer Journal*. 2024. Vol. 67. P. 782–805. DOI: https://doi.org/10.1093/comjnl/bxad019

101. Bracha G., Toueg S. Distributed deadlock detection. *Distributed Computing*. 1987. Vol. 2. P. 127–138. DOI: https://doi.org/10.1007/BF01782773.

102. Singhal M. Deadlock detection in distributed systems. *Computer*. 1989. Vol. 22, No. 11. P. 37–48. DOI: https://doi.org/10.1109/2.43525.

103. Eslamimehr M., Palsberg J. Sherlock: scalable deadlock detection for concurrent programs. *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 2014. P. 353–365. DOI: https://doi.org/10.1145/2635868.2635918.

104. Kalhauge C. G., Palsberg J. Sound deadlock prediction. *Proceedings of the ACM on Programming Languages*. 2018. Vol. 2, OOPSLA. Article 146. P. 1–29. DOI: https://doi.org/10.1145/3276516.

105. Cai Y., Meng R., Palsberg J. Low-overhead deadlock prediction. *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 2020. P. 1298–1309. DOI: https://doi.org/10.1145/3377811.3380367.

106. Huang Y., Ogles B., Mercer E. A predictive analysis for detecting deadlock in MPI programs. *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 2020. P. 18–28. DOI: https://doi.org/10.1145/3324884.3416588.

107. Zhou J., Yang H., Lange J., Liu T. Deadlock prediction via generalized dependency. *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2022. P. 455–466. DOI: https://doi.org/10.1145/3533767.3534377.

108. Tunç H. C., Mathur U., Pavlogiannis A., Viswanathan M. Sound dynamic deadlock prediction in linear time. *Proceedings of the ACM on Programming Languages*. 2023. Vol. 7, PLDI. Article 177. P. 1733–1758. DOI: https://doi.org/10.1145/3591291.

109. Zadeh L. A. Fuzzy sets. *Information and Control*. 1965. Vol. 8, No. 3. P. 338–353. DOI: https://doi.org/10.1016/S0019-9958(65)90241-X.

110. Dunn J. C. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*. 1973. Vol. 3, No. 3. P. 32–57. DOI: https://doi.org/10.1080/01969727308546046.

111. Mamdani E. H., Assilian S. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*. 1975. Vol. 7, No. 1. P. 1–13. DOI: https://doi.org/10.1016/S0020-7373(75)80002-2.

112. Bezdek J. C. Pattern Recognition with Fuzzy Objective Function Algorithms. *New York: Springer*, 1981. 256 p. DOI: https://doi.org/10.1007/978-1-4757-0450-1.

113. Sugeno M., Yasukawa T. A Fuzzy-Logic-Based Approach to Qualitative Modeling. *IEEE Transactions on Fuzzy Systems*. 1993. Vol. 1(1). P. 7–31. DOI: https://doi.org/10.1109/TFUZZ.1993.390281.

114. Xie X. L., Beni G. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1991. Vol. 13, No. 8. P. 841–847. DOI: https://doi.org/10.1109/34.85677.

115. Pal N. R., Bezdek J. C. On cluster validity for the fuzzy c-means model. *IEEE Transactions on Fuzzy Systems*. 1995. Vol. 3, No. 3. P. 370–379. DOI: https://doi.org/10.1109/91.413225.

116. Ю.П. Кльоц, С.В. Мостовий. Прогнозування взаємоблокувань процесів в комп'ютерних системах. *Електротехнічні та комп'ютерні системи* – 2015, № 19 (95) – С.205-209. URL: https://eltecs.op.edu.ua/index.php/journal/article/view/1897/946

117. 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, 14(3), pp. 1484–1492. URL: http://doi.org/10.11591/ijeecs.v14.i3.pp1484-1492

118. Мостовий С., Пирч О. Прогнозування стану взаємоблокування процесів у комп'ютерних системах. *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2025, №4. С.284-292. URL: https://doi.org/10.31891/2219-9365-2025-84-32

119. Мостовий С. Система прогнозування стану взаємоблокування процесів у комп'ютерних системах. *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2026, №2. С.418-424. URL: https://doi.org/10.31891/2219-9365-2026-86-49

120. Мостовий С.В. Архітектура системи прогнозування взаємоблокування процесів у комп'ютерній системі. *Збірник наукових праць за матеріалами XV Всеукраїнської науково-практичної конференції "Актуальні проблеми комп'ютерних наук АПКН-2023"*, м.Хмельницький, 17-18 листопада 2023р. / ХНУ Хмельницький, 2023. С. 343-345. URL: https://kn.khmnu.edu.ua/wp-content/uploads/sites/18/apkn-2023_corpuspaper.pdf

121. Мостовий С.В. Модель та метод побудови сигнатури процесу для прогнозування взаємоблокувань у комп'ютерній системі. *Збірник наукових праць за матеріалами XVI Всеукраїнської науково-практичної конференції "Актуальні проблеми комп'ютерних наук АПКН-2024"*, м.Хмельницький, 15-16 листопада 2024р. / ХНУ Хмельницький, 2024. С. 580-583. URL: https://kn.khmnu.edu.ua/wp-content/uploads/sites/18/apkn-2024-corpuspaper.pdf

122. Мостовий С.В. Метод прогнозування стану взаємоблокування процесів у комп'ютерній системі. *ITSec: Безпека інформаційних технологій: матеріали XV Міжнар.наук.-техн. конф.*, м. Тернопіль, 27-29 трав. 2026 р. / Тернопіль-Київ:ТНТУ-ДУІКТ, 2026. С. 378-380. URL: https://itsec.tntu.edu.ua/#conference-abstracts

