

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІНІСТЕРСТВА ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

КАЧУР АРТЕМ ВЯЧЕСЛАВОВИЧ

УДК 004.3:004.75:004.49

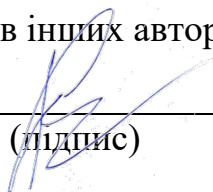
ДИСЕРТАЦІЯ

МЕТОДИ ТА ЗАСОБИ ЗАБЕЗПЕЧЕННЯ РЕЗИЛЬЄНТНОСТІ СИСТЕМ
ДОПОВНЕНОЇ ТА ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ

123 Комп'ютерна інженерія
(шифр і назва спеціальності)
12 Інформаційні технології
(галузь знань)

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і
текстів інших авторів мають посилання на відповідне джерело



А.В. Качур

(підпис)

Наукові керівники:

Лисенко Сергій Миколайович, доктор технічних наук, професор

Корецька Людмила Олександрівна, кандидат технічних наук, доцент

АНОТАЦІЯ

Качур А.В. Методи та засоби забезпечення резильєнтності систем доповненої та віртуальної реальності. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії з галузі знань 12 Інформаційні технології за спеціальністю 123 Комп'ютерна інженерія. – Хмельницький національний університет, Хмельницький, 2026.

Забезпечення резильєнтного та безперервного функціонування систем віртуальної реальності в умовах швидкоплинних змін зовнішнього середовища є визначальною науковою проблемою сучасних досліджень.

У дисертації виконано комплексний аналіз архітектурних підходів до побудови VR-систем із врахуванням апаратних, програмних, мережових та користувацьких компонентів, а також методів забезпечення їхньої резильєнтності. Розроблено математичну модель, що описує окремі функції впливу таких факторів, як температура навколишнього середовища, рівень шуму, навантаження системи та поведінка користувача, із призначенням вагових коефіцієнтів для кожної пари «фактор-компонент». Запропоновано інтегральний показник резильєнтності, який узагальнює ці функції та відображає загальний рівень резильєнтності системи. Для моделювання динамічних змін середовища й імітації збоїв реалізовано Simulink-модель із випадковими коливаннями параметрів і раптовими стрибками. Надалі проведено серію експериментів із апробацією запропонованого підходу, виконано порівняльний аналіз продуктивності та точності виявлення деградаційних станів, а також обґрунтовано напрямки подальших досліджень щодо розширення моделі та впровадження адаптивних механізмів самовідновлення.

Об'єктом дослідження є процес забезпечення резильєнтності систем віртуальної та доповненої реальності в умовах динамічно змінюваного зовнішнього середовища.

Предметом дослідження є архітектури VR-системи та методи оцінювання стану VR-системи й синтезу мітигаційних стратегій для відновлення працездатності VR-системи в умовах динамічно змінюваного зовнішнього середовища.

Метою дисертаційного дослідження є забезпечення резильєнтності систем віртуальної та доповненої реальності шляхом розроблення архітектури VR-системи та методів оцінювання стану VR-системи й синтезу мітигаційних стратегій для відновлення працездатності VR-системи в умовах динамічно змінюваного зовнішнього середовища.

Наукова новизна одержаних результатів полягає в наступному:

вперше розроблено:

1) метод синтезу мітигаційних стратегій, який, на відміну від відомих методів здійснює мультикритеріальний Парето-ефективний синтез мітигаційних стратегій шляхом одночасної оптимізації цільових функцій параметрів резильєнтності з калібрувальними коефіцієнтами, і який дозволяє забезпечувати резильєнтність системи в умовах динамічно змінюваного зовнішнього середовища;

2) метод оцінювання стану VR-системи, який, на відміну від відомих методів формує інтегральний показник резильєнтності, який ґрунтується на оцінці впливу факторів динамічно змінюваного середовища на компоненти системи і використовує систему вагових коефіцієнтів, отриману шляхом застосування нечіткої логіки;

набула подальшого розвитку:

3) архітектура VR-систем, яка на відміну від відомих архітектур інтегрує в собі комплекс компонентів, пов'язаних із програмно-апаратними засобами, методи оцінювання, синтезу мітигаційних стратегій в єдиний програмно-апаратний засіб, що дозволяє здійснювати резильєнтне функціонування систем;

удосконалено:

4) модель резильєнтного функціонування систем віртуальної та доповненої реальності, у якій на відміну від відомих моделей враховано вплив зовнішніх факторів, таких як зміна стану мережі, поведінка користувачів, навантаження на систему, апаратні обмеження та фактори навколишнього середовища на основні компоненти системи, включаючи апаратне та програмне забезпечення, дані та користувачів.

Практичне значення отриманих результатів. Практична цінність отриманих результатів полягає у доведенні розроблених теоретичних положень до рівня

прикладних методів і програмного засобу для оцінювання стану систем віртуальної та доповненої реальності, виявлення критичних факторів деградації та синтезу мітигаційних стратегій. Отримані результати можуть бути використані розробниками XR-систем, системними архітекторами, інженерами з надійності та відмовостійкості, а також фахівцями дослідницьких і випробувальних лабораторій під час проєктування, тестування, експлуатації та модернізації програмно-апаратних комплексів віртуальної та доповненої реальності. Результати роботи можуть бути використані під час розроблення та випробування XR-систем для освітніх і тренувальних комплексів, медичних симуляторів, промислових систем візуалізації та підготовки персоналу, а також інших застосувань, у яких критичними є безперервність роботи, стабільність продуктивності, допустимий час відновлення та безпека користувача. Дисертаційне дослідження виконувалось у рамках науково-дослідної тематики Хмельницького національного університету: держбюджетної науково-дослідної теми №1Б-2026 «Система забезпечення резильєнтності до витоку конфіденційної інформації в корпоративних мережах в умовах впливів комп'ютерних атак» (номер держреєстрації 0126U002082), в якій автор дисертації був виконавцем. Практичні здобутки було використано в навчальному процесі кафедри комп'ютерної інженерії та інформаційних систем Хмельницького національного університету для практичних занять із курсів «Системне програмне забезпечення» та «Об'єктно-орієнтовані технології програмування». Така комплексна реалізація сприяє подальшому розвитку наукових досліджень у сфері резильєнтності віртуальних середовищ і може бути адаптована під потреби промислових та академічних лабораторій.

У вступі обґрунтовано актуальність забезпечення резильєнтності систем віртуальної та доповненої реальності в умовах динамічних змін середовища. Визначено мету, об'єкт, предмет і задачі дослідження, сформульовано наукову новизну та практичне значення одержаних результатів. Наведено відомості про особистий внесок здобувача, апробацію та публікації за темою дисертації.

У першому розділі проведено огляд сучасних підходів до забезпечення резильєнтності систем доповненої та віртуальної реальності. Спочатку

сформульовано означення резильєнтності XR-системи та умови збереження функціонування і завершеного відновлення, а також обґрунтовано актуальність дослідження: із зростанням застосування XR у медицині, освіті, індустрії розваг і промисловості ціна раптової деградації імерсивного середовища суттєво зростає. Далі проаналізовано відомі методи за сімома напрямками: кількісне оцінювання резильєнтності та моделі деградації і відновлення, самовідновлення програмного забезпечення на основі контрольних точок, омолодження і мікроперезапуску, автономне керування та самоадаптивні архітектури із замкненим контуром, проєктування, орієнтоване на відновлення, апаратна надлишковість і відмовостійкість, перевірка резильєнтності контрольованим внесенням відмов та методи, специфічні для імерсивних систем. За результатами огляду визначено прогалини щодо оцінювання стану, синтезу мітигаційних дій та кількісного підтвердження відновлення, які формують постановку задачі дослідження.

У другому розділі розроблено методичний базис для кількісної оцінки резильєнтності XR-систем. Система розглядається як чотирикомпонентний вектор, що включає апаратне забезпечення, програмне забезпечення, дані та користувачі. На кожен з компонентів впливають п'ять джерел зовнішніх факторів: стан мережі, навколишнє середовище, поведінка користувача, навантаження на систему та апаратні обмеження. Для кожної пари «фактор–компонент» запропоновано окрему функцію оцінки, що повертає булевий або нормалізований числовий результат, а їхній набір із двадцяти формул дозволяє відстежувати ключові показники в реальному часі. Далі описано побудову матриці взаємодій факторів із компонентами та введення інтегрального показника резильєнтності як зваженої суми цих функцій із вагою, що відображає важливість кожного впливу. Такий метод створює основу для подальшої оптимізації: на підставі одержаної метрики можна ранжувати слабкі місця системи й обирати пріоритетні засоби їхньої нейтралізації.

У третьому розділі описано цикл забезпечення резильєнтності VR-систем, який поєднує оцінювання стану, локалізацію вузьких місць, синтез мітигаційних стратегій та переоцінку системи у замкнений контур. Формалізовано критерії резильєнтності: надійність, доступність, відмовостійкість, цілісність даних, час відновлення,

стабільність продуктивності та безпеку користувача, і на їх основі поставлено задачу мультикритеріального Парето-ефективного синтезу мітигаційних стратегій із параметричним перетворювачем мітигаційного вектора та калібрувальними коефіцієнтами. Запропоновано метод синтезу резильєнтних XR-архітектур як обернене відображення цілей резильєнтності на параметри проектування із розв'язанням задачі оптимізації нормалізованої зваженої агрегації критеріїв.

У четвертому розділі описано реалізацію програмно-технічного засобу для синтезу та оцінювання резильєнтності VR-систем у вигляді кросплатформного десктопного застосунку з обчислювальним ядром на TypeScript, інтерфейсом на React та пакуванням Electron. Наведено архітектуру застосунку, реалізацію двадцяти оцінювальних функцій, нечіткого аналізу ієрархій та модуля синтезу мітигаційних стратегій, а також симулятор підключення VR-пристроїв. Виконано експериментальну верифікацію за чотирма операційними сценаріями, проаналізовано вагову матрицю, результати оптимізації мітигації та динаміку просідання й відновлення резильєнтності, а також підтверджено узгодженість результатів десктопної реалізації із Simulink-моделюванням за уніфікованих вхідних даних.

Ключові слова: віртуальна реальність; резильєнтність VR-систем; надійність комп'ютерних систем; динамічне середовище моделювання; компонентні взаємодії; інтегральний показник резильєнтності; безпека VR-середовища; адаптивне тестування; оцінка продуктивності віртуальних систем.

ANNOTATION

Kachur, A. V. Methods and Means for Ensuring the Resilience of Augmented and Virtual Reality Systems. – Qualifying scientific work submitted as a manuscript.

Dissertation submitted for the degree of Doctor of Philosophy in the field of knowledge 12 Information Technologies, speciality 123 Computer Engineering. – Khmelnytskyi National University, Khmelnytskyi, 2026.

Ensuring the resilient and continuous operation of virtual reality systems under rapidly changing environmental conditions constitutes a key scientific challenge in contemporary research.

This dissertation presents a comprehensive analysis of architectural approaches to the design of VR systems, taking into account hardware, software, network and user-related components, as well as methods for ensuring their resilience. A mathematical model has been developed to describe individual functions representing the effects of factors such as ambient temperature, noise level, system load and user behaviour, with weighting coefficients assigned to each factor–component pair. An integral resilience indicator has been proposed that aggregates these functions and represents the overall resilience level of the system. To model dynamic environmental changes and simulate failures, a Simulink model incorporating random parameter fluctuations and sudden changes has been implemented. A series of experiments was subsequently conducted to validate the proposed approach, including a comparative analysis of performance and the accuracy of degradation-state detection. Directions for further research involving the extension of the model and the introduction of adaptive self-recovery mechanisms have also been substantiated.

The object of the research is the process of ensuring the resilience of virtual and augmented reality systems under dynamically changing environmental conditions.

The subject of the research comprises VR-system architectures and methods for assessing the state of a VR system and synthesising mitigation strategies aimed at restoring its operability under dynamically changing environmental conditions.

The aim of the dissertation research is to ensure the resilience of virtual and augmented reality systems through the development of a VR-system architecture and

methods for assessing the state of a VR system and synthesising mitigation strategies for restoring its operability under dynamically changing environmental conditions.

The scientific novelty of the results obtained is as follows.

For the first time, the following have been developed:

1. A method for the synthesis of mitigation strategies which, unlike existing methods, performs multicriteria Pareto-efficient synthesis of mitigation strategies through the simultaneous optimisation of resilience-related objective functions with calibration coefficients, thereby enabling system resilience to be maintained under dynamically changing environmental conditions.

2. A method for assessing the state of a VR system which, unlike existing methods, forms an integral resilience indicator based on an assessment of the effects of dynamically changing environmental factors on system components and employs a system of weighting coefficients obtained through the application of fuzzy logic.

The following has been further developed:

3. A VR-system architecture which, unlike existing architectures, integrates a set of hardware- and software-related components together with methods for state assessment and the synthesis of mitigation strategies into a unified software–hardware system, thereby enabling resilient system operation.

The following has been improved:

4. A model of resilient operation of virtual and augmented reality systems which, unlike existing models, accounts for the effects of external factors, including changes in network conditions, user behaviour, system load, hardware constraints and environmental factors, on the principal system components, including hardware, software, data and users.

Practical significance of the results obtained. The practical value of the results lies in the implementation of the developed theoretical principles in the form of applied methods and a software tool for assessing the state of virtual and augmented reality systems, identifying critical degradation factors and synthesising mitigation strategies.

The results obtained may be used by XR-system developers, system architects, reliability and fault-tolerance engineers, as well as specialists working in research and testing laboratories, during the design, testing, operation and modernisation of virtual and

augmented reality hardware–software systems. The results may also be applied to the development and testing of XR systems used in educational and training environments, medical simulators, industrial visualisation and personnel-training systems, and other applications in which operational continuity, performance stability, acceptable recovery time and user safety are critical.

The dissertation research was conducted within the framework of the research activities of Khmelnytskyi National University, specifically the state-funded research project No. 1B-2026, “System for Ensuring Resilience to Confidential Information Leakage in Corporate Networks under the Influence of Cyberattacks” (state registration No. 0126U002082), in which the author of the dissertation participated as a researcher. The practical results were incorporated into the educational process of the Department of Computer Engineering and Information Systems at Khmelnytskyi National University for practical classes in the courses “System Software” and “Object-Oriented Programming Technologies”. This integrated implementation contributes to the further development of research into the resilience of virtual environments and may be adapted to the requirements of industrial and academic laboratories.

The Introduction substantiates the relevance of ensuring the resilience of virtual and augmented reality systems under dynamic environmental changes. It defines the aim, object, subject and tasks of the research, formulates its scientific novelty and practical significance, and presents the author’s contribution, dissemination activities and publications related to the dissertation.

Chapter 1 reviews contemporary approaches to ensuring the resilience of augmented and virtual reality systems. First, a definition of XR-system resilience and the conditions required to maintain operation and achieve complete recovery are formulated, and the relevance of the research is substantiated: as the use of XR technologies expands in medicine, education, entertainment and industry, the consequences of sudden degradation of immersive environments become increasingly significant. Existing methods are subsequently analysed in seven areas: quantitative resilience assessment and degradation and recovery models; software self-recovery based on checkpointing, software rejuvenation and microreboots; autonomic management and closed-loop self-adaptive architectures;

recovery-oriented design; hardware redundancy and fault tolerance; resilience verification through controlled fault injection; and methods specific to immersive systems. On the basis of this review, research gaps relating to state assessment, the synthesis of mitigation actions and the quantitative verification of recovery are identified, forming the basis for the formulation of the research problem.

Chapter 2 develops a methodological framework for the quantitative assessment of XR-system resilience. The system is represented as a four-component vector comprising hardware, software, data and users. Each component is affected by five categories of external factors: network conditions, the environment, user behaviour, system load and hardware constraints. For each factor–component pair, an individual assessment function is proposed that produces either a Boolean or a normalised numerical result. The resulting set of twenty functions enables key indicators to be monitored in real time. The chapter further describes the construction of a factor–component interaction matrix and introduces an integral resilience indicator expressed as a weighted sum of these functions, with each weight representing the significance of the corresponding influence. This method provides a basis for subsequent optimisation: the resulting metric can be used to rank system vulnerabilities and select priority measures for their mitigation.

Chapter 3 describes an iterative cycle for ensuring the resilience of VR systems, combining state assessment, bottleneck identification, the synthesis of mitigation strategies and system reassessment within a closed loop. The resilience criteria of reliability, availability, fault tolerance, data integrity, recovery time, performance stability and user safety are formalised. On this basis, the problem of multicriteria Pareto-efficient synthesis of mitigation strategies is formulated using a parametric transformation of the mitigation vector and calibration coefficients. A method for synthesising resilient XR architectures is proposed as an inverse mapping of resilience objectives onto design parameters, with the optimisation problem solved through normalised weighted aggregation of the criteria.

Chapter 4 describes the implementation of a software and technical system for the synthesis and assessment of VR-system resilience in the form of a cross-platform desktop application with a TypeScript computational core, a React-based interface and Electron packaging. The application architecture, the implementation of twenty assessment

functions, a fuzzy analytic hierarchy process, a mitigation-strategy synthesis module and a VR-device connection simulator are presented. Experimental verification is conducted using four operational scenarios. The weighting matrix, mitigation optimisation results, and the dynamics of resilience degradation and recovery are analysed. The consistency of the results produced by the desktop implementation with those obtained through Simulink modelling under standardised input conditions is also confirmed.

Keywords: virtual reality; VR-system resilience; computer-system reliability; dynamic simulation environment; component interactions; integral resilience indicator; VR-environment security; adaptive testing; virtual-system performance assessment.

Список публікацій здобувача за темою дисертації

Наукові праці, в яких опубліковані основні наукові результати дисертації

1. Лисенко С., Качур А. Виклики в технологіях віртуальної реальності: оптимізація архітектури систем віртуальної реальності. ВІСНИК, ХНУ. 2024. №1(77). С. 288-297. <https://doi.org/10.31891/2219-9365-2024-77-39>
2. S. Lysenko, A. Kachur. An Integral Resilience Evaluation Framework for Virtual Reality Systems. Центральнoукраїнський науковий вісник. Технічні науки, ЦНТУ. 2025. №12(43). С.80-89 [https://doi.org/10.32515/2664-262X.2025.12\(43\).1.80-89](https://doi.org/10.32515/2664-262X.2025.12(43).1.80-89)
3. Лисенко С., Качур А. Модель відмовостійкого функціонування систем віртуальної реальності в контексті впливу динамічно змінюваного середовища. Measuring and Computing Devices in Technological Processes. 2025. № 3 (83). 499–510. <https://doi.org/10.31891/2219-9365-2025-83-62>
4. Лисенко С., Качур А., Корецька Л. Метод синтезу резильєнтних архітектур систем віртуальної та доповненої реальності. Центральнoукраїнський науковий вісник. Технічні науки. 2026. №13(44). С. 59-72 [https://doi.org/10.32515/2664-262X.2026.13\(44\).59-72](https://doi.org/10.32515/2664-262X.2026.13(44).59-72)

Праці, які засвідчують апробацію матеріалів дисертації

5. Sergii Lysenko, Artem Kachur. Challenges Towards VR Technology: VR Architecture Optimization. Proceedings of 2023 IEEE 13th International Conference on

Dependable Systems, Services and Technologies (DeSSerT-2023, Athens, Greece, October 13-15, 2023) – pp.1-9, DOI:10.1109/DESSERT61349.2023.10416538.

6. Artem Kachur, Sergii Lysenko, Oleh Bodnaruk, Piotr Gaj. Methods of improving security and resilience of VR systems' architecture. CEUR-WS. Vol. 3675. pp. 285-304. URL: <https://ceur-ws.org/Vol-3675/paper21.pdf>.

7. Sergii Lysenko, Artem Kachur. A Resilience Assurance Process Model for Enhancing Virtual Reality Architectural Design. Proceedings of 2024 IEEE 14th International Conference on Dependable Systems, Services and Technologies (DeSSerT-2024, Athens, Greece, October 11-13, 2024).

8. Качур А.В., Лисенко С.М. Виклики в технологiях вiртуальної реальностi. XV Всеукраїнська науково-практична конференцiя “Актуальнi проблеми комп'ютерних наук (АПКН – 2023)”, м. Хмельницький, ХНУ, 17-18 листопада 2023. С. 121-123.

9. Качур А.В., Лисенко С.М. Модель процесу забезпечення резильєнтностi архiтектури систем вiртуальної реальностi. XVI Всеукраїнська науково-практична конференцiя “Актуальнi проблеми комп'ютерних наук (АПКН – 2024)”, м. Хмельницький, ХНУ, 15-16 листопада 2024. С. 340-342.

10. Лисенко С.М., Качур А.В., Модель процесу забезпечення резильєнтностi для оптимiзацiї архiтектури систем вiртуальної реальностi. Проблеми iнформатики та моделювання (ПiМ-2024). Тези двадцять четвертої мiжнародної науково-технiчної конференцiї (20 – 23 вересня 2024 року), с. 79-81.