123. Мостовий С. Аналіз методів та моделей вирішення задачі взаємоблокування процесів у комп'ютерних системах. *Вісник Хмельницького національного університету. Технічні науки*. 2021, №5. С.27-31. DOI: http://doi.org/10.31891/2307-5732-2021-301-5-27-31

124. Oleg Savenko, Mykola Fedula, Jan Rabchan, Kyrylo Voznyi, Serhii Mostovyi, Anatolii Barabash. The method of parallelization of dynamic sequential system programs using networks of multifaceted processes. *Proceedings of the 7th International Workshop on Intelligent Information Technologies & Systems of Information Security (IntelITSIS-2026)*. / Khmelnytskyi, 2026. P. 20 URL: https://ceur-ws.org/Vol-4242/paper10.pdf

125. Deza M. M., Deza E. Encyclopedia of Distances. 4th ed. Berlin, Heidelberg: Springer, 2016. 756 p. DOI: https://doi.org/10.1007/978-3-662-52844-0.

126. Xu R., Wunsch D. C. Clustering. Hoboken, NJ: John Wiley & Sons, 2009. DOI: https://doi.org/10.1002/9780470382776.

127. Wang L.-X., Mendel J. M. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics*. 1992. Vol. 22, No. 6. P. 1414-1427. DOI: https://doi.org/10.1109/21.199466.

128. Tanenbaum A. S., Bos H. Modern Operating Systems. 5th ed. Pearson, 2022. ISBN 978-0-13-761887-3. URL: https://www.pearson.com/en-us/subject-catalog/p/modern-operating-systems/P200000003295/9780137618873

129. Савенко О.С., Кльоц Ю.П., Мостовий С.В. Дослідження та аналіз блокування процесів в комп'ютерній системі. *Вісник ХНУ* – Хмельницький: ХНУ, 2007. - №3. – С.248-251. URL: https://library.khmnu.edu.ua/old/inf_res/bibliogr/dor/2007/vykl_M.htm

130. С.В. Мостовий. Модель сигнатури процесу в операційній системі. *Вимірювальна та обчислювальна техніка в технологічних процесах* – Хмельницький, 2007, №1. – С.66-68. URL: https://library.khmnu.edu.ua/old/inf_res/bibliogr/dor/2007/vykl_M.htm

131. Савенко О.С., Мостовий С.В. Модель прогнозування стану процесів в комп'ютерній системі. *Радіоелектронні і комп'ютерні системи* – Харків: ХАІ, 2008. - №5 (32). – С.109-115. URL: http://nbuv.gov.ua/UJRN/recs_2008_5_22

132. С.В. Мостовий. Представлення вхідних даних для системи прогнозування стану процесів в персональному комп'ютері. *Вісник ХНУ* – Хмельницький: ХНУ, 2008, №4 – С.145-148. URL: https://library.khmnu.edu.ua/old/pdf/visnyk_tup/2008/2008-4-1t1-t.pdf

133. С.В. Мостовий. Алгоритми і програмні засоби прогнозування стану процесів в персональному комп'ютері. *Вісник ХНУ* – Хмельницький: ХНУ, 2008, №5 – С.124-130. URL: https://scholar.google.com/citations?hl=uk&user=V7QDT1QAAAAJ

134. С.В. Мостовий. Програмні засоби прогнозування стану процесів в персональному комп'ютері під керуванням ОС WINDOWS. *Вимірювальна та обчислювальна техніка в технологічних процесах* – Хмельницький, 2008, №1. – С.122-126. URL: https://scholar.google.com/citations?hl=uk&user=V7QDT1QAAAAJ

135. Савенко О.С., Мостовий С.В. Метод прогнозування стану процесів в персональних комп'ютерах. *Вісник ВПІ* – Вінниця: ВНТУ, 2008. – №, 6. – С.120-125. URL: https://visnyk.vntu.edu.ua/index.php/visnyk/article/view/683

136. Савенко О.С., Мостовий С.В. Алгоритм прогнозування стану процесів в персональному комп'ютері. *Радіоелектронні і комп'ютерні системи* – Харків: ХАІ, 2009. - №7(41). – С.32-36. URL: https://openarchive.nure.ua/bitstreams/e8ffe499-ab5b-44a0-b3f8-b50f7d0135cc/download

137. Савенко О.С., Мостовий С.В. Життєвий цикл процесів комп'ютерної системи. *Радіоелектронні і комп'ютерні системи* – Харків: ХАІ, 2010. - №7(48). – С.35-38. URL: https://openarchive.nure.ua/bitstreams/4a47657d-e0ae-455c-ba0a-16ecb0dffac5/download

138. С.В. Мостовий, О.Л. Ковтун, І.В. Прокопишен. Метод та засоби прогнозування стану взаємоблокування процесів в персональному комп'ютері. *Вісник ХНУ* – Хмельницький: ХНУ, 2010, №1 – С.123-127. URL: https://library.khmnu.edu.ua/old/inf_res/bibliogr/dor/2010/VYKL_VSI_prod.htm

139. Локазюк В.М., Савенко О.С., Мостовий С.В. Модель прогнозування стану взаємоблокування процесів комп’ютерної системи. *Вісник ВПІ* – Вінниця: ВНТУ, 2011. – № 4. – С.130-133. URL: https://visnyk.vntu.edu.ua/index.php/visnyk/article/view/1494

140. Мостовий С.В. Взаємоблокування процесів в комп’ютерних системах. *Вісник ХНУ* – Хмельницький: ХНУ, 2012. – № 4. – С.153-158. URL: https://journals.khnu.km.ua/vestnik/pdf/tech/2012_4/30mos.pdf

141. О.С. Савенко, С.В. Мостовий. Прогнозування потрапляння процесів у стан взаємоблокування. *Вісник ВПІ* – Вінниця: ВНТУ, 2013. – № 2. – С.81-86. URL:

https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/6721/1108.pdf?isAllowed=y&sequence=3

142. Ю.П. Кльоц, С.В. Мостовий. Оцінка часової складності методу прогнозування взаємоблокувань процесів в комп'ютерних системах. *Вісник ХНУ* – Хмельницький: ХНУ, 2014, №1 – С.87-93. URL: https://library.khmnu.edu.ua/old/inf_res/bibliogr/dor/2014/VYKL_VSI.htm

143. Савенко О.С., Мостовий С.В. Модель побудови сигнатури процесу в комп'ютерній системі. *Прогресивні інформаційні технології в науці та освіті. Збірник наукових праць* – Вінниця: Вінницький соціально-економічний інститут Університету "Україна", 2007. – С.58-63. URL: https://library.khmnu.edu.ua/old/inf_res/bibliogr/dor/2007/vykl_M.htm

144. Мостовий С.В. Система прогнозування стану процесів в персональному комп'ютері. *Збірник праць VIII Міжнародної конференції "Інтелектуальний аналіз інформації" (IAI-2008)* – Київ: Просвіта, 2008. – С.308-314. URL: https://scholar.google.com/citations?hl=uk&user=V7QDT1QAAAAJ

145. Савенко О.С., Мостовий С.В. Модель прогнозування стану процесів в комп'ютерній системі. *Збірник праць МНТК "CIET-2008"*, том 1 – Одеса: ОНПУ, 2008. – С.151. URL: https://scholar.google.com/citations?hl=uk&user=V7QDT1QAAAAJ

146. Мостовий С.В. Організація логічного висновку в системі прогнозування стану процесів в персональних комп'ютерах. *Матеріали X міжнародної науково-технічної конференції "Системний аналіз та інформаційні технології" (CAIT-2008)* – Київ: Національний технічний університет України „Київський політехнічний інститут", 2008. – С.386. URL: https://scholar.google.com/citations?hl=uk&user=V7QDT1QAAAAJ

147. Мостовий С.В. Система прогнозування стану процесів в персональному комп'ютері. *Матеріали III міжнародної конференції "Комп'ютерні науки та інформаційні технології" (CSIT-2008)* – Львів: ПП "Вежа і Ко", 2008. – С.309-312. URL: https://scholar.google.com/citations?hl=uk&user=V7QDT1QAAAAJ

148. Мостовий С.В. Прогнозування стану процесів в персональних комп'ютерах. *Міжнар. наук.-техн. конф. "Інтегровані комп'ютерні технології в машинобудуванні" (ІКТМ-2008): Тези доповідей*. – Харків: Нац. аерокосмічний ун.-т "Харк. авіац. ін.-т". – 2008. – С153. URL: https://scholar.google.com/citations?hl=uk&user=V7QDT1QAAAAJ

149. Савенко О.С., Мостовий С.В. Програмна реалізація методу прогнозування стану процесів в персональному комп'ютері. *Матеріали XI міжнародної науково-технічної конференції "Системний аналіз та інформаційні технології" (САІТ-2009)* – Київ: Національний технічний університет України „Київський політехнічний інститут", 2009. – С.381. URL: https://er.chdtu.edu.ua/bitstream/ChSTU/3082/1/ebook-SAIT2009-v1.0-%D1%80%D0%BE%D0%B7%D0%B1%D0%BB%D0%BE%D0%BA%D0%BE%D0%B2%D0%B0%D0%BD%D0%BE.pdf

150. Кльоц Ю.П., Мостовий С.В. Алгоритм прогнозування стану процесів в персональному комп'ютері. *Збірник праць МНТК "СІЕТ-2009"*, том 1 – Одеса: ОНПУ, 2009. – С.187. URL: https://library.khmnu.edu.ua/old/inf_res/bibliogr/dor/2009/VYKL_VSI.htm

151. Oleg Savenko, Yuri Klyots, Sergey Mostovuy. Method and algorithms of forecasting of a status of deadlock of processes in personal computers. *Праці IV міжнародної конференції "Сучасні комп'ютерні системи та мережі: розробка та використання" (ACSN'2009)* – Львів: Львівський національний університет "Львівська політехніка", 2009. – С.111-115. URL: https://library.khmnu.edu.ua/old/inf_res/bibliogr/dor/2009/inoz.htm

152. О.С. Савенко, С.В. Мостовий. Особливості програмної реалізації методу прогнозування стану процесів в персональному комп'ютері у ОС Linux. *Прогресивні інформаційні технології в науці, освіті та економіці. Збірник наукових праць* – Вінниця: Вінницький кооперативний інститут, 2009. – С.-216-222. URL: https://library.khmnu.edu.ua/old/inf_res/bibliogr/dor/2009/VYKL_VSI.htm

153. В.М. Локазюк, О.С. Савенко, С.В.Мостовий. Модель прогнозування стану взаємоблокування процесів комп'ютерної системи. *Матеріали X міжнародної конференції "Контроль і управління в складних системах (КУСС-2010)"* – Вінниця: Вінницький національний технічний університет, 2010. – С.107. URL: https://library.khmnu.edu.ua/old/inf_res/bibliogr/dor/2010/VYKL_VSI_prod.htm