ЗМІСТ

АНОТАЦІЯ	2
ANNOTATION.....	7
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	16
ВСТУП.....	17
РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ЗАБЕЗПЕЧЕННЯ РЕЗИЛЬЄНТНОСТІ СИСТЕМ ДОПОВНЕНОЇ ТА ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ.	29
1.1 Поняття та критерії резильєнтності систем віртуальної та доповненої реальності	29
1.1.1 Визначення резильєнтності та умови відновлення.....	29
1.1.2 Класифікація впливів та профілювання XR-систем	30
1.2 Актуальність застосування методів та засобів забезпечення резильєнтності систем доповненої та віртуальної реальності.....	37
1.3 Аналіз відомих методів забезпечення резильєнтності VR-систем	39
1.3.1 Методи кількісного оцінювання резильєнтності та моделі деградації і відновлення	39
1.3.2 Методи самовідновлення програмного забезпечення	41
1.3.3 Методи автономного керування та самоадаптивні програмні архітектури	42
1.3.4 Методи проектування, орієнтованого на відновлення	44
1.3.5 Методи синтезу апаратної надлишковості та відмовостійкості у VR- системах.....	45
1.3.6 Методи перевірки резильєнтності контрольованим внесенням відмов	46
1.3.7 Методи забезпечення резильєнтності, специфічні для імерсивних систем	47
1.4 Узагальнене порівняння відомих методів забезпечення резильєнтності XR- систем	49
1.5 Висновки та постановка задачі	51
РОЗДІЛ 2. МОДЕЛЬ ТА МЕТОДИ ЗАБЕЗПЕЧЕННЯ РЕЗИЛЬЄНТНОСТІ СИСТЕМ ВІРТУАЛЬНОЇ ТА ДОПОВНЕНОЇ РЕАЛЬНОСТІ.....	54

2.1 Модель резильєнтного функціонування систем віртуальної та доповненої реальності	55
2.2 Метод оцінювання стану VR-системи в умовах динамічно змінюваного середовища	70
2.2.1 Інтегральний показник резильєнтності системи	70
2.2.2 Визначення вагових коефіцієнтів методом нечіткого аналізу ієрархій	76
2.2.3 Обґрунтування порогових і нормувальних параметрів оцінювальних функцій	84
2.3 Дослідження методу оцінювання стану VR-системи	88
2.3.1 Побудова симуляційної моделі	88
2.3.2 Результати моделювання сценаріїв деградації	90
2.3.3 Аналіз чутливості та стійкості інтегрального показника	96
2.4 Висновки до розділу 2	99
РОЗДІЛ 3. МЕТОДИ СИНТЕЗУ ТА ОЦІНЮВАННЯ РЕЗИЛЬЄНТНОСТІ СИСТЕМ ВІРТУАЛЬНОЇ ТА ДОПОВНЕНОЇ РЕАЛЬНОСТІ	100
3.1 Цикл забезпечення резильєнтності VR-систем	100
3.2 Метод синтезу мітигаційних стратегій для VR-систем	109
3.3 Синтез резильєнтних XR-архітектур	112
3.4 Дослідження методів забезпечення резильєнтності XR-систем	121
3.4.1 Методика та умови дослідження	121
3.4.2 Дослідження методу оцінювання стану XR-системи	123
3.4.3 Дослідження методу синтезу мітигаційних стратегій	126
3.4.4 Дослідження методу синтезу резильєнтних XR-архітектур	128
3.4.5 Дослідження узагальненого методу забезпечення резильєнтності	130
3.4.6 Порівняння запропонованих методів з відомими	132
3.5 Висновки до розділу 3	138

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ ДЛЯ СИНТЕЗУ ТА ОЦІНЮВАННЯ РЕЗИЛЬЄНТНОСТІ СИСТЕМ ВІРТУАЛЬНОЇ ТА ДОПОВНЕНОЇ РЕАЛЬНОСТІ.....	140
4.1 Обґрунтування вибору технологічного стеку	140
4.2 Архітектура програмного забезпечення	141
4.4 Реалізація інтерфейсу користувача	149
4.5 Симуляція підключення VR-пристроїв.....	153
4.6 Пакування та розповсюдження десктопного застосунку.....	154
4.7 Верифікація програмної реалізації.....	155
4.8 Висновки до розділу 4	160
ВИСНОВКИ.....	161
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ.....	163
ДОДАТОК А. СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА	178
ДОДАТОК Б. АКТИ ВПРОВАДЖЕННЯ.....	180
ДОДАТОК В. ВИХІДНІ ДАНІ ТА КОНТРОЛЬНІ РОЗРАХУНКИ	186
В.1 Параметри та сценарії розрахунку.....	186
В.2 Контрольні результати та перевірка ваг.....	189
В.3 Граничні та повторні перевірки	190
ДОДАТОК Г. ПРОГРАМНИЙ КОД ОБЧИСЛЮВАЛЬНОГО ЯДРА.....	193
Г.1 Оцінювальні функції та інтегральний показник.....	193
Г.2 Нечіткий аналіз ієрархій.....	197
Г.3 Параметричний перетворювач і цільові функції	200
Г.4 Програма контрольних перевірок	206

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

VR – віртуальна реальність

AR – доповнена реальність

XR – узагальнене позначення віртуальної, доповненої та змішаної реальності

HMD – шолом віртуальної реальності

TFN – трикутне нечітке число

АНР – метод аналізу ієрархій

FPGA – програмована вентильна матриця

FIR – фільтр із кінцевою імпульсною характеристикою

TMR – трикратне модульне резервування

STMR – вибіркове трикратне модульне резервування

VRPR – резервування зниженої точності на основі більшості голосів

CPU – центральний процесор

ВСТУП

Актуальність роботи. Сучасні системи віртуальної та доповненої реальності дедалі ширше застосовуються в медицині, освіті, індустрії розваг і промисловому виробництві. Розширення сфер їх використання супроводжується ускладненням апаратно-програмної інфраструктури, що підвищує вимоги до надійності, безперервності та стійкості функціонування XR-систем. Під час їх експлуатації необхідно враховувати широкий спектр потенційних проблем, зокрема відмови апаратних компонентів, порушення конфіденційності користувацьких даних, вразливості прошивок і драйверів, а також вплив динамічних змін умов роботи системи. Водночас наявні підходи не забезпечують уніфікованої кількісної оцінки резильєнтності XR-систем з урахуванням сукупного впливу таких чинників, що ускладнює розроблення механізмів адаптивного відновлення, підтримання працездатності та оптимізації продуктивності.

Аналіз існуючих методів і засобів забезпечення резильєнтності обчислювальних та XR-систем показав, що наявні рішення охоплюють окремі аспекти оцінювання деградації, відновлення працездатності, самоадаптації та відмовостійкості, однак не забезпечують їх узгодженого застосування в межах єдиного процесу забезпечення резильєнтності XR-систем. Особливістю таких систем є одночасний вплив динамічно змінюваних факторів на апаратне та програмне забезпечення, дані й користувача, а також жорсткі часові обмеження, пов'язані з необхідністю підтримання неперервності імерсивного сеансу та сенсомоторної узгодженості. Тому методи, розроблені переважно для серверних, мережових та інших обчислювальних систем, не можуть бути безпосередньо застосовані до XR-систем без урахування їхньої багатокомпонентної структури та специфічних умов функціонування.

Існуючі методи кількісного оцінювання резильєнтності дозволяють характеризувати глибину деградації, тривалість перебування системи в деградованому стані та час її відновлення. Проте такі методи переважно оперують уже сформованим агрегованим показником і не визначають способу його отримання

на основі сукупності вимірюваних параметрів XR-середовища. Зокрема, залишається невирішеним питання спільного врахування характеристик мережі, параметрів навколишнього середовища, поведінки користувача, системного навантаження та апаратних обмежень, які мають різну фізичну природу, розмірність та ступінь впливу на окремі компоненти системи. Відсутність єдиної нормованої шкали не дозволяє зіставляти внесок цих факторів у загальну деградацію та однозначно визначати поточний стан XR-системи. Це зумовлює необхідність формалізованого представлення взаємодій між факторами та компонентами системи й формування на їх основі інтегрального показника резильєнтності.

Методи самовідновлення програмного забезпечення, що ґрунтуються на використанні контрольних точок, відкаті стану, профілактичному омолодженні та мікроперезапуску окремих компонентів, забезпечують відновлення після вже виявленого збою. Водночас вони не визначають універсального способу виявлення деградації XR-системи за сукупністю різнорідних параметрів та не встановлюють залежності між причиною погіршення стану і вибором конкретної відновлювальної дії. Для XR-систем ця проблема посилюється необхідністю локалізувати не лише несправний компонент, але й конкретну взаємодію «фактор–компонент», оскільки однаковий зовнішній вплив може мати різні наслідки для апаратного забезпечення, програмних модулів, потоків даних і користувача. Отже, застосування механізмів самовідновлення потребує кількісного оцінювання стану системи та локалізації джерел її деградації.

Методи автономного керування та самоадаптивні архітектури формують замкнений контур моніторингу, аналізу, планування та виконання коригувальних дій і дають змогу змінювати конфігурацію системи без участі оператора. Однак такі підходи переважно визначають загальну структуру процесу адаптації, не надаючи формалізованого методу синтезу конфігурації системи за кількома взаємозалежними та суперечливими критеріями. Зниження графічної складності, перерозподіл обчислювальних ресурсів, зміна режимів енергоспоживання чи мережевих параметрів можуть покращувати один показник резильєнтності та водночас погіршувати інший. Тому вибір мітигаційних дій для XR-систем потребує

одночасного врахування надійності, доступності, відмовостійкості, цілісності даних, часу відновлення, стабільності продуктивності та безпеки користувача. Відомі підходи не забезпечують систематичного синтезу таких стратегій на основі єдиного критеріального простору та пошуку компромісних рішень.

Методи проєктування, орієнтованого на відновлення, зосереджуються насамперед на скороченні часу повернення системи до працездатного стану за рахунок ізоляції відмов, безпечного перезапуску, відкочування змін та обмеження поширення наслідків збою. Такий підхід дозволяє нормувати час відновлення, проте сам по собі не визначає, яким чином має бути змінена конфігурація XR-системи для досягнення необхідного рівня резильєнтності та як оцінити результат застосування декількох мітигаційних дій. Для XR-систем недостатньо встановити факт відновлення окремого компонента: необхідно кількісно підтвердити повернення системи в цілому до допустимого стану та перевірити збереження критичних функцій у встановлених часових межах.

Методи апаратної надлишковості та резервування підвищують стійкість окремих апаратних трактів за рахунок дублювання компонентів, резервних каналів, механізмів виправлення помилок і балансування навантаження. Водночас вони орієнтовані переважно на апаратний рівень і не враховують у єдиній моделі програмні, інформаційні та поведінкові складові XR-системи. Крім того, використання резервування пов'язане з додатковими витратами обчислювальних, енергетичних та апаратних ресурсів, тому саме по собі збільшення надлишковості не гарантує оптимального стану системи. Необхідним є узгодження таких засобів з іншими мітигаційними діями залежно від поточного стану системи та характеру виявленої деградації.

Методи контрольованого внесення відмов та інженерії хаосу дозволяють експериментально перевіряти працездатність механізмів відновлення й виявляти приховані залежності між компонентами. Однак результати таких експериментів часто характеризують реакцію на окремий тип відмови або зміну окремого показника. Для оцінювання резильєнтності XR-систем необхідно не лише встановити факт успішного відновлення, а й кількісно оцінити глибину деградації, динаміку

повернення системи до працездатного стану та внесок застосованих мітигаційних дій у відновлення. Без спільної кількісної метрики результати різних сценаріїв внесення відмов складно зіставляти між собою та використовувати для обґрунтованого вибору конфігурації системи.

Специфічні для імерсивних систем механізми, зокрема компенсація пропущених кадрів, фовеальний рендеринг, динамічна зміна рівня деталізації та кероване зниження графічного навантаження, дозволяють підтримувати окремі критичні функції XR-системи за короткочасної деградації. Проте такі засоби мають локальний характер і, як правило, спрямовані на конкретний компонент або окремий показник якості. Вони не забезпечують узгодженого вибору дій з урахуванням стану всієї системи та можливих наслідків мітигації для інших компонентів і користувача.

Таким чином, існуючі методи та засоби забезпечення резильєнтності розв'язують окремі складові задачі, проте їх просте поєднання не формує комплексного рішення для XR-систем через відсутність спільної моделі представлення стану, єдиної нормованої метрики та узгодженого критерію вибору відновлювальних дій. Невирішеними залишаються задачі кількісного оцінювання стану XR-системи з урахуванням взаємодії різнорідних факторів і компонентів, синтезу мітигаційних стратегій за сукупністю критеріїв резильєнтності, кількісного підтвердження результатів відновлення та інтеграції зазначених процесів у єдину архітектуру. Це визначає необхідність розроблення архітектури VR-системи, методів оцінювання її стану та синтезу мітигаційних стратегій для забезпечення й відновлення працездатності в умовах динамічно змінюваного зовнішнього середовища. Отже, актуальною є задача забезпечення резильєнтності систем віртуальної та доповненої реальності шляхом розроблення архітектури VR-системи та методів оцінювання стану VR-системи й синтезу мітигаційних стратегій для відновлення працездатності VR-системи в умовах динамічно змінюваного зовнішнього середовища.

Значний внесок у розвиток моделей, методів та засобів забезпечення резильєнтності, надійності, відмовостійкості та безпеки комп'ютерних і XR-систем зробили, В. С. Харченко [75, 121–123], О. С. Савенко [75, 121], К. Ю. Бобровнікова

[75, 121], О. О. Ілляшенко [122, 123], Г. В. Фесенко [122, 123], І. В. Бабешко [122, 123], О. В. Боднарчук [84], О. Павлова, А. Башта, М. Ковтонюк [29], А. Саченко та ін. [78].

Серед іноземних дослідників вагомий внесок у розвиток теорії резильєнтності, методів кількісного оцінювання деградації та відновлення, самоадаптивних систем, відмовостійких архітектур і технологій забезпечення резильєнтності XR-систем зробили I. Linkov [7, 8], A. Kott [8], R. Ross [9], C. Poulin, M. B. Kane [10], M. S. Elbamby, C. Perfecto, M. Bennis [11], P. Kumari, P. Kaur [13], D. Weyns [16, 105, 106, 133], F. Hu, Y. Deng, W. Saad [19], C. Rosenthal, N. Jones [26], A. Basiri [27], J. Chakareski [120], J. Cámara та D. Garlan [15, 134] та ін.

Незважаючи на значну кількість наукових досліджень у сфері забезпечення резильєнтності комп'ютерних та XR-систем, наявні результати не забезпечують комплексного розв'язання задачі їх функціонування в умовах динамічно змінюваного середовища. Зокрема, залишаються невирішеними питання зведення різномірних параметрів стану апаратного і програмного забезпечення, даних та користувацького контуру до єдиного кількісного показника резильєнтності; локалізації критичних взаємодій між факторами впливу та компонентами системи; синтезу мітигаційних стратегій з одночасним урахуванням декількох суперечливих критеріїв; кількісної перевірки завершеності відновлення; а також інтеграції процесів оцінювання, адаптації та відновлення в єдину архітектуру XR-системи.

Таким чином, існує суперечність між потребою в забезпеченні резильєнтного функціонування систем віртуальної та доповненої реальності в умовах динамічно змінюваного зовнішнього середовища, з одного боку, та недостатністю існуючих методів і засобів комплексного оцінювання стану таких систем, синтезу мітигаційних стратегій і кількісного підтвердження відновлення їх працездатності, з іншого боку. Відтак, забезпечення резильєнтності систем віртуальної та доповненої реальності є актуальною науково-прикладною задачею, одним із шляхів розв'язання якої є розроблення архітектури XR-систем, методів оцінювання їх стану та синтезу мітигаційних стратегій для відновлення працездатності в умовах динамічно змінюваного зовнішнього середовища.

Зазначена науково-прикладна задача відповідає предметній області Стандарту вищої освіти України зі спеціальності 123 Комп'ютерна інженерія для третього (освітньо-наукового) рівня вищої освіти, зокрема, таким об'єктам вивчення та діяльності: програмно-технічні засоби (апаратні, програмовні, реконфігуровні, системне та прикладне програмне забезпечення) комп'ютерів та комп'ютерних систем універсального та спеціального призначення, в тому числі стаціонарних, мобільних, вбудованих, розподілених тощо, ... інтерфейси та протоколи взаємодії їх компонентів; інформаційні процеси, технології, методи, способи та системи автоматизованого та автоматичного проектування; налагодження, ... й експлуатації, ... засоби підтримки керування життєвим циклом вказаних програмно-технічних засобів; методи та способи опрацювання інформації, математичні моделі обчислювальних процесів, технології виконання обчислень, ... безпечних, автономних, адаптивних, інтелектуальних, розумних тощо, архітектура та організація функціонування відповідних програмно-технічних засобів.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційне дослідження виконувалось у рамках науково-дослідної тематики Хмельницького національного університету: держбюджетної науково-дослідної теми №1Б-2026 «Система забезпечення резильєнтності до витоку конфіденційної інформації в корпоративних мережах в умовах впливів комп'ютерних атак» (номер держреєстрації 0126U002082), в якій автор дисертації був виконавцем.

Мета і завдання дослідження.

Об'єктом дослідження є процес забезпечення резильєнтності систем віртуальної та доповненої реальності в умовах динамічно змінюваного зовнішнього середовища.

Предметом дослідження є архітектури VR-системи та методи оцінювання стану VR-системи й синтезу мітигаційних стратегій для відновлення працездатності VR-системи в умовах динамічно змінюваного зовнішнього середовища.

Метою дисертаційного дослідження є забезпечення резильєнтності систем віртуальної та доповненої реальності шляхом розроблення архітектури VR-системи та методів оцінювання стану VR-системи й синтезу мітигаційних стратегій для

відновлення працездатності VR-системи в умовах динамічно змінюваного зовнішнього середовища.

Для досягнення зазначеної мети визначено такі завдання дослідження:

1. Провести аналіз існуючих архітектур XR-систем та методів до оцінки їхньої резильєнтності, виділити ключові фактори впливу та функціональні компоненти системи.
2. Розробити модель резильєнтного функціонування систем віртуальної та доповненої реальності.
3. Розробити метод оцінювання стану VR-системи в умовах динамічно змінюваного середовища.
4. Розробити метод синтезу мітигаційних стратегій для VR-систем.
5. Розробити архітектуру резильєнтних VR-систем, яка інтегрує в собі комплекс компонентів, методи оцінювання, синтезу мітигаційних стратегій в єдиний програмно-апаратний засіб.
6. Реалізувати комплексний програмний засіб для автоматизованої оцінки та синтезу резильєнтності VR-системи та провести експериментальну верифікацію.
7. Апробувати запропоновані методи у прикладних VR-сценаріях та оцінити їхній вплив на підвищення загальної резильєнтності системи.

Методи дослідження. Методи дослідження, застосовані в роботі, включають:

1. Абстрактно-алгебраїчний метод для формалізації архітектури VR-системи й опису взаємодій її компонентів у вигляді множин та операцій над ними. Це дозволило однозначно визначити структуру системи та взаємозв'язки «фактор—компонент».
2. Теорія множин і матрична алгебра для побудови матриці взаємодій зовнішніх і внутрішніх факторів із компонентами системи та визначення інтегрального показника резильєнтності як зваженої суми елементів цієї матриці.
3. Нечітко-логічне моделювання для призначення вагових коефіцієнтів індивідуальним функціям впливу, що дає змогу інкорпорувати експертні оцінки й нечіткі знання про важливість кожного «фактор—компонент».

4. Теорія алгоритмів та методи оптимізації, зокрема розробка адаптивного алгоритму динамічного розподілу обчислювальних ресурсів, що забезпечує балансування навантаження між гетерогенними вузлами в реальному часі.

5. Моделювання дискретних подій і стохастичне моделювання в середовищі Simulink з використанням випадкових флуктуацій параметрів і раптових стрибків для імітації змін зовнішнього середовища та відмовостійкості системи.

6. Експериментальні та статистичні методи аналізу результатів, включно з порівняльним дослідженням продуктивності та точності виявлення деградаційних станів, а також оцінкою впливу запропонованих заходів на підвищення резильєнтності системи.

7. Методика розробки та апробації заходів мінімізації негативного впливу (адаптивне згладжування навантаження, динамічна корекція параметрів моделювання), що дозволила оцінити їхню ефективність у підвищенні загальної продуктивності та стабільності VR-систем.