154. О.С. Савенко, С.В. Мостовий. Модель станів комп'ютерних систем для визначення взаємоблокування процесів. *Збірник праць МНТК "CIET-2011"* - С.102. URL: https://library.khmnu.edu.ua/old/inf_res/bibliogr/dor/2011/VYKL_VSI_prod.htm

155. О.С. Савенко, С.В. Мостовий. Дослідження станів комп'ютерних систем для задачі прогнозування стану взаємоблокування процесів. *Збірник наукових праць Бучацького інституту менеджменту і аудиту*, №7, 2011 - С.- 411-415. URL: https://library.khmnu.edu.ua/old/inf_res/bibliogr/dor/2011/VYKL_VSI_prod.htm

156. О.С. Савенко, С.В. Мостовий. Прогнозування потрапляння процесів у стан взаємоблокування в комп'ютерних системах. *Контроль і управління в складних системах: Тези доповідей міжнародної конференції* - Вінниця: ВНТУ, 2012 - с.105-106. URL: https://library.khmnu.edu.ua/old/inf_res/bibliogr/dor/2012/VYKL_VSI_prod.htm

157. Мостовий С.В., Касіяник О.О. Програмні засоби визначення зміни стану процесів в комп'ютерній системі. *Інтелектуальні технології в системному програмуванні: Збірник праць науково-практичної конференції молодих вчених та студентів* - Хмельницький: ПП Гонта А.С., 2012 - с.98-105. URL: https://library.khmnu.edu.ua/old/inf_res/bibliogr/dor/2012/VYKL_VSI.htm

158. Pastukh O., Yatsyshyn V. Development of Software for Neuromarketing Based on Artificial Intelligence and Data Science Using High-Performance Computing and Parallel Programming Technologies. *Scientific Journal of TNTU*, 2024, Vol. 113, No. 1, pp. 143–149. DOI: https://doi.org/10.33108/visnyk_tntu2024.01.143

159. Yatsyshyn V., Pastukh O., Palamar A., Zharovskyi R. Technology of Relational Database Management Systems Performance Evaluation during Computer

Systems Design. *Scientific Journal of TNTU*, 2023, Vol. 109, No. 1, pp. 54–65. DOI: https://doi.org/10.33108/visnyk_tntu2023.01.054

160. Zadvornyi O., Pastukh O. Comparative Analysis of Machine Learning Algorithms for Market Capitalization Time Series Forecasting. *Scientific Journal of TNTU*, 2025, Vol. 119, No. 3, pp. 26–34. DOI: https://doi.org/10.33108/visnyk_tntu2025.03.026.

ДОДАТКИ

ДОДАТОК А

Список публікацій здобувача за темою дисертації

Наукові праці, в яких опубліковані основні наукові результати дисертації

1. Ю.П. Кльоц, С.В. Мостовий. Прогнозування взаємоблокувань процесів в комп'ютерних системах. *Електротехнічні та комп'ютерні системи.* – 2015, № 19 (95) – С.205-209. URL: https://eltecs.op.edu.ua/index.php/journal/article/view/1897/946

2. 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, 14 (3), pp. 1484–1492. URL: http://doi.org/10.11591/ijeecs.v14.i3.pp1484-1492 (індексована в наукометричній базі Scopus)

3. Мостовий С. Аналіз методів та моделей вирішення задачі взаємоблокування процесів у комп'ютерних системах. *Вісник Хмельницького національного університету. Технічні науки*. 2021, №5. С.27-31. URL: http://doi.org/10.31891/2307-5732-2021-301-5-27-31

4. Мостовий С., Пирч О. Прогнозування стану взаємоблокування процесів у комп'ютерних системах. *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2025, №4. С.284-292. URL: https://doi.org/10.31891/2219-9365-2025-84-32

5. Мостовий С. Система прогнозування стану взаємоблокування процесів у комп'ютерних системах. *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2026, №2. С.418-424. URL: https://doi.org/10.31891/2219-9365-2026-86-49

Праці, які засвідчують апробацію матеріалів дисертації

6. Мостовий С.В. Архітектура системи прогнозування взаємоблокування процесів у комп'ютерній системі. *Збірник наукових праць за матеріалами XV Всеукраїнської науково-практичної конференції "Актуальні проблеми комп'ютерних наук АПКН-2023"*, м.Хмельницький, 17-18 листопада 2023р. / ХНУ Хмельницький, 2023. С. 343-345. URL: https://kn.khmnu.edu.ua/wp-content/uploads/sites/18/apkn-2023_corpuspaper.pdf

7. Мостовий С.В. Модель та метод побудови сигнатури процесу для прогнозування взаємоблокувань у комп'ютерній системі. *Збірник наукових праць за матеріалами XVI Всеукраїнської науково-практичної конференції "Актуальні проблеми комп'ютерних наук АПКН-2024"*, м.Хмельницький, 15-16 листопада 2024р. / ХНУ Хмельницький, 2024. С. 580-583. URL: https://kn.khmnu.edu.ua/wp-content/uploads/sites/18/apkn-2024-corpuspaper.pdf

8. Мостовий С.В. Метод прогнозування стану взаємоблокування процесів у комп'ютерній системі. ITSec: *Безпека інформаційних технологій: матеріали XV Міжнар.наук.-техн. конф.*, м. Тернопіль, 27-29 трав. 2026 р. / Тернопіль-Київ:ТНТУ-ДУІКТ, 2026. С. 378-380 URL: https://itsec.tntu.edu.ua/#conference-abstracts

9. Oleg Savenko, Mykola Fedula, Jan Rabchan, Kyrylo Voznyi, Serhii Mostovyi, Anatolii Barabash. The method of parallelization of dynamic sequential system programs using networks of multifaceted processes. *Proceedings of the 7th International Workshop on Intelligent Information Technologies & Systems of Information Security (IntelITSIS-2026)*. / Khmelnytskyi, 2026. P. 20 URL: https://ceur-ws.org/Vol-4242/paper10.pdf (індексована в наукометричній базі Scopus)

ДОДАТОК Б

Акти впровадження

«Затверджую»
Директор ТОВ «ІТТ»
Сімогук В.С.
«___» червня 2026 р.