Наукова новизна отриманих результатів полягає в наступному:

вперше розроблено:

1) метод синтезу мітигаційних стратегій, який, на відміну від відомих методів здійснює мультикритеріальний Парето-ефективний синтез мітигаційних стратегій шляхом одночасної оптимізації цільових функцій параметрів резильєнтності з калібрувальними коефіцієнтами, і який дозволяє забезпечувати резильєнтність системи в умовах динамічно змінюваного зовнішнього середовища;

2) метод оцінювання стану VR-системи, який, на відміну від відомих методів формує інтегральний показник резильєнтності, який ґрунтується на оцінці впливу факторів динамічно змінюваного середовища на компоненти системи і використовує систему вагових коефіцієнтів, отриману шляхом застосування нечіткої логіки;

набула подальшого розвитку:

3) архітектура VR-систем, яка на відміну від відомих архітектур інтегрує в собі комплекс компонентів, пов'язаних із програмно-апаратними засобами, методи

оцінювання, синтезу мітигаційних стратегій в єдиний програмно-апаратний засіб, що дозволяє здійснювати резильєнтне функціонування систем;

удосконалено:

4) модель резильєнтного функціонування систем віртуальної та доповненої реальності, у якій на відміну від відомих моделей враховано вплив зовнішніх факторів, таких як зміна стану мережі, поведінка користувачів, навантаження на систему, апаратні обмеження та фактори навколишнього середовища на основні компоненти системи, включаючи апаратне та програмне забезпечення, дані та користувачів.

Практичне значення отриманих результатів. Практична цінність отриманих результатів полягає у доведенні розроблених теоретичних положень до рівня прикладних методів і програмного засобу для оцінювання стану систем віртуальної та доповненої реальності, виявлення критичних факторів деградації та синтезу мітигаційних стратегій. Отримані результати можуть бути використані розробниками XR-систем, системними архітекторами, інженерами з надійності та відмовостійкості, а також фахівцями дослідницьких і випробувальних лабораторій під час проєктування, тестування, експлуатації та модернізації програмно-апаратних комплексів віртуальної та доповненої реальності. Результати роботи можуть бути використані під час розроблення та випробування XR-систем для освітніх і тренувальних комплексів, медичних симуляторів, промислових систем візуалізації та підготовки персоналу, а також інших застосувань, у яких критичними є безперервність роботи, стабільність продуктивності, допустимий час відновлення та безпека користувача. Дисертаційне дослідження виконувалось у рамках науково-дослідної тематики Хмельницького національного університету: держбюджетної науково-дослідної теми №1Б-2026 «Система забезпечення резильєнтності до витоку конфіденційної інформації в корпоративних мережах в умовах впливів комп'ютерних атак» (номер держреєстрації 0126U002082), в якій автор дисертації був виконавцем.

Особистий внесок здобувача. Всі основні результати дисертаційного дослідження, які представлені до захисту, отримані автором особисто.

У наукових працях, опублікованих у співавторстві, здобувачеві належать: у [80] – формулювання основних викликів оптимізації архітектури VR-систем та розроблення концептуальної моделі оцінювання їх функціонування; С. М. Лисенку – наукове консультування, критичний перегляд та редакційне опрацювання матеріалів статті; у [81] – розроблення математичної моделі резильєнтного функціонування VR-системи з урахуванням впливу динамічно змінюваного середовища; С. М. Лисенку – наукове консультування, обговорення отриманих результатів та редагування рукопису; у [82] – розроблення інтегрального показника резильєнтності та способу агрегування індивідуальних функцій впливу з використанням вагових коефіцієнтів; С. М. Лисенку – методичне консультування, критичний аналіз одержаних результатів та редакційне опрацювання публікації; у [83] – формування підходу до оптимізації архітектурних компонентів VR-систем в умовах ресурсних обмежень та проведення відповідних досліджень; С. М. Лисенку – наукове консультування, організаційний супровід дослідження та критичний перегляд результатів; у [84] – розроблення та апробація методів підвищення безпеки й резильєнтності архітектури VR-систем у розподіленому середовищі; С. М. Лисенку – наукове консультування та критичний аналіз результатів, О. В. Боднаруку – участь в аналізі предметної області та обговоренні результатів, Р. Гаї – критичний перегляд матеріалів і редакційно-організаційний супровід підготовки публікації; у [85] – розроблення процесної моделі забезпечення резильєнтності та механізмів адаптивного розподілу ресурсів; С. М. Лисенку – методичне консультування, аналіз та обговорення одержаних результатів, редакційне опрацювання рукопису; у [86] – виокремлення ключових напрямів дослідження викликів VR-технологій та проведення їх аналітичного огляду; С. М. Лисенку – наукове консультування, систематизація висновків та редагування матеріалів; у [87] – визначення структурних зв'язків між компонентами системи та їх ролі у забезпеченні резильєнтності; С. М. Лисенку – обговорення результатів, методичне консультування та редакційне опрацювання матеріалів; у [88] – розроблення та дослідження параметрів оцінювання резильєнтності з метою підвищення точності та обчислювальної ефективності моделі; С. М. Лисенку –

критичний аналіз результатів, наукове консультування та організаційно-редакційний супровід підготовки публікації.

Апробація результатів дисертації. Апробацію основних положень, наукових результатів та висновків дисертаційної роботи проведено на науковому семінарі кафедри комп'ютерної інженерії та інформаційних систем Хмельницького національного університету. Наукові результати роботи доповідались та обговорювались на таких міжнародних і всеукраїнських наукових заходах: 2023 IEEE 13th International Conference on Dependable Systems, Services and Technologies (DeSSerT-2023) (Афіни, Греція, 13–15 жовтня 2023 р.); XV Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук» (АПКН-2023) (Хмельницький, 17–18 листопада 2023 р.); 5th International Workshop on Intelligent Information Technologies and Systems of Information Security (IntelITSIS'2024) (Хмельницький, 28 березня 2024 р.); XXIV Міжнародній науково-технічній конференції «Проблеми інформатики та моделювання» (ПІМ-2024) (20–23 вересня 2024 р.); 2024 IEEE 14th International Conference on Dependable Systems, Services and Technologies (DeSSerT-2024) (Афіни, Греція, 11–13 жовтня 2024 р.); XVI Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук» (АПКН-2024) (Хмельницький, 15–16 листопада 2024 р.); XVII Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук» (АПКН-2025) (Хмельницький, 14–15 листопада 2025 р.); 2025 IEEE 15th International Conference on Dependable Systems, Services and Technologies (DeSSerT-2025) (Афіни, Греція, 19–21 грудня 2025 р.); The 2nd International Workshop on Advanced Applied Information Technologies: AI & DSS (AdvAIT-2025) (Хмельницький, Україна, 5 грудня 2025 р.); The 16th International Conference on Advanced Computer Information Technologies (ACIT'2026) (Злін, Чехія, 2-4 вересня 2026 р.)

Публікації. За результатами проведених досліджень основні наукові результати опубліковано у 4 наукових статтях в трьох фахових наукових журналах України. Апробація засвідчена публікаціями 6 праць в матеріалах міжнародних та всеукраїнських конференцій, з яких 3 праці проіндексовано у наукометричній базі Scopus.

Структура та обсяг дисертації. Дисертація складається з анотації, змісту, переліку умовних скорочень, вступу, чотирьох розділів, висновків, списку використаних джерел із 134 найменувань на 15 сторінках та 4 додатків на 30 сторінках. Загальний обсяг дисертаційної роботи становить 208 сторінок друкованого тексту, з них 143 сторінок основного тексту. Дисертація містить 15 рисунків та 37 таблиць.

РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ЗАБЕЗПЕЧЕННЯ РЕЗИЛЬЄНТНОСТІ СИСТЕМ ДОПОВНЕНОЇ ТА ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ

1.1 Поняття та критерії резильєнтності систем віртуальної та доповненої реальності

1.1.1 Визначення резильєнтності та умови відновлення

У дослідженні резильєнтність XR-системи визначається як здатність зберігати критичні функції під дією зовнішніх і внутрішніх впливів, переходити у контрольований деградований режим без загрози користувачеві та відновлювати заданий рівень працездатності. Система, що витримує навантаження, але не відновлює функціональні показники у встановлений час, не вважається резильєнтною.

Критичною функцією вважається функціональна можливість системи, деградація якої унеможлиблює виконання цільового сценарію, порушує сенсомоторну узгодженість або створює загрозу безпеці. Оцінювальні функції розділу 2 є математичними залежностями й до цього поняття не належать. Поділ функцій на критичні та некритичні визначається профілем місії конкретного застосування та фіксується до оцінювання резильєнтності, що узгоджується з моделлю якості програмних систем [89] і принципами інженерії кіберрезильєнтності [9].

До критичних функцій XR-систем віднесено формування стереозображення з дотриманням допустимої затримки та частоти кадрів, оскільки перевищення затримки пов'язане з виникненням кінетозу [90, 91], а відповідні вимоги для імерсивних систем визначено у [130]. Критичними також є відстеження положення й орієнтації шолома та контролерів, втрата якого може спричинити дезорієнтацію і травмування [92]; просторова та часова узгодженість сенсорних каналів, порушення якої спричиняє сенсорний конфлікт [93]; підтримання безпечної зони переміщення

[94]; забезпечення цілісності, автентичності та конфіденційності трекінгових і біометричних даних [95, 96] відповідно до загальних вимог [127]; а також підтримання безпечного теплового режиму та рівня живлення обладнання [97]. До некритичних віднесено функціональні можливості, що визначають якість подання, але не його припустимість: деталізацію геометрії та текстур, складність моделі освітлення, дальність промальовування, згладжування, обсяг фонових кешування та періодичність необов'язкової телеметрії. Саме вони утворюють резерв деградації, за рахунок якого мітигаційні стратегії, розглянуті у підрозділі 3.2, вивільняють ресурси для збереження критичних функцій. Практичним підтвердженням дієвості такого резерву є механізми керованого зниження якості подання, зокрема фовеальний рендеринг і динамічна зміна рівня деталізації, кількісні оцінки вивільнюваного ресурсу для яких наведено у [98, 99].

Уведений поділ дає змогу відрізнити контрольований деградований режим від відмови. Якщо зниження загальної якості сеансу досягнуто виключно за рахунок некритичних функціональних можливостей, а рівні виконання всіх критичних функцій залишаються не нижчими за власні допуски, система перебуває у контрольованому деградованому режимі, який означенням резильєнтності визнається припустимим. Якщо ж хоча б одна критична функція втрачає виконуваність, подія класифікується як відмова незалежно від значення інтегрального показника. Ця умова використовується далі як обов'язкове обмеження задачі синтезу мітигаційних стратегій у підрозділі 3.2, оскільки стратегія, що підвищує інтегральний показник за рахунок втрати критичної функції, є недопустимою.

Класи критичних функцій XR-системи, показники їх виконання та умови невиконаності узагальнено у таблиці 1.1.

1.1.2 Класифікація впливів та профілювання XR-систем

Умови збереження функціонування та завершеного відновлення, сформульовані вище, визначаються як властивостями системи, так і характером

впливу та вимогами застосування. Тому для подальшого аналізу відомих методів необхідно впорядкувати впливи за походженням, часовим характером і спостережуваністю, а вимоги застосувань – за профілями місії. Потреба в такому впорядкуванні зумовлена швидким розширенням переліку джерел впливу у сучасних імерсивних середовищах, що показано в узагальнювальних оглядах галузі [125].

Таблиця 1.1 – Критичні функції XR-системи та показники їх виконання

Клас критичної функції	Показник виконання	Умова невиконуваності	Наслідок для користувача
Формування та подання зображення	Затримка «рух – фотон», мс; частота кадрів, Гц	Перевищення бюджету затримки або зниження частоти кадрів нижче профільного мінімуму	Кінетоз, розрив занурення
Відстеження положення й орієнтації (6DoF)	Похибка визначення положення, мм; частота оновлення пози, Гц	Перевищення допуску похибки або втрата трекінгу	Дезорієнтація, ризик механічної травми
Узгодженість сенсорних каналів	Розсинхронізація візуального, звукового та інерційного каналів, мс	Перевищення порогу розсинхронізації, за якого виникає сенсорний конфлікт	Вестибулярний конфлікт, запаморочення
Підтримання меж безпечної зони	Затримка попередження, мс; частка виявлених виходів за межі зони	Пропуск події виходу за межі або запізнення попередження	Фізична травма при контакті з оточенням

Закінчення таблиці 1.1

Цілісність і захищеність даних сеансу	Частка неспотворених кадрів даних; результат перевірки автентичності	Будь-яке непідтверджене порушення цілісності або автентичності	Компрометація трекінгових і біометричних даних
Цілісність і захищеність даних сеансу	Частка неспотворених кадрів даних; результат перевірки автентичності	Будь-яке непідтверджене порушення цілісності або автентичності	Компрометація трекінгових і біометричних даних
Тепловий режим і живлення	Температура поверхні контакту, °C; резерв заряду, %	Перевищення допустимої температури контакту або вичерпання резерву заряду	Термічний дискомфорт, раптове завершення сеансу

За походженням впливи на XR-системи поділяються на такі класи:

- середовищні впливи: зміни температури та вологості навколишнього середовища, рівня освітленості робочої зони, акустичного шуму та інших фізичних параметрів середовища. Вони можуть впливати на роботу оптичних та інерційних сенсорів, тепловий режим обладнання та комфорт користувача;
- мережеві впливи: зниження пропускну здатності, збільшення затримки та джитера, втрата пакетів, нестабільність або розрив з'єднання. Для XR-систем із розподіленою обробкою даних та віддаленим формуванням контенту такі впливи безпосередньо позначаються на своєчасності передавання даних і стабільності імерсивного сеансу;

- обчислювальні впливи: перевантаження центрального та графічного процесорів, конкуренція за ресурси оперативної пам'яті, пропускну здатність пам'яті та засоби зберігання даних. Наслідками можуть бути зниження частоти кадрів, збільшення часу обробки даних та загальне погіршення продуктивності системи;
- апаратні впливи: перегрів компонентів, тепловий дроселінг, зниження рівня заряду або доступної потужності, дрейф характеристик сенсорів, а також часткові або повні відмови апаратних модулів. Такі впливи можуть призводити як до поступового погіршення характеристик, так і до раптової втрати окремих функцій системи;
- поведінкові впливи: інтенсивність і різкість рухів користувача, тривалість сеансу, зміна патернів взаємодії та відхилення від передбаченого сценарію використання. Вони можуть змінювати навантаження на систему, ускладнювати оброблення сенсорних даних та впливати на безпеку й комфорт користувача;
- кібернетичні впливи – несанкціонований доступ до потоків даних, їх спотворення, порушення конфіденційності, атаки типу «відмова в обслуговуванні» та інші цілеспрямовані втручання. У межах моделі такі впливи доцільно враховувати через їх вимірювані наслідки для мережі, програмного забезпечення, даних і користувацького контуру.

За часовим характером впливи поділяються на:

- раптові – стрибкоподібні зміни параметрів або миттєві відмови, які проявляються протягом короткого інтервалу часу;
- поступові – повільна деградація характеристик, наприклад накопичення тепла, зниження ресурсу акумулятора або зростання фрагментації пам'яті;
- періодичні – циклічні зміни навантаження чи параметрів середовища, що повторюються з певною закономірністю;
- стохастичні – випадкові коливання параметрів і шумова складова вимірювань, які не мають наперед визначеної часової структури.

За спостережуваністю впливи поділяються на:

- безпосередньо вимірювані – такі, що фіксуються штатними засобами телеметрії, зокрема параметри мережі, температури, рівень завантаження обчислювальних ресурсів, стан живлення та інші технічні показники;
- опосередковано спостережувані – такі, що не вимірюються безпосередньо, але можуть бути визначені за змінами інших параметрів системи або за їх наслідками. До цього класу можуть належати окремі поведінкові та кібернетичні впливи.

Запропонована класифікація дозволяє описувати кожний вплив не лише за його джерелом, а й за характером розвитку в часі та можливістю його виявлення. Це створює основу для формування набору оцінювальних параметрів, локалізації джерел деградації та вибору відповідних мітигаційних дій. Профіль місії задає допустимі межі деградації та граничний час відновлення для конкретного застосування. Одна й та сама технічна деградація має різну ціну: втрата точності трекінгу на кілька міліметрів є неприпустимою в задачах хірургічного планування і несуттєвою в розважальному застосуванні, тоді як короткочасне зниження частоти кадрів у тренажері оператора спричиняє помилку дії, а у статичному демонстраційному сценарії залишається непоміченим. Відмінність вимог за галузями застосування підтверджують огляди імерсивних систем у медичній освіті [102], хірургічній підготовці [103], вищій освіті [104] та промислових застосуваннях [69]. Практика нормування цільових рівнів і граничного часу відновлення для критично важливих процесів унормована у сфері управління безперервністю діяльності [126], а узгодження цих величин із загальним управлінням ризиками організації – у [108]. Формально профіль місії подано четвіркою, що містить мінімально допустимий рівень виконання критичних функцій, необхідний рівень відновлення, граничний час відновлення та вектор коефіцієнтів значущості критичних функцій із формули (1.1):

$$P = \langle q_{\min}, q_{\text{req}}, T_{\text{rec}}, \gamma \rangle \quad (1.1)$$

Типові профілі місії XR-систем та відповідні їм межі деградації наведено у таблиці 1.2. Значення подано як орієнтовні межі, що використовуються далі для нормування параметрів оцінювальних функцій і для інтерпретації результатів

мітигації; у прикладному проєкті вони уточнюються за вимогами замовника та документацією обладнання.

Наведена класифікація визначає вимоги до методів забезпечення резильєнтності. Метод оцінювання стану має приймати різномірні за фізичною природою параметри всіх шести груп впливів і зводити їх до єдиної шкали. Метод виявлення деградації має одночасно опрацьовувати раптові та поступові впливи. Метод синтезу мітигаційних дій має враховувати профіль місії, оскільки саме він визначає, які функції належать до критичних і який час відновлення є припустимим. Відповідність відомих методів цим вимогам розглянуто у підрозділі 1.3, а узагальнене зіставлення виконано у підрозділі 1.4.

Таблиця 1.2 – Профілі місії XR-систем та допустимі межі деградації

Профіль місії	q_{min}	q_{req}	T_{rec}	Домінантні критичні функції
Медичний: діагностика, планування втручань	0,90	0,98	до 1 с	Цілісність даних, точність відстеження положення
Промисловий: дистанційне керування, обслуговування	0,85	0,95	до 2 с	Затримка подання, межі безпечної зони
Тренувальний та освітній	0,75	0,92	до 3 с	Узгодженість сенсорних каналів, стабільність частоти кадрів
Розважальний	0,60	0,85	до 5 с	Стабільність частоти кадрів

Нехай $q(t)$ є нормованим показником виконання критичних функцій, q_{min} є мінімально допустимим рівнем під час впливу, q_{req} є рівнем, який має бути відновлено, а T_{rec} є граничним часом відновлення. Якщо вплив діє на інтервалі $[t_0; t_1]$, система зберігає функціонування за умови $q(t) \geq q_{min}$ для всіх t цього інтервалу. Відновлення вважається завершеним, якщо існує момент $t_r \leq t_1 + T_{rec}$, після якого $q(t)$

$\geq q_{req}$. Ці дві умови використовуються далі для інтерпретації інтегрального показника, часу відновлення та результатів мітигації.

Для однозначності викладу надалі використано такі позначення. VR означає систему повного занурення користувача у синтезоване середовище. AR означає систему, що поєднує сприйняття фізичного середовища з цифровими об'єктами. XR є узагальненим позначенням імерсивних систем, до якого належать VR, AR та змішана реальність. Оскільки віртуальна реальність є підмножиною імерсивних систем, для положень, справедливих для всього класу імерсивних систем, використовується позначення XR, а позначення VR і AR застосовуються лише тоді, коли твердження стосується конкретної технології. Комбіноване позначення «VR/XR» не використовується як надлишкове. Для конкретної технології застосовано окреме позначення VR або AR. У всьому авторському тексті вжито термін «доповнена реальність».

XR-системи належать до спеціалізованих кіберфізичних систем реального часу з людиною в контурі керування. Їх визначальними рисами є жорсткі обмеження затримки, безперервний сенсорний зворотний зв'язок, залежність якості роботи від фізичного стану користувача та прямий вплив програмно-апаратних збоїв на сприйняття і безпечність взаємодії. У медичних, навчальних, промислових і тренувальних застосуваннях допустимі пороги деградації різняться, тому вони задаються профілем місії та враховуються під час нормування параметрів.

Безпека розглядається як одна зі складових резильєнтності XR-системи. Вона охоплює захист інформації, протидію несанкціонованим діям, цілісність даних та безпечність взаємодії користувача з віртуальним середовищем. Резильєнтність має ширший функціональний зміст у межах цієї роботи: система повинна витримати вплив, зберегти критичні функції, обмежити наслідки та відновити працездатність. Кіберзагрози враховуються через їх спостережувані наслідки для мережі, програмного забезпечення, даних і користувача.