АКТ
про впровадження результатів дисертаційної роботи
старшого викладача кафедри кібербезпеки Хмельницького національного університету Мостового Сергія Володимировича
«Метод та засоби прогнозування стану взаємоблокування процесів в комп'ютерних системах»

Результати дисертаційної роботи Мостового С.В. старшого викладача кафедри кібербезпеки Хмельницького національного університету впроваджені на ТОВ «ІТТ».

При розробленні засобів прогнозування стану взаємоблокування процесів в комп'ютерних системах на ТОВ «ІТТ» були отримані Мостовим С.В. особисто такі результати:

1) архітектура засобів прогнозування стану взаємоблокування процесів у комп'ютерних системах, яка, на відміну від відомих підходів, орієнтованих переважно на виявлення або усунення взаємоблокування після його виникнення, передбачає поєднання модулів збору даних про процеси та ресурси, побудови сигнатур процесів, формування бази нечітких правил і прогнозування стану взаємоблокування, що забезпечує завчасне оцінювання ризику переходу процесів у стан взаємоблокування та створює основу для прийняття превентивних рішень щодо керування ресурсами комп'ютерної системи;

2) метод прогнозування стану взаємоблокування процесів у комп'ютерній системі, який, на відміну від відомих методів виявлення взаємоблокувань після їх виникнення або запобігання їм на основі наперед заданих обмежень розподілу ресурсів, забезпечує завчасне оцінювання ризику переходу процесів у стан взаємоблокування шляхом обробки сигнатур процесів базою нечітких правил, сформованою за результатами кластеризації, що дає змогу враховувати поточний ресурсний стан системи, особливості взаємодії процесів із ресурсами та формувати підстави для прийняття превентивних рішень щодо керування ресурсами комп'ютерної системи.

Отриманні результати дозволили підвищити стабільність роботи комп'ютерних систем та зменшити часові витрати на виконання базових задач у ТОВ «ІТТ» шляхом удосконалення архітектури засобу та методу прогнозування стану взаємоблокування процесів у комп'ютерних системах більш, ніж у 2,3 рази.

Цей акт не є підставою для фінансових розрахунків.

Заступник директора Іванов О.В.

Системний адміністратор Веремеєнко В.А.

ДОВІДКА

про впровадження результатів дисертаційної роботи
старшого викладача кафедри кібербезпеки Хмельницького національного університету Мостового Сергія Володимировича
за темою «Метод та засоби прогнозування стану взаємоблокування процесів в комп'ютерних системах»

Результати дисертаційної роботи старшого викладача кафедри кібербезпеки Хмельницького національного університету Мостового С.В. впроваджені в ПП «Авіві».

У процесі впровадження методів та засобів прогнозування стану взаємоблокування процесів в інформаційних системах ПП «Авіві» були використані результати, одержані Мостовим С.В. особисто:

1) розроблено архітектуру засобів прогнозування стану взаємоблокування процесів у комп'ютерних системах, яка, на відміну від відомих підходів, орієнтованих переважно на виявлення або усунення взаємоблокування після його виникнення, передбачає поєднання модулів збору даних про процеси та ресурси, побудови сигнатур процесів, формування бази нечітких правил і прогнозування стану взаємоблокування, що забезпечує завчасне оцінювання ризику переходу процесів у стан взаємоблокування та створює основу для прийняття превентивних рішень щодо керування ресурсами комп'ютерної системи;

2) розроблено метод прогнозування стану взаємоблокування процесів у комп'ютерній системі, який, на відміну від відомих методів виявлення взаємоблокувань після їх виникнення або запобігання їм на основі наперед заданих обмежень розподілу ресурсів, забезпечує завчасне оцінювання ризику переходу процесів у стан взаємоблокування шляхом обробки сигнатур процесів базою нечітких правил, сформованою за результатами кластеризації, що дає змогу враховувати поточний ресурсний стан системи, особливості взаємодії процесів із ресурсами та формувати підстави для прийняття превентивних рішень щодо керування ресурсами комп'ютерної системи.

Упровадження зазначених результатів в ПП «Авіві» на 9% зменшило час обробки запитів до інформаційних ресурсів підприємства без використання додаткового обладнання.

Ця довідка не є підставою для фінансових розрахунків.

29.06.2026 р.

Директор ПП «Авіві» _____ В. В. Аскеров

«Затверджую»
В.о. ректора Хмельницького
національного університету
_____Віктор ЛОПАТОВСЬКИЙ
«06» 06 2026 р.