Мітигаційна стратегія є набором апаратних, програмних та алгоритмічних дій, спрямованих на зменшення ймовірності збою, обмеження його наслідків і відновлення працездатності. У XR-системі мітигація має запобігти повній зупинці

критичних функцій, зберегти узгодженість сенсорного зворотного зв'язку та повернути показники системи до допустимого рівня.

У контексті VR-систем резильєнтність апаратного забезпечення включає здатність апаратних компонентів, таких як процесор, дисплей, сенсори, батарея, функціонувати при відхиленнях від номінальних умов: підвищеній температурі, надмірному навантаженні, зниженому рівні заряду батареї.

Кількісна оцінка резильєнтності вимагає побудови інтегрального показника, який агрегує стан усіх компонентів системи з урахуванням вагової важливості факторів впливу. Така задача повинна бути вирішена через розробку відповідних методів та архітектури.

1.2 Актуальність застосування методів та засобів забезпечення резильєнтності систем доповненої та віртуальної реальності

Поширення XR-систем збільшує значущість їх стійкості до мережових коливань, теплових і обчислювальних перевантажень, відмов компонентів та кібернетичних втручань. Імерсивна взаємодія потребує одночасного збереження технічної працездатності й безпеки користувача, тому оцінювання стану має бути пов'язане із засобами відновлення та перевіркою їх результативності.

У XR-сеансі користувач перебуває всередині контуру керування, а переривання зображення або розсинхронізація сенсорних каналів порушують взаємодію. Самовідновлення має охоплювати виявлення деградації за телеметрією, локалізацію джерела та застосування компенсувальних дій у межах часу, заданого профілем місії.

Окремим виміром актуальності є вплив збоїв XR-систем на фізичний стан користувача. Раптове падіння частоти кадрів, зростання затримки відтворення руху, розсинхронізація візуального та вестибулярного каналів або втрата позиційного відстеження провокують кінетоз, запаморочення, порушення рівноваги та зорову втоми; у навчальних і медичних застосуваннях такі стани неприпустимі. Експериментальні дослідження продуктивності й самопочуття користувачів у

імерсивних середовищах підтверджують прямий зв'язок між нестабільністю роботи системи та симптомами кінетозу [12]. Тому резильєнтність XR-системи слід розглядати одночасно як вимогу технічної надійності та як умову безпечності для людини, що додатково обґрунтовує введення до складу критеріїв оцінювання показника безпеки користувача.

Перелічені аспекти узагальнено у таблиці 1.3.

Таблиця 1.3 – Аспекти забезпечення резильєнтності систем доповненої та віртуальної реальності

Аспекти	Методи
Кількісне оцінювання резильєнтності	Інтегральні показники стану, криві деградації та відновлення, порогові умови повернення працездатності
Самовідновлення програмного забезпечення	Контрольні точки та відкат, профілактичний перезапуск, мікроперезапуск ізольованих модулів
Автономне керування та самоадаптація	Контур моніторингу, аналізу, планування та виконання, архітектурні моделі самоадаптації, політики реконфігурації
Проектування, орієнтоване на відновлення	Ізоляція відмов, безпечний перезапуск, скасування помилкових дій, скорочення часу відновлення
Апаратна надлишковість та відмовостійкість	Дизайн подвійної системи, техніки виявлення та виправлення помилок, балансування навантаження
Перевірка резильєнтності	Контрольоване внесення відмов, інженерія хаосу, репетиції відновлення у наближених до експлуатації умовах

1.3 Аналіз відомих методів забезпечення резильєнтності VR-систем

1.3.1 Методи кількісного оцінювання резильєнтності та моделі деградації і відновлення

Сучасні рамки резильєнтності розглядають здатність системи передбачати впливи, витримувати їх, відновлюватися та адаптуватися як єдину властивість, спільну для технічних і організаційних систем [7]. Для обчислювальних систем і мереж цю властивість деталізовано у [8]: резильєнтність поєднує запобігання, поглинання впливу, відновлення працездатності та подальше пристосування, а кібернетичні втручання розглядаються нарівні з відмовами обладнання і помилками експлуатації. Для XR-систем таке трактування є принциповим, оскільки середовище експлуатації змінюється безперервно і не може бути повністю передбачене на етапі проектування. Прикладні дослідження резильєнтності обчислювальних систем за наявності кіберзагроз, зокрема таксономія та онтологія резильєнтності [75], самоадаптивні методи її забезпечення [76], виявлення багатовекторних кібератак засобами машинного навчання [121] та безпеко-орієнтований аналіз безпечності автономних систем [122, 123], підтверджують застосовність цього трактування до широкого класу технічних систем.

Інженерне забезпечення кіберрезильєнтності обчислювальних систем унормовано у [9]. Документ визначає чотири цілі, а саме передбачити, витримати, відновитися та адаптуватися, і техніки їх досягнення, зокрема аналітичний моніторинг, сегментацію, різноманітність і надлишковість, та пов'язує кожну техніку з етапами життєвого циклу системи. Це дає змогу закладати вимоги резильєнтності у проектування VR-системи, а не додавати їх після розгортання.

Кількісне оцінювання резильєнтності будується на аналізі траєкторії нормованого показника виконання критичних функцій у часі. Типова траєкторія містить фази номінального функціонування, просідання під дією впливу, стабілізації у деградованому режимі та відновлення (рис. 1.1). Систематизацію мір, які обчислюються за такою кривою, виконано у [10]: глибина просідання, площа під

кривою за час інциденту та час повернення показника до необхідного рівня. Саме такі умови збереження функціонування і завершеного відновлення сформульовано у підрозділі 1.1 через величини $q(t)$, $q_{\min}$, q_{req} та T_{rec} .

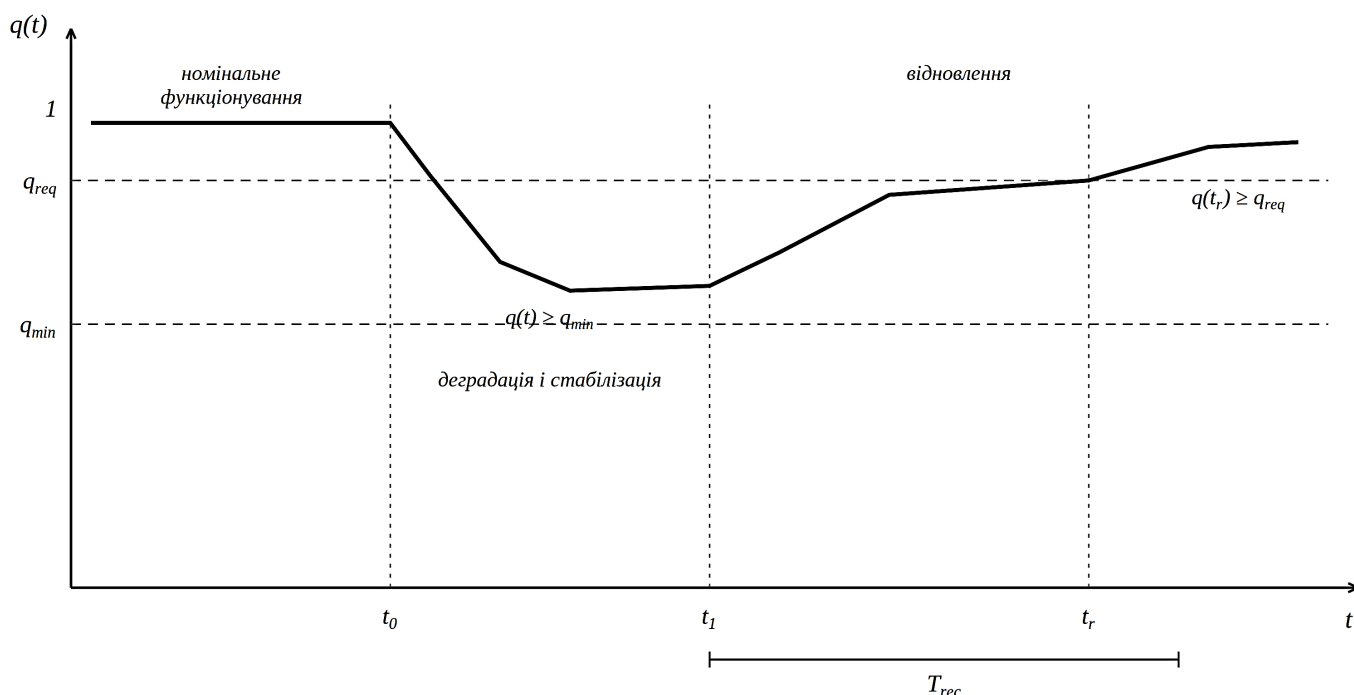


Рисунок 1.1 – Фази деградації та відновлення показника виконання критичних функцій за [10]

Особливості імерсивних систем формалізовано у вимогах до наднадійної віртуальної реальності з малою затримкою [11]: сеанс вимагає одночасного дотримання жорстких бюджетів затримки, високої пропускної здатності та неперервності подання кадрів, а порушення будь-якої з цих умов користувач сприймає негайно. Тому загальні міри резильєнтності слід застосовувати до VR-систем з урахуванням людини у контурі керування: та сама технічна деградація має різну ціну для різних сценаріїв застосування.

Розглянуті роботи дають узгоджений понятійний апарат і загальні міри, проте не пропонують способу пов'язати інтегральний показник резильєнтності VR-системи

з вимірюваними параметрами середовища та зі станом окремих компонентів. Ця прогалина визначає перше завдання дослідження.

1.3.2 Методи самовідновлення програмного забезпечення

Систематизацію механізмів відновлення програмних систем у хмарних середовищах виконано у [13]. Реактивні механізми охоплюють контрольні точки стану з відкатом і перезапуском, повторне виконання запитів та реплікацію; проактивні передбачають профілактичне омолодження програмного забезпечення та випереджувальну міграцію компонентів, які усувають накопичені дефекти виконання до того, як вони призведуть до відмови. Для VR-системи профілактичні дії доцільно виконувати у паузах взаємодії, щоб уникнути видимого переривання сеансу.

Практичні шаблони побудови надійних розподілених застосунків узагальнено у [14]: періодичні перевірки працездатності, автоматичний перезапуск контейнерів, реплікація та винесення стану компонентів у спеціалізовані сховища.

Для мікросервісних архітектур каталог шаблонів резильєнтності формалізовано у [15]: повторні спроби, часові обмеження, розмикачі та перегородки, які утримують збій у межах невеликої відновлюваної одиниці.

Спільною рисою розглянутих механізмів є цикл самовідновлення: виявлення відхилення, діагностування, вибір дії відновлення та її застосування з подальшою перевіркою результату. Для імерсивних застосувань визначальним є конфлікт між тривалістю дії відновлення та вимогами реального часу, оскільки вікно часу на відновлення обмежене тривалістю кількох кадрів.

Перелічені механізми утворюють послідовність дій самовідновлення, подану на рис. 1.2: періодична фіксація стану, виявлення збою за телеметрією, вибір між відкатом, мікроперезапуском і профілактичним омолодженням, перевірка відновленого стану та повернення до штатного функціонування. Водночас жодна з розглянутих робіт не визначає, які саме параметри VR-середовища слід

використовувати як ознаки деградації і як пов'язати вибір дії відновлення з кількісною оцінкою стану системи.

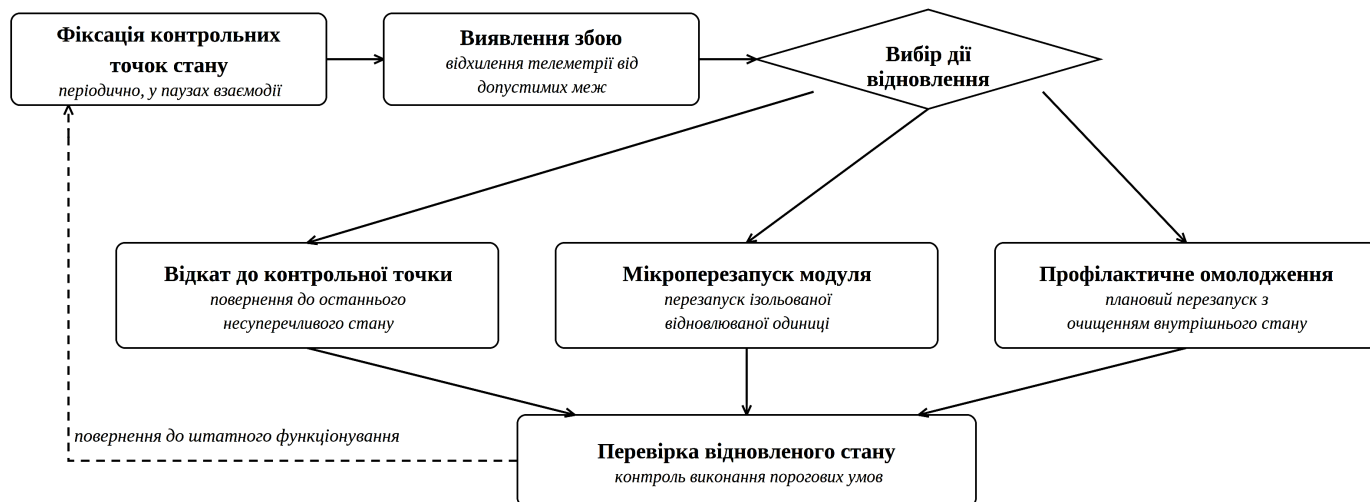


Рисунок 1.2 – Узагальнена послідовність дій самовідновлення програмного забезпечення

1.3.3 Методи автономного керування та самоадаптивні програмні архітектури

Сучасне узагальнення самоадаптивних програмних систем подано у [16]: система доповнюється керуючим модулем, який реалізує замкнений контур з етапами моніторингу, аналізу, планування та виконання над спільною базою знань і здійснює адаптацію без участі оператора. Самовідновлення при цьому розглядається як один із чотирьох класів адаптації поряд із самоконфігуруванням, самооптимізацією та самозахистом. Схему контуру подано на рис. 1.3.

Систематичний огляд самоадаптивних систем за категоріями та прикладними галузями виконано у [17]. Проаналізовано типові архітектури контурів, засоби спостереження і виконавчі механізми та показано, що найбільший ефект

самоадаптація дає у системах з мінливим середовищем і жорсткими вимогами до якості обслуговування, до яких належать і імерсивні застосування.

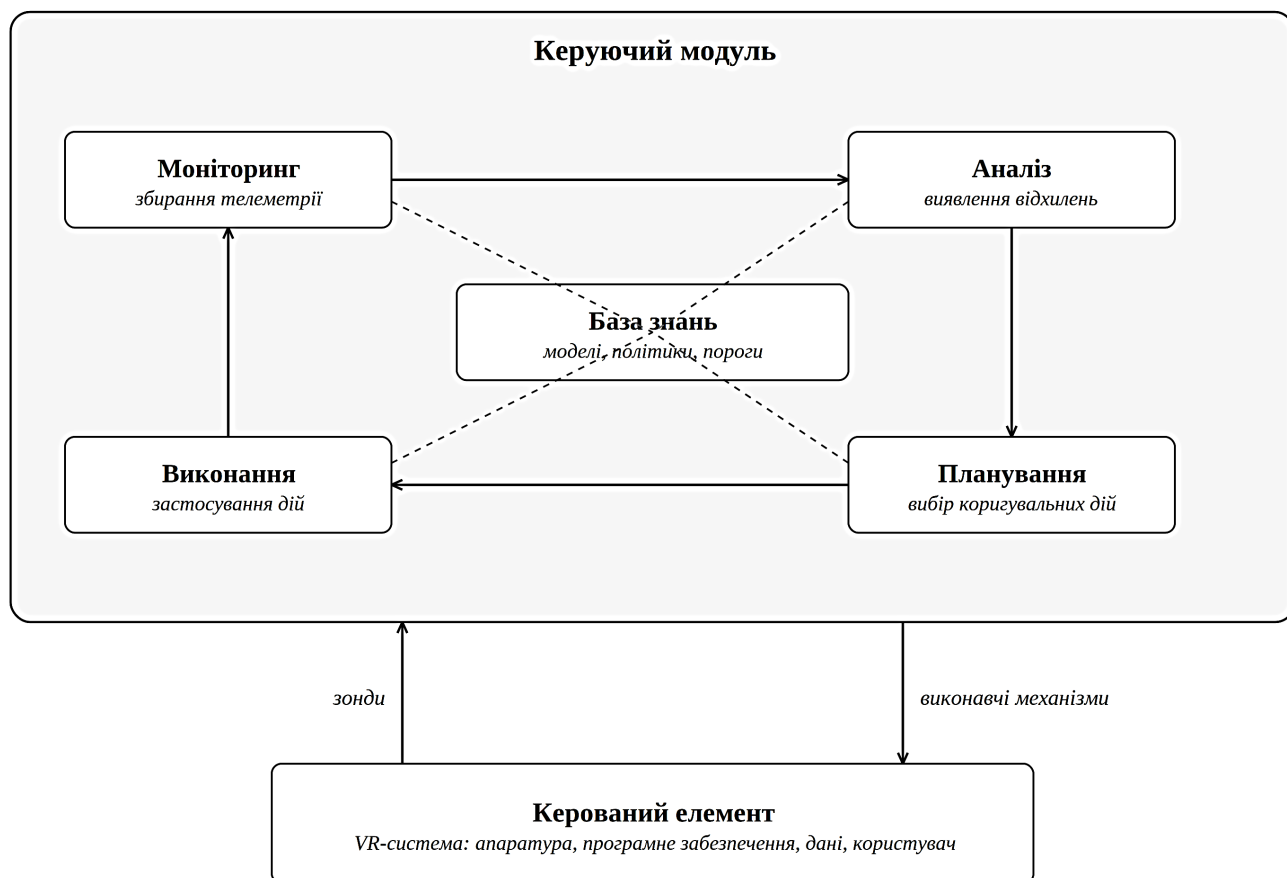


Рисунок 1.3 – Замкнений контур автономного керування за [16]

Прикладом зрілої самоадаптації у мультимедійних системах є доставляння панорамного відео: у [18] узагальнено рішення, які за спостереженням каналу і напрямку погляду користувача змінюють швидкість потоку, розбиття кадру на плитки та глибину буфера, утримуючи якість сприйняття під час деградації мережі. Для VR-систем аналогічними виконавчими механізмами слугують рівень деталізації сцени, частота кадрів, розподіл обчислень і режими енергоспоживання.

Вимоги та рішення для віртуальної реальності у стільникових мережах систематизовано у [19]: неперервність сеансу підтримується поєднанням кешування, обчислень на межі мережі та адаптивного кодування. Водночас розглянуті контури залишають відкритими два питання, критичні для імерсивних систем: як кількісно

оцінити стан системи за десятками різномірних параметрів середовища і як синтезувати конфігурацію, що одночасно покращує кілька суперечливих показників. Відповіді на ці питання формуються у розділах 2 і 3 цієї роботи.

1.3.4 Методи проєктування, орієнтованого на відновлення

Проєктування, орієнтоване на відновлення, виходить із того, що збої апаратури, програмного забезпечення та помилки супроводження є неминучими, тому головним показником якості системи стає час повернення до працездатного стану. Масштабне дослідження практик постачання програмного забезпечення [20] відносить час відновлення після збою до чотирьох ключових показників ефективності інженерних організацій, що зміщує пріоритети конструювання: замість подальшого ускладнення засобів запобігання розвиваються засоби швидкої локалізації, ізоляції та усунення наслідків збою.

Операційні практики, які скорочують цей час, систематизовано у [21]: поетапні канаркові випуски, автоматичне відкочування змін, сценарії реагування на інциденти та їх регулярні розбори. На рівні архітектури застосунку відповідні шаблони стабільності описано у [22]: часові обмеження, розмикачі, перегородки та швидке відмовляння, які утримують збій у межах однієї відновлюваної одиниці. Ці прийоми узгоджуються з механізмами, розглянутими у пункті 1.3.2, і спираються на ті самі контрольні точки та ізоляцію стану.

Проєктування з розрахунку на відновлення закріплено і у практиці побудови захищених та надійних систем [23], де унормовано вимоги до відкочування, відтворюваності конфігурацій та обмеження радіуса впливу змін. Для XR-систем цінність підходу полягає у явному нормуванні часу відновлення: умови завершеного відновлення, сформульовані у підрозділі 1.1 через граничний час $T_{\text{рес}}$, безпосередньо відповідають цій постановці, оскільки механізм відновлення вважається придатним лише тоді, коли повертає показник виконання критичних функцій до необхідного рівня у межах відведеного часу.

1.3.5 Методи синтезу апаратної надлишковості та відмовостійкості у VR-системах

Резервування апаратного забезпечення та відмовостійкість потрібні для безперервної роботи VR-систем. Відмова складного апаратного компонента може погіршити продуктивність і перервати взаємодію. Резервні компоненти та механізми відновлення зменшують ризик зупинки, підвищують довіру користувачів і захищають критичні застосунки, зокрема медичні симуляції, навчальні вправи та промислові операції.

Методи досягнення апаратного резервування є загальними для систем загального призначення та VR, однак існують спеціальні методи, пов'язані з обробкою цифрових зображень. В умовах, таких як космос, електронні схеми схильні до збоїв через радіацію. Резервування часто використовується для зменшення цих збоїв разом з урахуванням низької потужності і малого розміру для підвищення енергоефективності та зменшення ваги і вартості. Триплетне модульне резервування (TMR) є популярним підходом, який споживає більше площі та енергії порівняно з одиничною схемою. Вибіркове TMR та резервування зниженої точності на основі більшості голосів пропонують перспективні рішення, особливо для застосувань, толерантних до помилок, таких як обробка цифрових зображень у космічних системах. Ці методи оцінюються за допомогою MATLAB та фізичної реалізації з використанням 28-нм CMOS технології [24].

Використання технік резервування апаратного забезпечення на базі FPGA може значно допомогти у створенні відмовостійких цифрових фільтрів для VR-середовищ. Зростаюча залежність від комунікацій та обробки сигналів в повсякденному житті вимагає надійніших пристроїв з мінімальними перехідними помилками. Для цього застосовується метод п'ятиразового резервування (5MR) [25], що підвищує надійність апаратного забезпечення, схильного до відмов. Фільтри з кінцевою імпульсною характеристикою важливі для обробки сигналів, включаючи шумозаглушення, використовуючи ПЛІС методи з Xilinx Vivado EDA.

Результати симуляції показують, що всі конфігурації 5MR ефективно знижують шум сигналу ЕКГ, використовуючи інструмент Xilinx EDA, оптимізуючи використання площі. Інтеграція цих конфігурацій у VR-апаратне забезпечення може значно покращити стабільність і надійність, забезпечуючи більш плавний досвід віртуальної реальності.

1.3.6 Методи перевірки резильєнтності контрольованим внесенням відмов

Механізми відновлення потребують перевірки до настання реальних інцидентів. Дисципліну такої перевірки узагальнено у [26]: висувається гіпотеза про усталений стан системи, у середовищі, наближеному до експлуатаційного, вноситься корекція, після чого поведінка системи порівнюється з очікуваною. Планомірні експерименти виявляють приховані залежності та підтверджують, що засоби відновлення працездатні саме тоді, коли вони потрібні.

Промислове впровадження цього підходу описано у [27]: платформа автоматично формує контрольну та експериментальну групи запитів, вносить відмову лише у малу частку трафіку і безперервно порівнює ключові показники обох груп, зупиняючи експеримент у разі виходу за допустимі межі. Для VR-систем аналогічне внесення відмов охоплює погіршення характеристик мережі, обмеження обчислювальних ресурсів, імітацію перегріву та збої окремих сенсорів.

Дослідницькі інструменти поширюють контрольоване внесення відмов на рівень виконуваного коду: у [28] представлено систему, яка під час роботи застосунку збурює оброблення виняткових ситуацій, автоматично перевіряє гіпотези про поведінку кожного обробника і формує кількісні висновки про здатність програмного забезпечення відновлюватися після збоїв.

Для імерсивних систем перевірка резильєнтності має поєднуватися з кількісним оцінюванням: результати внесення відмов набувають сенсу лише відносно інтегрального показника стану та порогових умов відновлення. Саме таке поєднання реалізовано у розділі 4 цієї роботи, де сценарії деградації відтворюються у

симуляційному середовищі, а відновлення підтверджується поверненням інтегрального показника до безпечної зони.

1.3.7 Методи забезпечення резильєнтності, специфічні для імерсивних систем

Окрему групу утворюють механізми, розроблені безпосередньо для імерсивних систем. Їхньою спільною ознакою є прив'язка до конвеєра відтворення кадрів і до бюджету затримки «рух – фотон», через що вони не мають прямих аналогів у серверних застосуваннях.

Компенсація пропущених кадрів реалізується часовим передбаченням пози та повторним проєктуванням раніше сформованого кадру на уточнене положення голови. Механізм утримує видиму частоту подання за короткочасного зниження продуктивності графічної підсистеми, тобто діє як локальний засіб збереження критичної функції подання зображення. Вимоги до бюджету затримки та до механізмів її компенсації для імерсивних послуг унормовано у [130], а прикладні програмні інтерфейси, що надають доступ до передбаченої пози, стандартизовано у [132]. Межі застосовності механізму визначаються тривалістю інтервалу передбачення [90].

Керована зміна обчислювального навантаження реалізується фовеальним рендерингом, динамічною зміною рівня деталізації, змінним масштабом буфера подання та спрощенням моделі освітлення. Огляд методів фовеального рендерингу, їх реалізацій та кількісних оцінок вивільнюваного обчислювального ресурсу виконано у [98, 99]. Саме такі механізми утворюють основу мітигаційних дій, параметризованих у підрозділі 3.2.

Рознесення обчислень між пристроєм і межею мережі, кешування та передбачення видимості розглянуто у роботах, присвячених доставлянню імерсивного відео та віртуальній реальності у стільникових мережах [18, 19, 101]. Архітектури мобільних обчислень на межі мережі для імерсивних застосувань систематизовано у [100, 124], а способи адаптивного розподілу потоків між

користувачами – у [120]. Такі рішення підвищують стійкість до мережових впливів, проте переносять частину ризику на канал зв'язку: за одночасного погіршення характеристик каналу і зростання локального навантаження механізм втрачає працездатність, оскільки обидва напрями розвантаження виявляються заблокованими. Референсну архітектуру обчислень на межі мережі унормовано у [129], а формати подання та профілі потокового передавання імерсивного вмісту – у [115, 131].

Надлишковість каналів відстеження реалізується злиттям оптичних та інерційних даних, що дає змогу зберігати визначення пози за короткочасної втрати оптичного каналу через засвітку чи перекриття маркерів. Апаратні засоби надлишковості, розглянуті у пункті 1.3.5, зокрема модульне резервування та резервування зниженої точності [24, 25], застосовні до сенсорних трактів і трактів обробки зображення, однак їх упровадження обмежене вимогами до маси, габаритів та енергоспоживання носимого обладнання.

Механізми безпеки користувача охоплюють динамічну зміну меж безпечної зони, зниження інтенсивності руху віртуальної камери, перехід на дискретні способи переміщення та примусове завершення сеансу за перевищення допустимих показників. Експериментальні дослідження самопочуття користувачів у імерсивних середовищах підтверджують, що ці механізми безпосередньо впливають на симптоми кінетозу [12, 91, 93], а огляди ергономічних ризиків імерсивної роботи [94] відносять їх до обов'язкових складових безпечного сеансу. Отже, вони належать до засобів забезпечення резильєнтності, а не лише до засобів підвищення комфорту. Технічні засоби зменшення кінетозу для шоломів віртуальної реальності унормовано окремим стандартом [116].

Керування тепловим режимом носимого обладнання реалізується обмеженням потужності, динамічним масштабуванням напруги і частоти [55, 56] та перенесенням обчислень. Особливістю імерсивних систем є те, що тепловий режим одночасно є технічним обмеженням і безпековою вимогою, оскільки поверхня обладнання перебуває у тривалому контакті з тілом користувача, а допустимі температури дотику унормовані [97].

Спільним обмеженням розглянутої групи є локальність. Кожен механізм оптимізує окремий показник у межах власного рівня, спирається на власний набір ознак і не пов'язує рішення про своє застосування з кількісною інтегральною оцінкою стану системи. Через це одночасне застосування кількох механізмів не гарантує покращення стану.

1.4 Узагальнене порівняння відомих методів забезпечення резильєнтності XR-систем

Групи методів зіставлено за вісьмома критеріями, що впливають із визначення резильєнтності, критичних функцій і профілів місії підрозділу 1.1. Критерії узгоджено з моделлю якості систем [89], вимогами кіберрезильєнтності [9, 107], процесами життєвого циклу [117] та характеристиками довіри до автоматизованих систем [128].

К1 – кількісне оцінювання поточного стану системи єдиним інтегральним показником, визначеним на нормованій шкалі.

К2 – врахування різнорідних за фізичною природою параметрів середовища, мережі, навантаження, апаратних обмежень та поведінки користувача у межах однієї моделі.

К3 – локалізація критичної пари «фактор – компонент», тобто визначення джерела деградації з точністю до конкретної взаємодії, а не до системи в цілому.

К4 – систематичний синтез мітигаційних дій за виявленим джерелом деградації замість добирання дій за досвідом розробника.

К5 – одночасне врахування кількох суперечливих критеріїв резильєнтності: надійності, доступності, відмовостійкості, цілісності, часу відновлення, стабільності продуктивності та безпеки користувача.

К6 – врахування користувача як повноправного компонента системи, стан якого оцінюється разом із технічними компонентами.

К7 – кількісне підтвердження завершеності відновлення шляхом перевірки умов, сформульованих через показник виконання критичних функцій і граничний час відновлення.

К8 – придатність до часових обмежень імерсивної взаємодії, тобто виконуваність повного циклу виявлення та реагування у вікні, сумірному з тривалістю кількох кадрів.

Результат зіставлення груп методів за наведеними критеріями подано у таблиці 1.4. Позначення «+» відповідає повному виконанню критерію в межах групи методів, «±» – частковому або опосередкованому виконанню, «-» – невиконанню.

Таблиця 1.4 – Зіставлення відомих груп методів забезпечення резильєнтності за критеріями К1–К8

Група методів	К1	К2	К3	К4	К5	К6	К7	К8
Кількісне оцінювання резильєнтності та моделі деградації і відновлення (1.3.1)	+	–	–	–	±	–	+	±
Самовідновлення програмного забезпечення (1.3.2)	–	–	±	±	–	–	±	±
Автономне керування та самоадаптивні архітектури	±	±	±	±	±	–	±	±
Проектування, орієнтоване на відновлення	±	–	–	±	–	–	+	–
Апаратна надлишковість та відмовостійкість	–	–	±	±	–	–	–	+
Перевірка резильєнтності контрольованим внесенням відмов	±	±	±	–	–	–	±	–
Механізми, специфічні для імерсивних систем	–	±	–	±	–	±	–	+

За таблицею 1.4 кількісне оцінювання забезпечує нормовану міру стану (K1), проте повне поєднання різнорідних параметрів XR-середовища (K2) і локалізація конкретної взаємодії «фактор – компонент» (K3) залишаються нерозв'язаними. Оцінювання компонента в цілому не розрізняє наслідків мережевих, теплових і поведінкових впливів на нього.

Каталоги шаблонів задають можливі дії без загального правила їх синтезу [15], тому виконання K4 потребує окремої процедури вибору. Суперечливість критеріїв K5 вимагає мультикритеріальних методів [110, 112]. Урахування користувача (K6) у специфічних імерсивних механізмах [94] залишається частковим і потребує узгодження з технічними показниками.

Методи кількісного оцінювання та проєктування, орієнтованого на відновлення, нормують час повернення працездатності (K7), але не пов'язують його з внеском кожної коригувальної дії. Апаратні та імерсивні механізми підтримують часові обмеження взаємодії (K8), тоді як серверні механізми можуть допускати триваліші перерви.

За рядками таблиці видно розмежування ролей: оцінювання задає міри, самовідновлення реалізує реакцію, а самоадаптивні архітектури організовують контур керування [16, 105, 106]. Апаратне резервування захищає окремі тракти, а внесення відмов перевіряє механізми реагування [113, 114]. Ці ролі потребують спільного опису стану та узгодженого критерію вибору дій.

Отже, необхідний єдиний ітераційний процес, який поєднує оцінювання стану, локалізацію критичних взаємодій, мультикритеріальний синтез мітигації та перевірку завершеного відновлення. Спільні параметри й шкали мають забезпечити узгодженість часткових методів. Це визначає постановку задачі у підрозділі 1.5.

1.5 Висновки та постановка задачі

Проведений аналіз показує, що дослідження резильєнтності обчислювальних систем сформували повний набір базових механізмів: кількісні міри деградації та відновлення, контрольні точки і відкат, профілактичне омолодження,

мікроперезапуск, замкнені контури самоадаптації, апаратну надлишковість та методики контролюваного внесення відмов. Кожен з цих механізмів підтверджений експериментально у своїй предметній галузі.

Водночас безпосереднє перенесення цих результатів на XR-системи наштовхується на суттєві перешкоди. Розглянуті методи створювалися переважно для серверних та мережевих застосувань, де допустимі секунди недоступності і відсутня людина всередині контуру керування. В імерсивній системі вплив діє одночасно на апаратну частину, програмне забезпечення, дані та користувача, а вікно часу на реакцію обмежене вимогами сенсомоторної узгодженості.

Перша прогалина стосується оцінювання стану. Загальні міри резильєнтності описують траєкторію одного показника, проте не визначають, як звести до нього десятки різнорідних параметрів VR-середовища, а без такого зведення неможливо ані виявити деградацію на ранній стадії, ані порівняти альтернативні конфігурації.

Друга прогалина стосується вибору дій. Контури самоадаптації визначають послідовність кроків, але не дають методу синтезу конфігурації за кількома суперечливими критеріями одночасно: надійністю, часом відновлення, продуктивністю та безпечністю користувача.

Третя прогалина стосується перевірки. Контрольоване внесення відмов дає якісний висновок про працездатність механізмів, однак для VR-систем потрібне кількісне підтвердження.

Отже, необхідним є єдиний ітераційний процес, який поєднує кількісне оцінювання стану VR-системи, локалізацію вузьких місць, синтез мітигаційних стратегій та повторне оцінювання після їх застосування, а також програмний засіб, що автоматизує цей процес і дає змогу відтворювати сценарії деградації та відновлення.

Метою дисертаційного дослідження є забезпечення резильєнтності систем віртуальної та доповненої реальності шляхом розроблення архітектури VR-системи та методів оцінювання стану VR-системи й синтезу мітигаційних стратегій для відновлення працездатності VR-системи в умовах динамічно змінюваного зовнішнього середовища.

Розв'язання науково-прикладної задачі потребує вирішення таких наукових задач:

- розробка методу синтезу мітигаційних стратегій, який дозволяє забезпечувати резильєнтність системи в умовах динамічно змінюваного зовнішнього середовища;
- розробка методу оцінювання стану VR-системи, який ґрунтується на оцінці впливу факторів динамічно змінюваного середовища на компоненти системи і використовує систему вагових коефіцієнтів, отриману шляхом застосування нечіткої логіки;
- розвиток архітектури VR-систем, яка інтегрує в собі комплекс компонентів, пов'язаних із програмно-апаратними засобами, методи оцінювання, синтезу мітигаційних стратегій в єдиний програмно-апаратний засіб, що дозволяє здійснювати резильєнтне функціонування систем;
- удосконалення моделі резильєнтного функціонування систем віртуальної та доповненої реальності, у якій враховано вплив зовнішніх факторів, таких як зміна стану мережі, поведінка користувачів, навантаження на систему, апаратні обмеження та фактори навколишнього середовища на основні компоненти системи, включаючи апаратне та програмне забезпечення, дані та користувачів.

Основні результати розділу опубліковані у працях [80, 83, 86].

РОЗДІЛ 2. МОДЕЛЬ ТА МЕТОДИ ЗАБЕЗПЕЧЕННЯ РЕЗИЛЬЄНТНОСТІ СИСТЕМ ВІРТУАЛЬНОЇ ТА ДОПОВНЕНОЇ РЕАЛЬНОСТІ

Платформи XR містять апаратні засоби, програмні модулі, засоби обробки даних і користувацький контур. До апаратної частини належать сенсори руху, дисплеї, контролери, графічні та центральні процесори. Програмна частина охоплює драйвери, прошивки, проміжні програмні шари та рушії візуалізації. Засоби обробки даних забезпечують зберігання, передавання, захист і синхронізацію потоків. Кожна група має власні механізми деградації, тому модель поділяє їх за основною функцією та оцінює окремо.

Особливу увагу необхідно приділити інтеграції підходів самовідновлення. Самовідновлення передбачає виявлення аномальних станів; переповнення буферів, падіння FPS нижче критичного порогу, невідповідність відліків акселерометра, а також автоматичне застосування коректувальних дій: перезапуск модулів, зниження графічних налаштувань, переведення в режим деградованої графіки, перерозподіл ресурсів між CPU і GPU.

Таким чином, платформи віртуальної та доповненої реальності, будучи складними системами реального часу з багаторівневою компонентною структурою, вимагають комплексного підходу до проєктування резильєнтності. Поєднання резервування, самовідновлення, адаптивного планування й захищеного обігу даних дозволяє забезпечити безперебійну роботу навіть за несприятливих умов, таких як перевантаження апаратури й перебої із мережею, або цілеспрямовані кібератаки з метою дестабілізації платформи. Такий рівень надійності є запорукою безпечного й комфортного користувацького досвіду, а також гарантією того, що XR-системи можуть ефективно застосовуватися в критичних сферах, де часові затримки та відмови неприйнятні.

2.1 Модель резильєнтного функціонування систем віртуальної та доповненої реальності

Задача забезпечення резильєнтності систем віртуальної та доповненої реальності починається з їхньої детальної чисельної оцінки, яка дозволяє перейти від абстрактних понять «резильєнтність» і «надійність» до конкретних числових значень і метрик. Через багаторівневу компонентну структуру XR-платформ необхідно розглянути кожен модуль окремо: апаратні сенсори, контролери, графічні рушії, мережеві протоколи і модулі обробки даних, а також зібрати для них набір показників, що характеризують продуктивність, затримки, відмовостійкість і ступінь захищеності. На основі цих індивідуальних значень формується матриця, яка відображає внесок кожного елемента в загальну резильєнтність системи.

Побудована декомпозиція є інваріантною щодо застосування, тоді як допустимі межі деградації окремих компонентів залежать від профілю місії, введеного у підрозділі 1.1.2. Тому модель параметризується профілем: та сама структура компонентів і факторів використовується для всіх застосувань, а профіль задає нормувальні величини оцінювальних функцій, коефіцієнти значущості критичних функцій і порогові умови відновлення. Така організація дає змогу переносити модель між застосуваннями без зміни її структури, замінюючи лише набір нормувальних параметрів.

Кожен компонент моделі розглядається одночасно як об'єкт впливу та як носій частини критичних функцій, визначених у підрозділі 1.1. Апаратне забезпечення відповідає за відстеження положення, подання зображення та утримання теплового режиму. Програмне забезпечення відповідає за формування кадрів, узгодженість сенсорних каналів і виконання мітигаційних дій. Компонент даних відповідає за цілісність, автентичність і своєчасність потоків трекінгової та прикладної інформації. Користувацький компонент відповідає за збереження меж безпечної зони та за фізіологічну припустимість сеансу. Таке співвіднесення компонентів із класами критичних функцій забезпечує безпосередній перехід від значення окремої оцінювальної функції до висновку про виконуваність відповідної критичної функції,

що використовується у підрозділі 3.1 під час опису переходів між станами компонентів.

Відсутність окремого мережевого компонента у векторі системи не означає нехтування мережею: стан мережі є фактором впливу, а не компонентом, оскільки мережа не виконує критичних функцій самостійно, а змінює умови їх виконання іншими компонентами. Такий поділ дає змогу уникнути подвійного врахування одного й того самого впливу, який виникає, якщо мережу одночасно розглядати як джерело впливу та як об'єкт впливу.

Вхідні дані зводяться до безрозмірних відношень і обмежуються інтервалом $[0; 1]$. Вагові коефіцієнти визначають важливість кожної пари «фактор, компонент». Інтегральний показник резильєнтності є зваженою сумою двадцяти оцінювальних функцій. Значення окремих елементів матриці дають змогу локалізувати причину деградації, а інтегральне значення характеризує поточний стан системи.

На підставі цих результатів можуть застосовуватися цільові методи зменшення негативного впливу: динамічне зниження графічної складності при падінні FPS або перерозподіл обчислювальних ресурсів на користь обробки сенсорних даних у разі втрати пакету мережевих даних. Реалізація механізму моніторингу в реальному часі дозволяє відстежувати зміни інтегрального показника резильєнтності та автоматично коригувати параметри роботи системи без втручання оператора. Завдяки цьому формується зворотний зв'язок: при досягненні критичних порогів спрацьовують контрзаходи, а накопичена статистика за тривалий період дозволяє будувати прогностичні моделі та планувати превентивні оновлення або апаратні модернізації.

У результаті формується система постійного моніторингу та адаптації. Вона фіксує поточний стан платформи і за історичними даними визначає стратегії підтримки потрібного рівня резильєнтності. Це забезпечує своєчасну реакцію на зміну мережевих умов, температурного режиму та обчислювального навантаження.