АКТ
про впровадження в освітній процес Хмельницького національного університету результатів дисертаційної роботи старшого викладача кафедри кібербезпеки Мостового Сергія Володимировича
«Метод та засоби прогнозування стану взаємоблокування процесів в комп'ютерних системах»

Ми, комісія в складі: декана факультету інформаційних технологій, професора кафедри комп'ютерної інженерії та інформаційних систем, д.т.н., професора Говорущенко Т. О. (голова комісії), завідувача кафедри кібербезпеки, к.т.н., доцента Кльоца Ю.П., доцента кафедри кібербезпеки, д.ф. Петляк Н.С., завідувача кафедри комп'ютерної інженерії та інформаційних систем, д.ф., доцента Павлової О. О., доцента кафедри комп'ютерної інженерії та інформаційних систем, к.т.н., доцента Нічепорука А.О. склала акт про те, що результати дисертаційної роботи Мостового С.В. впроваджені та використовуються в освітньому процесі Хмельницького національного університету при викладанні дисциплін на кафедрі кібербезпеки для здобувачів спеціальності F5 Кібербезпека, зокрема в курсах «Основи інформаційної безпеки», «Операційні системи та технології їх захисту» та на кафедрі комп'ютерної інженерії та інформаційних систем для здобувачів спеціальності F7 Комп'ютерна інженерія, зокрема в курсах «Системне програмне забезпечення», «Технологія проєктування спеціалізованих операційних систем».

При викладанні цих дисциплін викладачами кафедр використовувалися наступні матеріали досліджень, отримані Мостовим С.В. особисто:

1. Архітектура засобів прогнозування стану взаємоблокування процесів у комп'ютерних системах, особливістю якої є поєднання модулів збору даних, побудови сигнатур процесів, автоматизованого формування бази нечітких правил та прогнозування взаємоблокування, що дозволяє забезпечити своєчасне виявлення передумов виникнення взаємоблокування.

2. Метод побудови сигнатури процесу в комп'ютерній системі, який забезпечує формування впорядкованого вектора ознак із використанням ідентифікаційних, ресурсних, часових та поведінкових характеристик процесів, що підвищує інформативність їх подальшого аналізу.

3. Метод побудови бази нечітких правил на основі алгоритму нечіткої кластеризації Fuzzy C-Means, який забезпечує автоматизоване формування правил нечіткого логічного висновку та підвищує адаптивність системи до змін поведінки процесів.

4. Метод прогнозування стану взаємоблокування процесів, який використовує сформовану базу нечітких правил для оцінювання ризику переходу процесів у стан взаємоблокування та забезпечує підвищення достовірності прогнозування.

Отримані матеріали досліджень дозволили розробити лабораторні практикуми та оновити навчально-методичне забезпечення дисциплін із використанням архітектури прогнозування взаємоблокування процесів, методів побудови сигнатур процесів, автоматизованого формування бази нечітких правил і прогнозування стану взаємоблокування.

Говорущенко Т. О.

Кльоц Ю.П.

Павлова О. О.

Петляк Н.С.

Нічепорук А.О.

ДОДАТОК В

Інструкція користувача програмного засобу Deadlock Prediction System v1.0

1. Призначення програмного засобу

Програмний засіб Deadlock Prediction System призначений для імітаційного моделювання ресурсної взаємодії процесів у комп'ютерній системі, дослідження умов виникнення взаємоблокувань, порівняння методів їх виявлення та запобігання, а також експериментального оцінювання методу прогнозування переходу процесів у стан взаємоблокування.

Програмна реалізація дає змогу моделювати виконання множини процесів, їх конкуренцію за неподілювані системні ресурси, утримання й очікування ресурсів, виникнення циклів очікування, примусове звільнення ресурсів і перезапуск процесів.

Для прогнозування небезпечних ресурсних ситуацій використовується подання поточного стану ресурсів і запиту процесу у вигляді вектора ознак. Результат прогнозування визначається на основі попередньо сформованих центрів кластерів і бази правил.

Програмний засіб також забезпечує автоматичне накопичення статистичних даних про перебіг моделювання, виявлені взаємоблокування, час очікування ресурсів, перезапуски процесів, результати прогнозування та обчислювальні витрати досліджуваних алгоритмів.

Версія програмного засобу, що відповідає експериментальним дослідженням, розміщена за адресою: https://github.com/Serhii-Mostovyi/Deadlock-prediction-system/releases/tag/v1.0

2. Склад програмного проєкту

Після завантаження та розпакування проєкт має таку структуру:

Deadlock-prediction-system/



├── README.md

```
├── requirements.txt
├── .gitignore
│
├── src/
│   └── deadlock_simulation.py
│
├── data/
│   ├── processes.csv
│   ├── processes_500.csv
│   ├── resources.csv
│   ├── cluster_centers.csv
│   └── rules.csv
│
├── results/
│
└── examples/
    └── results/
```

Файл src/deadlock_simulation.py містить основний програмний код імітаційної моделі. Каталог data містить вхідні дані, необхідні для виконання моделювання. Каталог results використовується для збереження результатів поточного експериментального прогону. Каталог examples/results містить приклади результатів виконання моделі. Файл requirements.txt містить перелік зовнішніх програмних залежностей.

3. Підготовка програмного середовища

Для роботи програмного засобу необхідно мати встановлений інтерпретатор Python та пакет SimPy, який використовується для дискретно-подієвого моделювання процесів і ресурсів. Після завантаження проєкту необхідно відкрити командний рядок або термінал у кореневому каталозі проєкту та виконати:

python -m pip install -r requirements.txt

У результаті буде встановлена необхідна зовнішня залежність SimPy. Модулі csv, time і pathlib, що також використовуються програмою, належать до стандартних засобів Python.

4. Вхідні дані

Основні вхідні дані розташовані в каталозі data.