Досліджувану XR-систему подамо впорядкованою четвіркою компонентних множин:

$$\mathcal{V} = (H, S, D, U) \quad (2.1)$$

де H є множиною апаратних компонентів, S є множиною програмних компонентів, D є множиною засобів зберігання, обробки, передавання та захисту даних, U є множиною характеристик користувацького контуру.

До множини H належать шоломи, контролери, сенсори, дисплеї, процесори, акумулятори та інші апаратні компоненти:

$$H = \{H_1, H_2, \dots, H_{n_H}\}, \quad |H| = n_H \quad (2.2)$$

де n_H є кількістю елементів множини H .

До множини S належать операційна система, драйвери, прошивки, прикладні модулі та рушії візуалізації:

$$S = \{S_1, S_2, \dots, S_{n_S}\}, \quad |S| = n_S \quad (2.3)$$

де n_S є кількістю елементів множини S .

До множини D належать засоби, основною функцією яких є зберігання, обробка, передавання або захист даних:

$$D = \{D_1, D_2, \dots, D_{n_D}\}, \quad |D| = n_D \quad (2.4)$$

де n_D є кількістю елементів множини D .

Множина U містить характеристики користувацького контуру, зокрема патерни руху, фізіологічний стан, когнітивне навантаження та параметри просторового сприйняття:

$$U = \{U_1, U_2, \dots, U_{n_U}\}, \quad |U| = n_U \quad (2.5)$$

де n_U є кількістю елементів множини U .

Повну множину компонентів утворює диз'юнктне об'єднання:

$$C = H \cup S \cup D \cup U \quad (2.6)$$

Множини H , S , D та U не перетинаються, оскільки кожний елемент віднесено до групи за його основною функцією. Їх потужності n_H , n_S , n_D та n_U в загальному випадку різні. Якщо фізичний пристрій виконує кілька функцій, у моделі створюються окремі функціональні представлення його апаратної, програмної та інформаційної ролей без повторного включення одного елемента до різних множин.

На компоненти діють п'ять груп факторів. Їх подамо вектором-стовпцем:

$$E = (E_N, E_{ENV}, E_B, E_S, E_{HW})^T \quad (2.7)$$

де E_N описує стан мережі, E_{ENV} описує параметри навколишнього середовища, E_B описує поведінку користувача, E_S описує системне навантаження, E_{HW} описує апаратні обмеження.

Кожній парі «фактор, компонент» ставиться у відповідність безрозмірна оцінювальна функція $f_{i,j}(t)$ зі значенням на інтервалі $[0; 1]$. Сукупність функцій утворює матрицю:

$$F(t) = \begin{bmatrix} f_{N,H} & f_{N,S} & f_{N,D} & f_{N,U} \\ f_{ENV,H} & f_{ENV,S} & f_{ENV,D} & f_{ENV,U} \\ f_{B,H} & f_{B,S} & f_{B,D} & f_{B,U} \\ f_{S,H} & f_{S,S} & f_{S,D} & f_{S,U} \\ f_{HW,H} & f_{HW,S} & f_{HW,D} & f_{HW,U} \end{bmatrix} \in [0,1]^{5 \times 4} \quad (2.8)$$

Матриця $F(t)$ має фіксовану розмірність 5×4 . Її рядки відповідають п'яти групам факторів, стовпці відповідають чотирьом компонентним множинам. Кома в нижньому індексі розділяє позначення фактора і компонента. Операція множення в індексах не використовується.

Кожний елемент матриці показує збережену частку працездатності відповідного компонента під дією заданого фактора у момент часу t .

Мережевий рядок матриці містить функції (2.9)–(2.12). Вони описують вплив пропускної здатності, затримки, джитеру і втрат пакетів на апаратне забезпечення, програмну частину, дані та користувацький контур.

Рядок умов середовища містить функції (2.13)–(2.16). Температура, шум і освітленість можуть спричинити перегрів, помилки сенсорів, зниження якості даних і дискомфорт користувача.

Рядок поведінки користувача містить функції (2.17)–(2.20). Інтенсивні або аномальні рухи збільшують навантаження, ускладнюють обробку команд, змінюють обсяг даних і можуть погіршити комфорт.

Стійкість до аномальних дій користувача забезпечується перевіркою вхідних потоків, обмеженням темпу подій, фільтрацією траєкторій, буферизацією та

контрольованим зниженням деталізації. Кількісні оцінки наведено у формулах (2.17)–(2.20), відповідні мітигаційні дії формалізовано у розділі 3.

Рядок системного навантаження містить функції (2.21)–(2.24). Перевантаження може підвищити температуру, зменшити частоту кадрів, сповільнити обробку даних і збільшити затримку взаємодії.

Рядок апаратних обмежень містить функції (2.25)–(2.28). До них належать температура компонентів, доступна потужність, ресурс акумулятора та місткість засобів зберігання.

Кіберзагрози враховуються за їх вимірюваними наслідками. Атака на канал зв'язку змінює затримку, джитер і втрати пакетів. Виснаження ресурсів підвищує системне навантаження. Компрометація даних знижує оцінки інформаційної підсистеми. Маніпуляція сенсорними потоками проявляється як аномальна поведінка. Походження впливу не змінює порядок обчислення, але враховується під час вибору мітигаційної стратегії.

Структуру системи, компонентні множини та групи факторів подано на рисунку 2.1.

Вплив мережі на апаратне забезпечення оцінюється за відношеннями фактичної пропускної здатності B до потрібної B_{req} , допустимої затримки L_{req} до фактичної L та еталонної потужності P_{ref} до фактичного енергоспоживання P :

$$f_{N,H}(t) = \left[\frac{B(t)}{B_{req}} \frac{L_{req}}{L(t)} \frac{P_{ref}}{P(t)} \right]_0^1 \quad (2.9)$$

Усі три множники у формулі (2.9) безрозмірні. B і B_{req} вимірюються в однакових одиницях пропускної здатності, L_{req} і L у мілісекундах, P_{ref} і P у ватах.

Формула (2.9) описує спільний ефект трьох відношень, тому її результат залежить від їх добутку. Розглянемо аналітичний приклад: фактична пропускна здатність становить половину потрібної, затримка дорівнює допустимій, а потужність – еталонній. Необмежений добуток дорівнює 0,5, отже оцінка також становить 0,5. Якщо за тих самих умов затримка подвоюється, друге відношення стає рівним 0,5, а загальна оцінка зменшується до 0,25. Це наслідок заданої функціональної залежності, а не окремий результат вимірювання обладнання.

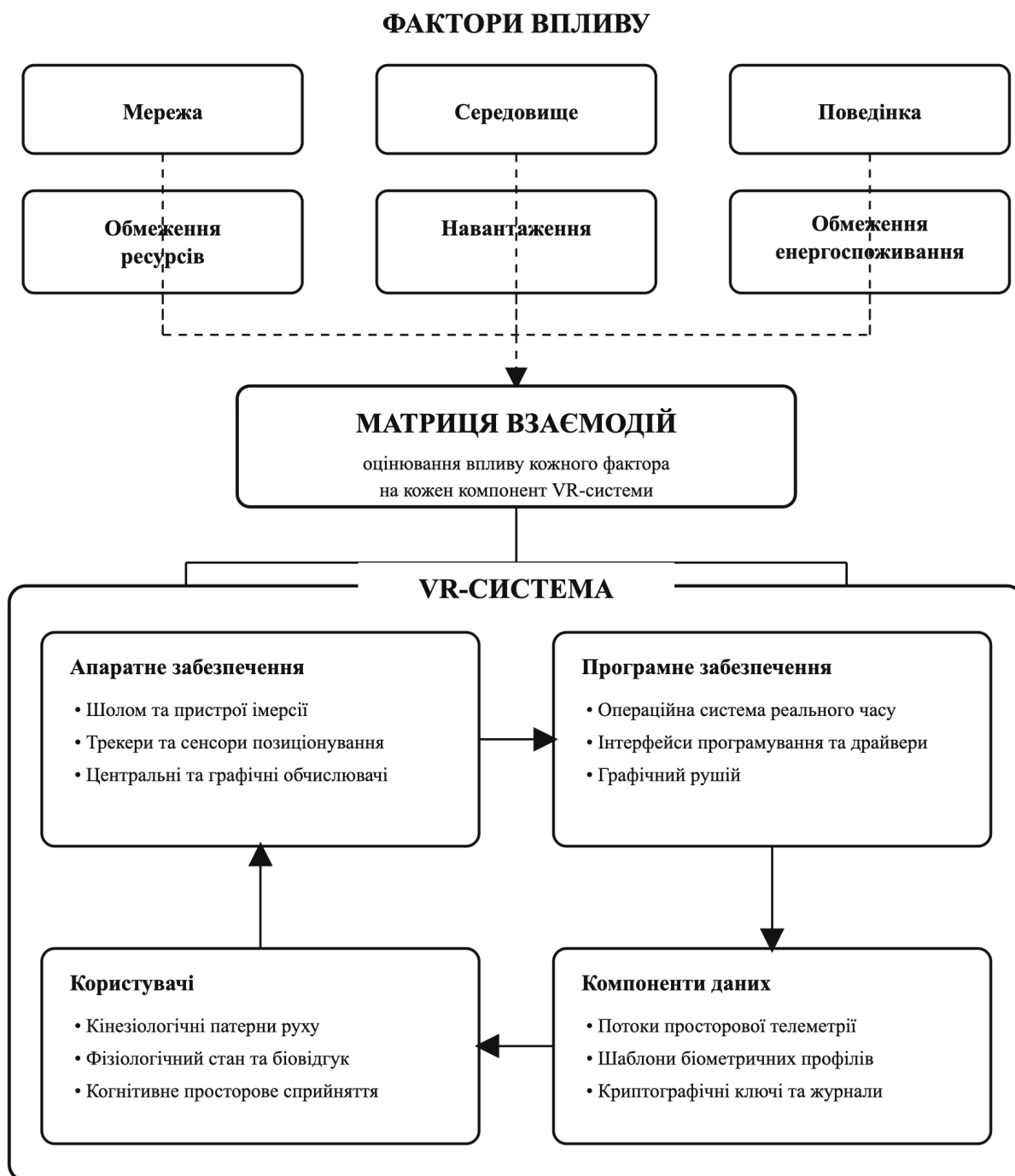


Рисунок 2.1 – Архітектура системи віртуальної та доповненої реальності

Для кожної пари «фактор, компонент» визначено окрему безрозмірну функцію. Оператор $[x]_0^1$ означає обмеження значення x інтервалом $[0; 1]$.

Оператор обмеження застосовується після множення. Наприклад, двократний запас пропускної здатності та двократне перевищення затримки за еталонної потужності дають добуток, рівний одиниці. Таким чином, у цій оцінці можливе взаємне компенсування відношень. Окремі вимоги до затримки при цьому

перевіряються іншими функціями мережевого рядка та умовами критичних функцій профілю місії. Одиничне значення одного елемента матриці не скасовує цих перевірок.

Вплив мережі на програмне забезпечення задається пороговою функцією затримки L та джиттеру J :

$$f_{N,S}(t) = \begin{cases} 1, & L(t) \leq L_{\max} \wedge J(t) \leq J_{\max}, \\ 0, & \text{інакше,} \end{cases} \quad (2.10)$$

$L_{\max}$ і $J_{\max}$ мають ті самі одиниці, що L і J . Значення функції дорівнює одиниці за дотримання обох меж.

Сполучення двох нерівностей у (2.10) означає, що запас за одним параметром не компенсує перевищення іншого. За затримки, меншої від граничної, та джиттеру, більшого від допустимого, оцінка дорівнює нулю. Так само вона обнуляється за надмірної затримки при малому джиттері. Рівність параметра відповідній межі належить допустимій області. Цю домовленість слід однаково застосовувати в математичній моделі, програмному коді та граничних контрольних прикладах.

Вплив мережі на дані враховує частку доставлених пакетів та відносну пропускну здатність:

$$f_{N,D}(t) = \left[(1 - p_{\text{loss}}(t)) \frac{B(t)}{B_{\text{req}}} \right]_0^1 \quad (2.11)$$

p_{loss} є часткою втрачених пакетів на інтервалі $[0; 1]$. Відношення B/B_{req} безрозмірне.

У формулі (2.11) частку втрат подають числом від нуля до одиниці: двом відсоткам відповідає 0,02. За пропускну здатності, рівної потрібній, оцінка становить 0,98. За її зменшення удвічі та незмінної частки втрат оцінка становить 0,49. Розрахунок показує, що однакова частка доставлених пакетів може супроводжуватися різною забезпеченістю потоку даних пропускну здатністю. Перевірка цієї функції тому має змінювати обидва аргументи, а не лише втрати пакетів.

Вплив мережевої затримки на користувача оцінюється відносно граничної затримки L_{max} :

$$f_{N,U}(t) = \left[1 - \frac{L(t)}{L_{max}}\right]_0^1 \quad (2.12)$$

L і L_{max} задаються в мілісекундах, тому їх відношення безрозмірне.

Залежність (2.12) відрізняється від порогової умови (2.10). Якщо затримка становить половину граничної, користувацька оцінка дорівнює 0,5, хоча програмна порогова оцінка за допустимого джитеру ще дорівнює одиниці. При досягненні граничної затримки користувацька оцінка вже нульова. Це розмежовує запас якості взаємодії та факт дотримання технічної межі. Обидві величини залишаються в матриці, оскільки описують різні наслідки одного вимірюваного параметра.

У формулах (2.9)–(2.28) кожний розмірний параметр входить через відношення до величини тієї самої фізичної природи. Різниці температур діляться на допустимі різниці температур. Частки втрат і ймовірності є безрозмірними. Через це всі елементи матриці $F(t)$ мають значення на інтервалі $[0; 1]$ і можуть бути агреговані.

Вплив температури середовища на апаратне забезпечення задається порогом $T_{H,max}$:

$$f_{ENV,H}(t) = \begin{cases} 1, & T_{env}(t) \leq T_{H,max}, \\ 0, & T_{env}(t) > T_{H,max}, \end{cases} \quad (2.13)$$

T_{env} і $T_{H,max}$ вимірюються в градусах Цельсія.

Порогове подання (2.13) визначає допустимість теплових умов для апаратної групи без моделювання часової інерції нагрівання. Два стани з різними температурами нижче межі отримують однакову оцінку. З цієї функції не можна визначити час, що залишився до перегріву, або швидкість зростання температури. Для таких висновків потрібна температурна траєкторія чи динамічна теплова модель. У межах поточного оцінювання функція використовується як перевірка заданої умови в момент спостереження.

Вплив фонового шуму на програмну частину оцінюється відносно допустимого рівня N_{max} :

$$f_{ENV,S}(t) = \left[1 - \frac{N_{noise}(t)}{N_{max}} \right]_0^1 \quad (2.14)$$

N_{noise} і N_{max} задаються в однакових одиницях рівня шуму.

У залежності (2.14) рівень шуму виступає вхідною координатою прийнятої оцінної шкали. За значення, що дорівнює половині порогового рівня, функція повертає 0,5, а при досягненні порогу – нуль. Оскільки децибел є логарифмічним поданням, це відношення не слід тлумачити як частку акустичної потужності. Воно задає модельне зменшення оцінки відносно вибраної межі; фізичне тлумачення її величини залежить від калібрування за відповідним сценарієм взаємодії.

Вплив освітлення на якість даних сенсорів оцінюється відносно потрібної освітленості L_{opt} :

$$f_{ENV,D}(t) = \left[\frac{L_{light}(t)}{L_{opt}} \right]_0^1 \quad (2.15)$$

L_{light} і L_{opt} вимірюються в люксах.

Залежність від освітленості у (2.15) зростає до потрібного рівня, після чого насичується. Половині потрібної освітленості відповідає оцінка 0,5; потрібному та будь-якому більшому значенню – одиниця. Отже, модель відображає дефіцит освітлення, але окремо не штрафує надмірну освітленість чи засліплення сенсора. Ця особливість задає межу інтерпретації: однакове одиничне значення за різних великих освітленостей означає насичення функції, а не доведену тотожність умов оптичного відстеження.

Комфорт користувача визначається відхиленням температури середовища від комфортного значення T_{comf} :

$$f_{ENV,U}(t) = \left[1 - \frac{|T_{env}(t) - T_{comf}|}{\Delta T_{U,max}} \right]_0^1 \quad (2.16)$$

$\Delta T_{U,max}$ є допустимою різницею температур у градусах Цельсія.

Температурне відхилення у (2.16) розглядається відносно комфортного значення. У комфортній точці оцінка дорівнює одиниці; при абсолютному відхиленні, рівному половині допустимого інтервалу, – 0,5; при досягненні межі – нулю.

Однакові за величиною додатне та від'ємне відхилення мають однакову оцінку. Така симетрія є припущенням функціональної форми. Її придатність для конкретної групи користувачів визначається вибором комфортної температури й допустимого відхилення в профілі застосування.

Вплив інтенсивності рухів користувача на апаратне забезпечення оцінюється відносно граничної інтенсивності M_{max} :

Для початку, опишемо функцію, яка розраховує ймовірність нормального функціонування апаратних засобів під час неоднотипної поведінки користувача:

$$f_{B,H}(t) = \left[1 - \frac{M(t)}{M_{max}}\right]_0^1 \quad (2.17)$$

M і M_{max} задаються в однакових одиницях інтенсивності рухів.

Лінійна залежність (2.17) відображає зменшення модельного запасу апаратної групи зі зростанням інтенсивності рухів. Нульова інтенсивність дає одиничне значення, половина граничної – 0,5, а досягнення або перевищення межі – нуль. Вибраний показник інтенсивності має залишатися однаково визначеним у всіх порівнюваних сеансах. Заміна частоти рухів іншою характеристикою без переналаштування граничного значення змінила б зміст вхідної величини, хоча програмний тип міг би залишитися тим самим.

Спроможність програмного забезпечення обробляти взаємодії задається порогом частоти подій F_{max} :

$$f_{B,S}(t) = \begin{cases} 1, & F(t) \leq F_{max}, \\ 0, & F(t) > F_{max}, \end{cases} \quad (2.18)$$

F є частотою вхідних подій, F_{max} є максимально допустимою частотою в тих самих одиницях. Це позначення не пов'язане з комфортною розбіжністю рухів.

Функція (2.18) перевіряє достатність пропускнує спроможності програмного обробника подій. Частота на самій межі є допустимою, а найменше додатне перевищення переводить оцінку в нуль. У такій постановці рівень нижче межі не розрізняється за величиною запасу. Для оцінювання черги або часу очікування потрібні додаткові залежності, яких порогова функція не містить. Тому її результат

використовується як ознака виконання обмеження, спільно з неперервними оцінками інших взаємодій.

Вплив обсягу користувацьких даних оцінюється відношенням максимально оброблюваного обсягу V_{max} до фактичного обсягу V :

$$f_{B,D}(t) = \left[\frac{V_{max}}{V(t)} \right]_0^1, \quad V(t) > 0 \quad (2.19)$$

V і V_{max} задаються в однакових одиницях обсягу даних.

Для додатного обсягу даних, що не перевищує допустимого, функція (2.19) насичується на одиниці. При подвоєнні допустимого обсягу оцінка дорівнює 0,5, а при потроєнні – приблизно 0,333. Залежність відрізняється від лінійного штрафу: після перевищення межі вона зменшується обернено пропорційно обсягу. Нульовий обсяг потребує явної граничної домовленості, оскільки безпосереднє ділення на нього не визначене. У програмних перевірках додатка В для відсутності такого навантаження використано одиничну оцінку.

Вплив рухів на комфорт користувача оцінюється відносно комфортної інтенсивності M_{conf} :

$$f_{B,U}(t) = \left[1 - \frac{M(t)}{M_{conf}} \right]_0^1 \quad (2.20)$$

M і M_{conf} мають однакову фізичну розмірність.

Один параметр руху використовується у (2.17) і (2.20), але нормується на різні межі. Якщо комфортна інтенсивність менша від апаратної граничної, користувацька оцінка зменшується швидше. За базових меж 20 і 30 одиниць та інтенсивності 10 одиниць користувацька оцінка становить 0,5, а апаратна – приблизно 0,667. Цей аналітичний приклад показує, чому задовільний стан технічного компонента не є достатньою умовою комфортної взаємодії користувача.

Вплив системного навантаження на апаратне забезпечення оцінюється відносно граничного навантаження $C_{H,max}$:

$$f_{S,H}(t) = \left[1 - \frac{C_{load}(t)}{C_{H,max}} \right]_0^1 \quad (2.21)$$

C_{load} і $C_{H,max}$ задані в однаковій нормованій шкалі завантаження.

У формулі (2.21) навантаження та його межа мають належати одній шкалі. Значення 50 % у шкалі часток подається як 0,5. Якщо граничне навантаження дорівнює одиниці, оцінка становить 0,5; введення числа 50 замість 0,5 призвело б до штучного обнулення після обмеження інтервалом. Виявляти таку помилку потрібно до обчислення, оскільки результат, обмежений інтервалом $[0; 1]$, сам по собі не засвідчує правильності одиниць входу.