4.1. Файл процесів

За замовчуванням для моделювання використовується файл data/processes_500.csv. Файл визначає множину процесів і часові характеристики їх виконання. Для кожного процесу задаються такі поля:

Поле	Призначення
name	унікальний ідентифікатор процесу
arrival	момент надходження процесу в систему
delay1	тривалість виконання перед запитом першого ресурсу
res1	ідентифікатор першого ресурсу
delay2	тривалість виконання перед запитом другого ресурсу
res2	ідентифікатор другого ресурсу
delay_release2	тривалість виконання перед звільненням другого ресурсу
delay_release1	тривалість виконання перед звільненням першого ресурсу
delay_finish	тривалість завершальної ділянки виконання процесу

Значення часових параметрів і моменту надходження не повинні бути від’ємними. У межах одного процесу значення res1 і res2 повинні позначати різні ресурси. Кожний ідентифікатор ресурсу, зазначений у res1 або res2, повинен існувати у файлі resources.csv.

4.2. Файл ресурсів

Файл data/resources.csv описує системні ресурси. Він містить два обов’язкові поля:

Поле	Призначення
id	унікальний ідентифікатор ресурсу
capacity	кількість доступних екземплярів ресурсу

Поточна версія моделі орієнтована на неподілювані одноекземплярні ресурси, тому для всіх ресурсів використовується capacity = 1.

4.3. Центри кластерів

Файл data/cluster_centers.csv містить центри кластерів, які використовуються методами прогнозування. Першим полем є CenterID, інші поля містять координати відповідного центра кластера. Кількість координат кожного центра повинна дорівнювати подвоєній кількості ресурсів системи, оскільки вектор стану складається з двох частин: ознак поточної зайнятості ресурсів та ознак поточного запиту ресурсу.

4.4. База правил

Файл data/rules.csv містить правила, які використовуються під час прогнозування. Обов’язковими полями є:

Поле	Призначення
RuleID	ідентифікатор правила
Cluster	кластер, з яким пов’язане правило
Center	ідентифікатор відповідного центра
Output	числове значення виходу правила

Значення Output повинно належати діапазону від 0 до 1.

5. Вибір методу обробки взаємоблокувань

Перед запуском експерименту необхідно відкрити файл src/deadlock_simulation.py і встановити необхідне значення параметра STRATEGY = "значення". Програма підтримує такі режими роботи:

"ignore" – ігнорування взаємоблокувань із пасивним моніторингом для накопичення статистики;

"detect_only" – виявлення взаємоблокування за графом очікування без керуючого втручання;

"preempt" – виявлення взаємоблокування з примусовим звільненням ресурсів, відкатом і перезапуском обраного процесу;

"no_hold_wait" – запобігання умові hold-and-wait шляхом атомарного захоплення необхідної пари ресурсів;

"banker" – застосування алгоритму банкіра;

"rag" – застосування графа розподілу ресурсів із дугами заявок;

"heuristic_observe" – прогнозування після відмови у негайній видачі ресурсу без впливу прогнозу на подальше виконання;

"heuristic_prevent" – прогнозування після відмови у видачі ресурсу з виконанням превентивної дії при високому рівні ризику;

Режим heuristic_observe призначений насамперед для експериментального оцінювання достовірності прогнозування. Результат прогнозу в цьому режимі реєструється, але не змінює порядок виконання процесів.

У режимі heuristic_prevent, якщо результат прогнозування перевищує встановлений поріг ризику, виконується вибір одного процесу з конфліктної групи. Як жертва вибирається активний процес із найменшим часом перебування в системі. Його ресурси звільняються, після чого процес запускається повторно.

6. Основні параметри моделювання

У початковій частині файла deadlock_simulation.py задаються основні параметри експерименту:

CPU_COUNT = 2

QUANTUM = 2

MAX_LIFETIME = 10000

CHECK_INTERVAL = 1

VERBOSE = False

де:

CPU_COUNT – кількість модельованих процесорних ядер;

QUANTUM – величина кванта процесорного часу;

MAX_LIFETIME – максимально допустимий час перебування окремої спроби процесу в системі;

CHECK_INTERVAL – періодичність перевірки стану системи монітором взаємоблокувань;

VERBOSE – режим детального виведення подій моделювання.

Для масових експериментів рекомендується VERBOSE = False, оскільки в цьому випадку детальний журнал подій не виводиться в консоль.

Для методів прогнозування також використовується параметр HEURISTIC_THRESHOLD = 0.6. Він визначає граничне значення, після перевищення якого результат прогнозування інтерпретується як ризиковий.

7. Запуск програмного засобу

Після встановлення залежностей, перевірки вхідних CSV-файлів та вибору значення STRATEGY необхідно перейти до кореневого каталогу проєкту й виконати команду:

python src/deadlock_simulation.py

Під час запуску програма:

1) створює середовище дискретно-подієвого моделювання;
2) завантажує опис системних ресурсів;
3) перевіряє коректність вхідних даних;
4) для режимів прогнозування завантажує центри кластерів і базу правил;
5) формує модельні процеси відповідно до файла processes_500.csv;
6) виконує моделювання ресурсної взаємодії;
7) контролює виникнення взаємоблокувань;
8) накопичує статистичні характеристики;
9) після завершення моделювання записує результати у файли.

Після нормального завершення програми в командному рядку відображається коротка підсумкова інформація про кількість початкових процесів, кількість спроб виконання, завершені процеси, примусові завершення, відкати, перезапуски та зафіксовані взаємоблокування.

8. Результати моделювання

Результати поточного експерименту зберігаються в каталозі "results/". Ім’я кожного файла містить назву використаної стратегії. Наприклад, для STRATEGY = "heuristic_prevent" формуються файли "raw_stats_heuristic_prevent.csv",

"summary_heuristic_prevent.txt", "deadlock_log_heuristic_prevent.csv", "prediction_log_heuristic_prevent.csv.