Вплив навантаження на програмне забезпечення задається порогом $C_{S,max}$:

$$f_{S,S}(t) = \begin{cases} 1, & C_{load}(t) \leq C_{S,max}, \\ 0, & C_{load}(t) > C_{S,max}, \end{cases} \quad (2.22)$$

$C_{S,max}$ є максимально допустимим навантаженням, за якого програмне забезпечення підтримує задану швидкодію.

Неперервна апаратна оцінка (2.21) та порогова програмна оцінка (2.22) виконують різні діагностичні ролі. На рівні граничного програмного навантаження порогова функція ще дорівнює одиниці. Натомість лінійні функції можуть уже показувати відсутність запасу. Для вибору коригувальної дії доцільно аналізувати весь рядок, а не обирати одну з цих оцінок як універсальну. Відмінність між ними впливає з функціональної форми й не є розбіжністю обчислювальних реалізацій.

Вплив навантаження на обробку даних залежить від відносного обсягу даних і відносного навантаження:

$$f_{S,D}(t) = \left[1 - \frac{D_{size}(t)}{D_{max}} \frac{C_{load}(t)}{C_{D,max}} \right]_0^1 \quad (2.23)$$

D_{size}/D_{max} та $C_{load}/C_{D,max}$ є безрозмірними відношеннями.

У функції (2.23) штраф визначається добутком двох відносних навантажень. Якщо обсяг даних становить половину допустимого, а системне навантаження – половину своєї межі, добуток дорівнює 0,25 і оцінка становить 0,75. За збільшення обох відношень до одиниці оцінка стає нульовою. Отже, зміна одного аргументу має більший вплив за більшого значення другого. Саме всередині цієї оцінки подано спільний ефект обсягу даних і навантаження.

Вплив навантаження на користувача оцінюється відносно межі $C_{U,max}$, після якої затримка взаємодії стає неприйнятною:

$$f_{S,U}(t) = \left[1 - \frac{C_{load}(t)}{C_{U,max}} \right]_0^1 \quad (2.24)$$

C_{load} і $C_{U,max}$ використовують одну шкалу завантаження.

Параметр навантаження входить до чотирьох оцінок рядка, але межі апаратного, програмного, інформаційного та користувацького контурів можуть відрізнятися. Їх рівність у базовому профілі є параметризацією конкретного прикладу. Для іншої платформи допускається задати різні межі, зберігши функціональні залежності. При порівнянні конфігурацій ці межі мають бути зафіксовані, інакше зміна оцінки одночасно відображатиме зміну фактичного навантаження та зміну вимоги до нього.

Апаратна працездатність зберігається за одночасного дотримання температурної межі та мінімального залишкового ресурсу акумулятора:

$$f_{HW,H}(t) = \begin{cases} 1, & T_{hw}(t) \leq T_{H,max} \wedge b_{life}(t) \geq b_{min}, \\ 0, & \text{інакше,} \end{cases} \quad (2.25)$$

T_{hw} і $T_{H,max}$ задаються в градусах Цельсія. b_{life} і b_{min} є частками залишкового ресурсу.

Умова (2.25) поєднує теплову допустимість і достатність резерву живлення. Навіть за низької температури оцінка дорівнює нулю, якщо ресурс акумулятора менший від мінімального. Так само достатній заряд не компенсує порушення температурної межі. Значення на обох межах належать допустимій області. Оскільки перевірка є миттєвою, вона не прогнозує тривалість подальшої роботи батареї; такий прогноз потребував би моделі розряду та відомого режиму споживання.

Вплив апаратного обмеження потужності на програмне забезпечення оцінюється відносно граничної потужності P_{max} :

$$f_{HW,S}(t) = \left[1 - \frac{P(t)}{P_{max}} \right]_0^1 \quad (2.26)$$

P і P_{max} вимірюються у ватах. Температурний вплив у цій функції не дублюється, оскільки його враховано у формулах (2.25), (2.27) та (2.28).

Для залежності (2.26) нульове споживання дає одиничну модельну оцінку, а споживання на граничному рівні – нульову. Це виражає прийнятий штраф за використання енергетичного ресурсу й не означає, що фізична XR-система виконує свої функції за відсутності живлення. Тлумачення результату потребує одночасного врахування працездатності інших компонентів. Зазначений приклад ілюструє відмінність між математичною оцінкою окремого впливу та повним описом фізично можливого робочого стану.

Вплив температури та обсягу даних на інформаційну підсистему задається мінімумом двох безрозмірних оцінок:

$$f_{HW,D}(t) = \min \left\{ \left[\frac{T_{crit} - T_{hw}(t)}{T_{crit} - T_{safe}} \right]_0^1, \left[\frac{D_{max}}{D_{size}(t)} \right]_0^1 \right\} \quad (2.27)$$

У температурному відношенні чисельник і знаменник є різницями температур. У другому відношенні D_{max} і D_{size} мають однакові одиниці.

Мінімум у формулі (2.27) виділяє сильніше з двох обмежень. За температурної оцінки 0,5 і достатнього ресурсу для даних результат дорівнює 0,5. Якщо обсяг даних удвічі перевищує допустимий, його оцінка теж становить 0,5, і результат залишається тим самим. Покращення лише слабшого обмеження не змінює мінімум, доки воно не стане визначальним. Для температурної складової обов'язковою є нерівність критичної та безпечної температур: знаменник має бути додатним.

Вплив температури апаратури на користувача оцінюється відхиленням від комфортної температури T_{comf} :

$$f_{HW,U}(t) = \left[1 - \frac{|T_{hw}(t) - T_{comf}|}{\Delta T_{HW,max}} \right]_0^1 \quad (2.28)$$

$\Delta T_{HW,max}$ є допустимою різницею температур для безпечної та комфортної взаємодії.

У функції (2.28) температура апаратури порівнюється з комфортною температурою через окремий допустимий інтервал. Ця оцінка не тотожна двійковій

працездатності апаратної групи. Пристрій може ще відповідати технічній температурній межі, але вже мати низьку модельну оцінку комфорту. Застосування різних функцій до одного вимірювання забезпечує розрізнення цих наслідків без введення дубльованих каналів телеметрії.

Матриця взаємодій і простір вимірювань мають різну розмірність. Двадцять елементів матриці не означають двадцять незалежних сенсорів або двадцять незалежних випадкових подій. Один вимірюваний параметр може входити до кількох функцій; частина полів вхідного профілю є фіксованими вимогами. Тому до початку розрахунку потрібно відокремити фактичні спостереження від нормувальних і порогових величин. Перші описують стан, другі визначають шкалу, за якою цей стан оцінюється.

Під час порівняння двох відліків зміна фактичних параметрів за незмінного профілю має однозначний зміст: вона відображає зміну оцінюваного стану. Якщо одночасно змінено допустимі межі, відмінність результатів може бути спричинена лише новою шкалою. Наприклад, за сталої затримки збільшення її допустимої межі підвищує користувацьку оцінку, хоча мережа не прискорилася. Через це профіль вимог є складовою вихідних даних кожного порівняльного розрахунку.

Для обчислення матриці використовують узгоджений набір параметрів одного моменту або одного визначеного інтервалу спостереження. Комбінування давнього значення температури з поточним навантаженням і затримкою, усередненою за інший інтервал, створює стан, який міг не існувати в жодний момент часу. Математичні функції обчислюють для нього коректні числа, але їх фізичне тлумачення буде обмеженим. Тому часові мітки та тривалість усереднення мають супроводжувати підготовку вхідного вектора.

Так само потрібно розрізняти відсутність даних і нульове значення фізичного параметра. Відсутній пакет телеметрії не засвідчує нульову затримку, нульову температуру або відсутність навантаження. Автоматична заміна пропуску нулем змінює значення функцій у різних напрямках і може штучно підвищити оцінку. Для практичного застосування профіль оброблення даних має визначати допустимий вік

спостереження та правило відхилення неповного вектора. Це правило передує математичному оцінюванню і не змінює формул матриці.

2.2 Метод оцінювання стану VR-системи в умовах динамічно змінюваного середовища

2.2.1 Інтегральний показник резильєнтності системи

На основі визначених параметрів розроблено інтегральний метод оцінювання резильєнтності системи віртуальної реальності. Такий метод дозволяє комплексно аналізувати вплив множинних зовнішніх факторів на ключові компоненти системи: апаратне забезпечення, програмне забезпечення, дані та користувачів.

Метод оцінювання стану VR-системи призначений для перетворення сукупності різнорідних параметрів поточного стану системи та середовища у єдиний інтегральний показник резильєнтності з можливістю визначення взаємодій «фактор—компонент», що найбільше впливають на деградацію системи. Метод передбачає виконання таких кроків:

1. Формування профілю місії XR-системи. Визначаються умови та сценарій експлуатації системи, критичні функції, нормувальні й порогові значення параметрів, а також допустимі межі їх зміни. Профіль місії визначає параметризацію оцінювальних функцій та відносну значущість факторів впливу для конкретного застосування.

2. Збирання поточних параметрів стану системи. Із засобів телеметрії отримуються значення параметрів, що характеризують стан мережі, умови навколишнього середовища, поведінку користувача, системне навантаження та апаратні обмеження. Сукупність отриманих параметрів формує вхідні дані для поточного кроку оцінювання.

3. Нормування параметрів та обчислення оцінювальних функцій. Виміряні фізичні параметри перетворюються у безрозмірні значення на інтервалі $[0;1]$ відповідно до нормувальних і порогових параметрів, установлених профілем місії.

Для кожної пари «фактор–компонент» обчислюється відповідна оцінювальна функція $f f_{i,j}(t)$. У результаті формується матриця, що містить двадцять оцінок впливу п'яти груп факторів на чотири компоненти XR-системи.

4. Визначення вагових коефіцієнтів взаємодій «фактор–компонент». Для кожного компонента формується нечітка матриця парних порівнянь факторів впливу. Після її нормалізації, агрегування нечітких оцінок, дефазифікації та нормалізації отримується матриця вагових коефіцієнтів W , сума елементів якої дорівнює одиниці. Вагові коефіцієнти визначаються під час початкового налаштування методу та перераховуються у разі зміни профілю місії або вихідних експертних оцінок.

5. Обчислення інтегрального показника резильєнтності. Поточний стан XR-системи визначається шляхом агрегування елементів матриці оцінювальних функцій з урахуванням відповідних елементів вагової матриці W . Інтегральний показник $R_{VR}(t)$ обчислюється як зважена сума двадцяти взаємодій «фактор–компонент» і набуває значень на інтервалі $[0;1]$, де значення, близьке до одиниці, відповідає номінальному стану системи, а його зниження характеризує ступінь деградації.

6. Визначення ступеня та джерела деградації. Отримане значення інтегрального показника порівнюється з допустимими значеннями, заданими профілем місії. Одночасно аналізуються окремі елементи матриці з урахуванням їх вагових коефіцієнтів, що дозволяє визначити пари «фактор–компонент» із найбільшим внеском у дефіцит резильєнтності та локалізувати критичні взаємодії.

7. Формування результату оцінювання. Результатом методу є поточне значення інтегрального показника резильєнтності $R_{VR}(t)$, матриця оцінок взаємодій та перелік критичних пар «фактор–компонент». Отримані результати використовуються для моніторингу стану XR-системи та як вхідні дані для подальшого синтезу мітигаційних стратегій.

Резильєнтність VR-системи визначається як її здатність підтримувати оптимальну продуктивність і якість користувацького досвіду, незважаючи на можливі збої чи впливи, спричинені цими зовнішніми факторами. Важливість інтегрального підходу полягає в тому, що він враховує сукупний вплив усіх

релевантних чинників, дозволяючи отримати єдину метрику, яка відображає загальну резильєнтність системи.

Формула інтегральної оцінки резильєнтності консолідує вплив усіх релевантних факторів у єдиний показник. Цей показник розраховується на основі індивідуальних функцій впливу, які були визначені раніше, та кількісно описують, як зовнішні фактори, такі як мережеві умови, екологічні чинники, поведінка користувачів, системне навантаження й апаратні обмеження, впливають на кожен компонент системи.

Результуюча оцінка резильєнтності обчислюється як зважена сума впливів на окремі компоненти системи, де вагові коефіцієнти відображають відносну важливість кожного компонента для загальної резильєнтності системи.

Вагові коефіцієнти дають змогу оцінити поточний стан системи і визначити пари «фактор, компонент», що найбільше впливають на деградацію.

Для обчислення інтегрального показника впливу кожного фактора на кожен компонент системи призначається ваговий коефіцієнт від 0 до 1, що характеризує відносну значущість відповідної взаємодії. Сума всіх вагових коефіцієнтів дорівнює 1.

Матрицю вагових коефіцієнтів подамо у вигляді:

$$\mathbf{W} = \begin{bmatrix} w_{N,H} & w_{N,S} & w_{N,D} & w_{N,U} \\ w_{ENV,H} & w_{ENV,S} & w_{ENV,D} & w_{ENV,U} \\ w_{B,H} & w_{B,S} & w_{B,D} & w_{B,U} \\ w_{S,H} & w_{S,S} & w_{S,D} & w_{S,U} \\ w_{HW,H} & w_{HW,S} & w_{HW,D} & w_{HW,U} \end{bmatrix} \quad (2.29)$$

де $w_{i,j}$ – ваговий коефіцієнт впливу фактору i на компонент j . Методи визначення кожного з вагових коефіцієнтів будуть розглянуті в наступних розділах.

Сфера застосування системи безпосередньо впливає на параметризацію методу. Галузевий контекст визначає експертні матриці парних порівнянь і, як наслідок, вагову матрицю $\mathbf{W}$, а також граничні значення в оцінювальних функціях: для медичних і тренажерних застосувань підвищену вагу отримують безпека користувача та цілісність даних із жорсткішими порогами затримки, для промислового проектування – стабільність продуктивності й точність позиційного відстеження, для

розважальних платформ – плавність зображення за помірних вимог до цілісності. Отже, той самий математичний апарат налаштовується під конкретну сферу застосування без зміни структури моделі.

Усі ваги є безрозмірними, невід’ємними та нормованими так, щоб їх сума дорівнювала одиниці.

Інтегральний показник поточного стану XR-системи визначається поелементним добутком матриць W і $F(t)$ з підсумовуванням усіх двадцяти елементів:

$$R_{VR}(t) = \sum_{i=1}^5 \sum_{j=1}^4 w_{i,j} f_{i,j}(t), \quad w_{i,j} \geq 0, \quad \sum_{i=1}^5 \sum_{j=1}^4 w_{i,j} = 1 \quad (2.30)$$

де R_{VR} – загальний показник резильєнтності системи, w_i – ваговий коефіцієнт, визначений для кожного з факторів впливу на компонент, що оцінюється функцією $f_{i,j}$, і показує відносну важливість даного впливу в контексті резильєнтності системи віртуальної або доповненої реальності, $f_{i,j}$ – індивідуальні функції оцінки впливу, визначені раніше, які відповідають взаємодії між зовнішніми впливами та компонентами системи. Ці функції дозволяють кількісно оцінити, як конкретні зовнішні чинники, такі як умови навколишнього середовища, мережеві параметри або поведінка користувачів, впливають на апаратне забезпечення, програмне забезпечення, дані чи користувачів.

$R_{VR}(t)$ є безрозмірною величиною на інтервалі $[0; 1]$. Нуль відповідає повній втраті оцінюваних функцій, одиниця відповідає номінальному стану за заданими базовими значеннями.

Нормування ваг надає інтегральному показнику зміст опуклої комбінації двадцяти оцінок. Оскільки кожна оцінка невід’ємна, їх сума з невід’ємними вагами також невід’ємна. Оскільки жодна оцінка не перевищує одиниці, зважена сума не перевищує суми ваг, тобто одиниці. Для цього висновку не потрібні припущення про статистичну незалежність факторів: він безпосередньо випливає з меж значень та умови нормування у (2.30).

Сильніше твердження полягає в тому, що результат перебуває між найменшою та найбільшою оцінками тих взаємодій, які мають додатну вагу. Якщо всі двадцять оцінок однакові й дорівнюють певному числу, інтегральний показник дорівнює саме

цьому числу незалежно від розподілу нормованих ваг. Ця властивість дає просту контрольну умову для реалізації: рівномірну матрицю оцінок агрегування не повинно змінювати.

Для аналізу зручно розрізняти внесок у поточний показник та внесок у його дефіцит. Перший утворюється множенням ваги на оцінку взаємодії. Другий утворюється множенням тієї самої ваги на різницю між одиницею та оцінкою. Сума всіх дефіцитних внесків дорівнює різниці між одиницею та інтегральним показником. Це дозволяє перевірити повноту діагностичної декомпозиції: її складові мають відновлювати загальний дефіцит без додаткового масштабування.

Розглянемо умовну пару з вагою 0,10 та оцінкою 0,20. Її внесок у поточний показник становить 0,02, а внесок у дефіцит – 0,08. Інша пара з вагою 0,02 та нульовою оцінкою має дефіцит 0,02. Отже, найменша оцінка не обов'язково визначає найбільшу втрату інтегрального показника. Для локалізації використовується саме зважений дефіцит, що узгоджується з правилом (2.40). Наведені числа ілюструють арифметичну властивість агрегування.

Підсумовування внесків за рядками показує, яка група впливів формує поточний дефіцит. Підсумовування за стовпцями показує його розподіл між компонентами. Обидва способи дають однакову загальну суму, але відповідають різним діагностичним питанням. Рядок мережі може бути визначальним для пошуку джерела, тоді як стовпець користувача – для аналізу наслідків. Такий перегляд не потребує повторного обчислення оцінювальних функцій або зміни ваг.

Верхня межа можливого покращення однієї взаємодії також відома до синтезу мітигації: її оцінка не може перевищити одиницю. Отже, найбільший приріст інтегрального показника від покращення тільки цієї взаємодії дорівнює її поточному зваженому дефіциту. Якщо ж коригувальна дія змінює кілька параметрів, приріст потрібно обчислювати за всією матрицею. Сума окремих очікуваних ефектів без повторного оцінювання може бути некоректною через спільні аргументи функцій і насичення.

Однаковий інтегральний показник не визначає єдину матрицю стану. Наприклад, за однакових ваг двох взаємодій оцінки 1 і 0 дають ту саму суму, що

оцінки 0,5 і 0,5. При цьому діагностична картина суттєво різна: у першому випадку одна взаємодія повністю деградована, у другому обидві мають часткову деградацію. Тому разом зі скалярним результатом зберігають матрицю оцінок та карту внесків. Саме вони надають дані для змістового вибору мітигації.

Інтегральний показник не замінює незалежних обмежень критичних функцій. Високе середнє значення може співіснувати з неприпустимим станом взаємодії, що має малу вагу. Збільшення ваги підсилює її вплив на середнє, але не перетворює усереднення на логічну перевірку всіх вимог. Відтак порівняння з порогом загального показника доповнюється умовами профілю місії. Це забезпечує узгодження оцінювання стану з визначенням резильєнтності у розділі 1.

Декомпозиція також дозволяє встановити межі оцінки за неповної інформації. Якщо для частини взаємодій відомі точні оцінки, а для решти відомо лише належність інтервалу $[0; 1]$, зважена сума відомих внесків задає нижню межу. Верхня межа утворюється додаванням сумарної ваги невідомих взаємодій. Обидві межі впливають з невід'ємності ваг і не вимагають припущення про розподіл невідомих значень. Такий інтервал характеризує невизначеність розрахунку, а не фізичний прогноз стану.

Наприклад, нехай відомі внески разом становлять 0,70, а сумарна вага взаємодій із невідомими оцінками дорівнює 0,10. Тоді інтегральний показник перебуває в межах від 0,70 до 0,80. Вибір середини цього інтервалу як єдиного результату потребував би додаткової домовленості щодо невідомих оцінок. Без такої домовленості коректно зберегти інтервал або позначити результат неповним. Якщо ж один відсутній фізичний параметр впливає на кілька взаємодій, потрібно врахувати ваги всіх цих взаємодій.

Для кожної взаємодії можна також задати власний інтервал допустимих оцінок. Завдяки додатності ваг нижня межа інтегрального показника утворюється з нижніх меж усіх оцінок, а верхня – з верхніх. Якщо оцінки пов'язані спільними аргументами, отриманий інтервал може бути ширшим за фактично досяжний: його побудова допускає незалежне поєднання крайніх значень. Однак він залишається зовнішнім

обмеженням і може застосовуватися для консервативної перевірки порогового рішення.

Якщо весь інтервал оцінки перебуває нижче порогу виявлення деградації, рішення не залежить від невідомих значень у заданих межах. Якщо весь інтервал перебуває вище порогу, висновок щодо цього скалярного критерію також визначений. Перетин інтервалом порогу означає, що наявних даних недостатньо для однозначного рішення. Це не скасовує незалежних перевірок критичних функцій, але дозволяє явно відокремити невизначеність оцінювання від підтвердженої деградації.