Файл raw_stats_<strategy>.csv містить детальні статистичні дані щодо кожного процесу та кожної повторної спроби його виконання, зокрема:

– статус процесу;
– моменти надходження та завершення;
– час виконання;
– накопичений час очікування ресурсів;
– кількість блокувань;
– кількість перемикань квантів;
– участь у взаємоблокуваннях;
– кількість відкатів;
– результати прогнозування.

Файл summary_<strategy>.txt містить узагальнені результати експериментального прогону:

– кількість початкових процесів;
– загальну кількість спроб із урахуванням перезапусків;
– кількість успішно завершених процесів;
– кількість примусово завершених процесів;
– кількість відкатів і перезапусків;
– середні та максимальні часові характеристики;
– показники очікування ресурсів;
– кількість і характеристики взаємоблокувань;
– реальний час виконання симуляції;
– обчислювальні витрати алгоритмів.

Для режимів прогнозування додатково формується статистика використання правил. У режимі heuristic_observe розраховуються показники якості прогнозування для запитів, на яких прогноз фактично виконувався: TP, FP, TN, FN, Accuracy, Precision, Recall, Specificity та F1-score.

Файл deadlock_log_<strategy>.csv містить журнал зафіксованих взаємоблокувань із зазначенням time – момент виявлення взаємоблокування, size – кількість процесів у циклі, participants – ідентифікатори процесів-учасників.

Файл prediction_log_<strategy>.csv містить детальний журнал виконаних прогнозів. Для кожної події зберігаються момент прогнозування, процес, етап запиту ресурсу, номер спроби, застосоване правило, значення виходу правила, відстань до центра, результат класифікації ризику та інформація про виконане втручання.

9. Проведення порівняльного експерименту

Для забезпечення порівнюваності результатів різних методів доцільно використовувати однакові файли processes_500.csv та resources.csv. Для проведення окремого експерименту необхідно:

1. вибрати одну зі стратегій у параметрі STRATEGY;
2. запустити deadlock_simulation.py;
3. зберегти сформовані результати;
4. змінити значення STRATEGY;
5. повторити запуск за незмінних вхідних даних;
6. порівняти підсумкові файли відповідних стратегій.

Наприклад, для дослідження запропонованого методу доцільно виконати окремі прогони STRATEGY = "heuristic_observe" та STRATEGY = "heuristic_prevent". Перший режим дає змогу дослідити достовірність прогнозування без зміни природного розвитку ресурсної ситуації, другий – оцінити ефективність використання прогнозу для превентивного втручання.

10. Контроль коректності вхідних даних

Перед початком моделювання програма автоматично перевіряє основні умови коректності даних. Запуск припиняється з повідомленням про помилку, якщо:

- файл процесів не містить обов’язкових колонок;
- ідентифікатор процесу порожній або повторюється;
- часові параметри мають від’ємні значення;

– процес запитує один і той самий ресурс як res1 і res2;

– у процесі зазначено ресурс, якого немає у resources.csv;

– ідентифікатори ресурсів повторюються;

– для ресурсу задано capacity, відмінне від 1;

– файл правил не містить необхідних колонок;

– значення Output правила не належить діапазону [0;1];

– правило посилається на відсутній центр кластера;

– розмірність центра кластера не відповідає кількості ресурсів системи.

У разі появи такої помилки необхідно виправити відповідний CSV-файл і повторити запуск програми.

11. Завершення роботи

Після завершення експериментального прогону додаткових дій для коректного завершення програми не потрібно. Програма самостійно завершує середовище моделювання після виконання процесів, формує статистичні характеристики та записує результати в каталог results. Для проведення наступного експерименту необхідно змінити потрібні параметри або значення STRATEGY і повторно запустити програму.

12. Посилання на програмну реалізацію

Програмний засіб Deadlock Prediction System, версія v1.0, доступний у репозиторії GitHub https://github.com/Serhii-Mostovyi/Deadlock-prediction-system.

Зафіксована версія програмної реалізації, використана для експериментальних досліджень: https://github.com/Serhii-Mostovyi/Deadlock-prediction-system/releases/tag/v1.0.

ДОДАТОК Г

Результати експериментів

Результати імітаційного моделювання, використані для формування таблиць 4.1–4.5, розміщено у відкритому репозиторії програмної реалізації системи прогнозування взаємоблокувань у складі релізу Deadlock Prediction System – Dissertation Version 1.0: https://github.com/Serhii-Mostovyi/Deadlock-prediction-system/releases/tag/v1.0.

Відповідні CSV-файли містять результати одного прогону імітаційного моделювання для 10 000 процесів для кожного досліджуваного методу. Імена файлів сформовано за шаблоном:

table_4_X_METHOD.csv,

де 4_X відповідає номеру таблиці дисертаційної роботи, а METHOD – скороченому позначенню досліджуваного методу.

Використано такі групи файлів:

table_4_1_*.csv – результати моделювання для 30 ресурсів і 5 % процесів, що потрапляють до стану взаємоблокування;

table_4_2_*.csv – результати для 100 ресурсів і 5 % процесів;

table_4_3_*.csv – результати для 300 ресурсів і 5 % процесів;

table_4_4_*.csv – результати для 30 ресурсів і 10 % процесів;

table_4_5_*.csv – результати для 30 ресурсів і 15 % процесів.

Для кожної конфігурації наведено сім файлів відповідно до досліджуваних методів:

АБ – алгоритм банкіра;

ГО – метод аналізу графа очікування;

ГПР – метод примусового вилучення ресурсу або відкату;

ЗП – метод заборони на переривання;

ІГН – метод ігнорування взаємоблокувань;

МП – запропонований метод прогнозування взаємоблокувань;

ПВР – метод примусового вилучення ресурсу.