Уточнення спостережень доцільно спрямовувати насамперед на взаємодії з найбільшим добутком ваги на ширину інтервалу оцінки. Саме цей добуток визначає внесок взаємодії у ширину загального інтервалу. Повне уточнення маловагової взаємодії з вузьким інтервалом може майже не змінити певність рішення. Наведене правило є аналітичним наслідком аддитивної структури та описує можливий спосіб пріоритизації перевірки даних; окремий адаптивний модуль збору телеметрії в цій роботі ним не заявляється.

2.2.2 Визначення вагових коефіцієнтів методом нечіткого аналізу ієрархій

Процес верифікації резильєнтності складних технічних систем, таких як XR-платформи, нерозривно пов'язаний із необхідністю врахування суб'єктивної важливості різних чинників впливу. Оскільки вплив мережевих затримок може бути критичнішим для стабільності системи, ніж зміна рівня освітлення, виникає потреба в точному визначенні вагових коефіцієнтів $w_{i,j}$.

Запропонований метод поєднує нечіткий аналіз ієрархій із функціональною структурою оцінки резильєнтності систем віртуальної реальності. На відміну від існуючих рішень, які обмежуються оцінкою важливості факторів у статичних умовах, запропонований метод дозволяє інтегрувати нечіткі експертні знання безпосередньо у процес моделювання впливів у реальному часі.

Для формалізації лінгвістичних оцінок експертів використовуються трикутні нечіткі числа. Кожне таке число позначається як $a_{ij} = (l_{ij}, m_{ij}, u_{ij})$, де l – нижня межа, m – найбільш ймовірне значення, а u – верхня межа оцінки.

Першим етапом методу є побудова нечіткої матриці парних порівнянь. Метою цього етапу є формалізація експертних оцінок відносної важливості факторів впливу на компоненти VR-системи. Для кожного компонента системи формується матриця попарних порівнянь між факторами впливу, де елементи представлені у вигляді трикутних нечітких чисел:

$$\tilde{a}_{i,j} = (l_{i,j}, m_{i,j}, u_{i,j}) \quad (2.31)$$

де l_{ij}, m_{ij}, u_{ij} – мінімальне, найбільш ймовірне та максимальне значення оцінки відповідно. У випадку, коли фактор i менш важливий за фактор j , використовується обернене нечітке число: Множина S , використана вище для позначення сукупності компонентів, тотожна кортежу VR, введеному формулою (2.6), і вживається лише для зручності індексації компонентів.

$$\tilde{a}_{j,i} = \left(\frac{1}{u_{i,j}}, \frac{1}{m_{i,j}}, \frac{1}{l_{i,j}} \right) \quad (2.32)$$

На другому етапі матриця повинна бути нормалізована задля уніфікації шкали оцінок для забезпечення порівнюваності та підготовки для обчислення вагових коефіцієнтів. Обчислюються суми елементів по кожному стовпцю: $\sum l_j, \sum m_j, \sum u_j$. Після чого кожен елемент нормалізується за формулою:

$$\tilde{n}_{i,j} = \left(\frac{l_{i,j}}{\sum_{k=1}^n u_{k,j}}, \frac{m_{i,j}}{\sum_{k=1}^n m_{k,j}}, \frac{u_{i,j}}{\sum_{k=1}^n l_{k,j}} \right) \quad (2.33)$$

При нормалізації важливо зберегти монотонність і відношення невизначеності між елементами. У трикутному нечіткому числі ліве значення – найоптимістичніше, праве – найпесимістичніше. Відповідно, ліві межі слід нормалізувати на найбільші можливі значення, щоб оцінка не була занадто завищеною, а праві – на найменші, щоб оцінка не була заниженою.

Для визначення агрегованої ваги кожного фактору впливу на компонент системи здійснюється обчислення нечітких вагових коефіцієнтів. Для цього для кожного фактору оцінюється середнє значення нормалізованих оцінок:

$$w_i = \left(\frac{1}{n} \sum_{j=1}^n l_{i,j}, \frac{1}{n} \sum_{j=1}^n m_{i,j}, \frac{1}{n} \sum_{j=1}^n u_{i,j} \right) \quad (2.34)$$

У формулі (2.34) l_{ij} , m_{ij} та u_{ij} позначають відповідно нижню, модальну та верхню координати нормованого нечіткого числа $\tilde{n}_{ij}$, отриманого за (2.33).

На наступному етапі необхідно здійснити дефазифікацію (знаходження звичайного чіткого значення кожної з ваг) та нормалізувати їх так, щоб їх сума дорівнювала одиниці:

$$w_i^{\text{def}} = \frac{l_i + m_i + u_i}{3} \quad (2.35)$$

$$w_i^{\text{norm}} = \frac{w_i^{\text{def}}}{\sum_{k=1}^n w_k^{\text{def}}} \quad (2.36)$$

Важливим є те, що в межах нечіткої логіки допускаються як симетричні, так і асиметричні трикутні оцінки. Центр трикутника m може бути зміщеним відносно l та u , що дозволяє моделювати різні рівні впевненості в оцінках. Це суттєво підвищує точність представлення експертного знання в умовах невизначеності.

Запропонований метод визначення вагових коефіцієнтів на основі нечіткого аналізу ієрархій забезпечує формалізоване включення експертних знань у модель оцінювання резильєнтності систем віртуальної реальності. На відміну від класичних підходів, він дозволяє враховувати невизначеність, варіативність упевненості суджень, а також динамічний характер впливів, характерний для високотехнологічних VR-середовищ.

Інтеграція нечітких вагових коефіцієнтів у структуру інтегральної функції резильєнтності дає змогу врахувати невизначеність експертних суджень під час визначення відносної значущості взаємодій «фактор–компонент». Метод підтримує симетричні та асиметричні трикутні оцінки й може бути застосований у різних профілях експлуатації без зміни структури інтегрального показника.

Матриці парних порівнянь можуть бути симетричними за модальними оцінками або асиметричними за шириною інтервалів невизначеності. Обернені елементи визначаються формулою (2.32).

Запропонований метод визначення вагових коефіцієнтів на основі нечіткого аналізу ієрархій формалізує експертні судження у вигляді числових ваг із урахуванням заданих інтервалів невизначеності. На відміну від класичного АНР із чіткими оцінками за шкалою Сааті, нечітке представлення дає змогу задавати діапазон можливих значень відносної важливості. Після агрегації та дефазифікації отримуються нормовані ваги, які використовуються в інтегральному показнику резильєнтності. Такий підхід не усуває суб'єктивність вихідних оцінок, але робить процедуру їх перетворення на вагові коефіцієнти формалізованою та відтворюваною.

Зауважимо відмінність двох споріднених позначень. Інтегральний показник резильєнтності з формули (2.30) характеризує поточний стан системи в момент часу t і обчислюється за миттєвими значеннями оцінювальних функцій. Натомість підсумкова функція резильєнтності, згадана вище, є агрегованою оцінкою архітектури за весь інтервал місії для заданого вектора проектних параметрів та профілю зовнішніх впливів і використовується як цільова функція задачі синтезу, що формулюється у розділі 3.

Математична формалізація цього процесу передбачає побудову матриці нечітких парних порівнянь $A = [a_{ij}]$, де кожен елемент $a_{ij} = (l_{ij}, m_{ij}, u_{ij})$ відображає відносну важливість i -го фактора порівняно з j -м. Завдяки використанню асиметричних трикутних нечітких чисел, модель отримує здатність до «тонкого налаштування» під конкретні сценарії експлуатації: при моделюванні медичних VR-тренажерів вага фактора точності позиціонування може бути зміщена в бік максимальної впевненості, тоді як для ігрових додатків цей же параметр може мати ширший діапазон невизначеності. Такий метод придатний до повної автоматизації, де вагові коефіцієнти зможуть динамічно оновлюватися при зміні вхідного профілю навантаження, що забезпечує безперервну верифікацію стану резильєнтності системи протягом усього життєвого циклу симуляції.

У результаті застосування даного методу формується фінальна матриця вагових коефіцієнтів W , яка слугує операційним базисом для обчислення системних метрик M_k . Кожен елемент цієї матриці $w_{i,j}$ характеризує нормалізовану вагу впливу i -го зовнішнього фактора з множини E на j -й компонент архітектури з множини $C = \{H, S, D, U\}$.

Метод враховує технічні обмеження та пріоритетність функцій для конкретного профілю місії.

Отримані ваги використовуються для оцінювання резильєнтності та визначення найбільш критичних взаємодій.

Отримані ваги відображають відносну значущість факторів для заданого профілю місії та використовуються під час оцінювання резильєнтності й локалізації критичних взаємодій. Узгодженість модального зрізу матриці парних порівнянь перевіряється за відношенням узгодженості. У наведеному нижче числовому прикладі значення $CR = 0,014$ не перевищує прийнятого порогу $0,1$, тому вихідні судження для цієї матриці вважаються прийнятно узгодженими.

У результаті застосування запропонованого методу формується матриця вагових коефіцієнтів впливів факторів на компоненти системи віртуальної реальності. Кожен елемент цієї матриці $w_{i,j}$ характеризує нормалізовану вагу впливу i -го фактору з множини E на j -тий компонент з множини VR , враховуючи невизначеність експертних оцінок і логіку нечітких парних порівнянь

Отриману матрицю можна представити у вигляді:

$$W = \frac{1}{4} [w_{i,j}^{\text{norm}}]_{5 \times 4} \quad (2.37)$$

Множник $1/4$ відповідає рівній сумарній вазі чотирьох компонентів: кожен стовпець до глобальної нормалізації має одиничну суму, а після неї сума всіх двадцяти елементів W дорівнює одиниці.

Проілюструємо процедуру (2.31)–(2.36) повним числовим прикладом для компонента апаратного забезпечення. Порівнюються п'ять факторів у такому порядку: стан мережі, середовище, поведінка користувача, системне навантаження,

апаратні обмеження. Експертні базові оцінки за шкалою Сааті зведено у верхній трикутник матриці парних порівнянь: перевага стану мережі над середовищем оцінена величиною 2, над поведінкою користувача – 3, над навантаженням – 6, над апаратними обмеженнями – 8; перевага середовища над цими самими факторами – 3, 5 та 7 відповідно; поведінки користувача над навантаженням та обмеженнями – 2 та 3; навантаження над обмеженнями – 2. Кожна базова оцінка b подається трикутним нечітким числом із межами $b-1$, b , $b+1$, причому нижня межа не опускається нижче одиниці, діагональ складається з одиничних чисел, а симетричні елементи є оберненими нечіткими числами за (2.32).

Нечіткі суми стовпців, обчислені для нормалізації за (2.33), дорівнюють: для стану мережі 1,84; 2,13; 2,84, для середовища 2,54; 3,68; 4,92, для поведінки користувача 5,58; 7,83; 10,50, для навантаження 11,33; 14,50; 18,00, для апаратних обмежень 17,00; 21,00; 25,00. Після нормалізації, за якої нижні межі діляться на верхні суми, а верхні межі – на нижні суми, та усереднення рядків за (2.34) нечіткі вагові вектори становлять: стан мережі 0,261; 0,439; 0,718, середовище 0,195; 0,314; 0,531, поведінка користувача 0,074; 0,131; 0,230, навантаження 0,042; 0,072; 0,130, апаратні обмеження 0,029; 0,045; 0,076.

Дефазифікація центроїдним способом за (2.35) дає значення 0,472; 0,346; 0,145; 0,082; 0,050, які після нормалізації до одиничної суми за (2.36) утворюють вагові коефіцієнти факторів для апаратного компонента: 0,431; 0,316; 0,132; 0,075; 0,046. Відношення узгодженості модального зрізу цієї матриці становить 0,014 за припустимого порогу 0,1, що підтверджує несуперечливість вихідних суджень. Після аналогічних обчислень для решти трьох компонентів та глобальної нормалізації, за якої сума всіх двадцяти ваг дорівнює одиниці, отримані значення утворюють матрицю W ; робочі значення для реалізованого програмного засобу, обчислені цією самою процедурою з базових матриць порівнянь підрозділу 4.3.2, наведено в таблиці 4.4. Джерелом оцінок є узагальнені експертні судження щодо відносної критичності факторів для відповідного компонента, закладені у базові матриці порівнянь програмного засобу.

Запропонований метод дає змогу чисельно описати стан XR-системи та проаналізувати внесок кожного фактора у загальну деградацію.

Трикутне нечітке число у (2.31) задається трьома додатними координатами, впорядкованими від нижньої до верхньої. Додатність потрібна для обернення та нормування. На діагоналі використовуються одиничні числа, оскільки фактор порівнюється із самим собою. Значення нижнього трикутника визначаються через верхній за (2.32), тому незалежно задавати обидві половини матриці не потрібно. Такий порядок зменшує кількість незалежних суджень для однієї матриці 5×5 до десяти.

Обернення змінює порядок крайніх координат. Для умовної оцінки переваги (2; 3; 4) обернена оцінка становить (0,25; 0,333...; 0,5). Якщо інвертувати координати без перестановки нижньої та верхньої меж, результат уже не задовольнятиме їх упорядкованості. Важливо також, що обернене нечітке число загалом не має симетричної ширини відносно модальної координати, навіть якщо початкова оцінка була симетричною. Це природний наслідок нелінійності обернення.

У нормуванні (2.33) нижня координата ділиться на верхню суму стовпця, а верхня – на нижню. Для додатних упорядкованих вхідних координат такий вибір зберігає порядок нормованих координат. Наприклад, якщо сума стовпця становить (4; 6; 8), а його елемент – (1; 2; 3), нормоване число дорівнює (0,125; 0,333...; 0,75). Отриманий інтервал відображає застосоване правило обчислення; його не слід тлумачити як статистичний довірчий інтервал.

Після усереднення нормованих елементів рядка формується нечіткий ваговий вектор. Усереднення виконується окремо для кожної координати й не змішує нижні, модальні та верхні оцінки. Наступна дефазифікація перетворює кожний такий вектор на одне чітке число за (2.35). Проте сума цих чітких чисел не зобов'язана дорівнювати одиниці, тому крок (2.36) є самостійною необхідною операцією. Пропуск цього кроку змінив би масштаб інтегрального показника.

У наведеному числовому прикладі дефазифіковані оцінки приблизно дорівнюють 0,472; 0,346; 0,145; 0,082; 0,050. Їх сума становить близько 1,095. Ділення кожної оцінки на цю суму дає локальні ваги, наведені вище. Через округлення

проміжних чисел ручний розрахунок може відрізнятися в останньому десятковому знаку від результату програми, що зберігає повну числову точність. Округлення слід виконувати для подання результату, після завершення обчислювального ланцюжка.

Локальне та глобальне нормування виконують різні завдання. Локальний вектор визначає відносну важливість п'яти факторів для одного компонента. Чотири такі вектори разом мають суму, рівну чотирьом. За прийнятої рівної сумарної важливості компонентів множник у (2.37) зменшує кожний елемент у чотири рази. У результаті кожний стовпець глобальної матриці має суму 0,25, а сума всієї матриці дорівнює одиниці. Ці дві умови доцільно перевіряти окремо.

Рівність сум стовпців не означає рівності окремих ваг. У межах кожного компонента мережа, середовище та інші фактори можуть мати різну значущість. Водночас зміна локальних суджень за збереження прийнятого глобального нормування перерозподіляє важливість усередині стовпця, не змінюючи його сумарної частки. Для іншого розподілу важливості компонентів був би потрібний окремий рівень ваг, який не входить до використаної у роботі процедури.

Перевірка узгодженості модального зрізу та нечітка обробка суджень не є взаємозамінними. Відношення узгодженості характеризує логічну погодженість чітких центральних оцінок. Ширина нечітких чисел подає невизначеність заданої переваги. Матриця може мати мале відношення узгодженості й широкі інтервали або вузькі інтервали та суперечливі центральні судження. Тому у протоколі зберігають як вхідні трикутні числа, так і результат перевірки модального зрізу.

Процедура зважування має кілька контрольних випадків, незалежних від експертних переваг. Якщо всі парні порівняння дорівнюють одиниці, фактори рівнозначні: локальні ваги становлять 0,2, а після глобального нормування чотирьох компонентів кожна з двадцяти ваг дорівнює 0,05. Це дозволяє перевірити нормування та відсутність прихованої переваги окремих індексів.

Інший контрольний випадок — узгоджене переставлення рядків і стовпців матриці парних порівнянь. Локальний вектор ваг має переставитися відповідно, а агрегування з аналогічно зміненою матрицею оцінок — зберегти інтегральний показник. Це перевіряє коректність індексації незалежно від арифметичних операцій.

Порівняння нечіткої та чіткої процедур потребує однакової основи суджень. Якщо всі координати трикутних чисел збігаються, невизначеність зникає і нечіткі операції зводяться до точних. Для коректного зіставлення необхідно фіксувати вхідні матриці, спосіб нормування та правило агрегування.

Відношення узгодженості не замінює експертного обґрунтування переваг. Узгоджена матриця може відповідати іншому профілю застосування системи. Тому перевірка узгодженості виявляє внутрішні суперечності, але не визначає предметну коректність суджень, а результати зважування мають бути пов'язані з місією та функціями системи.

2.2.3 Обґрунтування порогових і нормувальних параметрів оцінювальних функцій

Оцінювальні функції (2.9)–(2.28) є безрозмірними, проте їх значення визначаються набором нормувальних і порогових параметрів, які мають фізичний зміст. Обґрунтованість інтегрального показника безпосередньо залежить від походження цих величин, тому їх джерела потребують окремого розгляду.

За способом побудови оцінювальні функції поділено на три типи. Порогові функції повертають нуль або одиницю залежно від того, чи перебуває параметр у допустимій області; вони описують впливи, для яких деградація має характер втрати виконуваності, наприклад перевищення критичної температури або порушення цілісності даних. Відносні функції повертають нормоване відношення поточного значення параметра до нормувального і описують впливи з поступовою деградацією, наприклад зростання завантаження обчислювача. Композитні функції утворюються добутком або мінімумом кількох відношень і описують впливи, дія яких проявляється лише за одночасного погіршення кількох параметрів.

Джерела нормувальних параметрів поділено на чотири групи. Першу утворюють вимоги профілю місії: необхідна пропускна здатність, бюджет затримки, необхідний рівень відновлення та граничний час відновлення; для імерсивних послуг відповідні орієнтовні значення унормовано у [130]. Другу утворюють характеристики

обладнання, зазначені в документації виробника: тепловий пакет, безпечний і критичний температурні пороги, гранична потужність, обсяг пам'яті та сховища. Третю утворюють нормативні вимоги до умов праці та безпеки контакту обладнання з тілом користувача, зокрема допустима температура поверхонь, доступних для дотику [97], та вимоги ергономіки взаємодії [118]. Четверту утворюють результати досліджень самопочуття користувачів імерсивних середовищ [91, 93, 94], з яких визначаються комфортна та гранична інтенсивність рухів і допустима розсинхронізація сенсорних каналів [90].

Нормувальні та порогові параметри, використані у роботі, а також джерела їх значень наведено у таблиці 2.1. Значення відповідають базовому профілю, застосованому в експериментальних сценаріях розділу 3; для інших профілів вони замінюються без зміни вигляду оцінювальних функцій.

Таблиця 2.1 – Нормувальні та порогові параметри оцінювальних функцій

Параметр	Зміст	Джерело значення	Базовий профіль
B_{req}	Необхідна пропускна здатність каналу подання	Вимоги профілю місії до потоку подання	100 Мбіт/с
L_{req}, L_{max}	Необхідна та гранично допустима затримка	Бюджет затримки «рух – фотон» профілю місії	50 мс; 100 мс
p_{loss}	Допустима частка втрачених пакетів	Вимоги до цілісності потоку даних	2 %
$T_{comf}, \Delta T_{U,max}$	Комфортна температура довкілля та допустиме відхилення	Нормативні вимоги до умов праці	25 °C; ± 10 °C

Закінчення таблиці 2.1

Параметр	Зміст	Джерело значення	Базовий профіль
N_{max}	Гранично допустимий рівень акустичного шуму	Нормативні вимоги до умов праці та розпізнавання звукових підказок	За профілем місії
L_{opt}	Оптимальна освітленість робочої зони	Документація системи оптичного відстеження	За документацією
M_{comf}, M_{max}	Комфортна та гранична інтенсивність рухів користувача	Дослідження самопочуття користувачів імерсивних середовищ	20 Гц; 30 Гц
$C_{H,max}, C_{S,max}, \dots$	Гранично допустиме завантаження компонентів	Характеристики обчислювальної платформи	1,0 (нормоване)
D_{max}	Гранично допустимий обсяг даних сеансу	Обсяг сховища та пропускна здатність тракту	8 ГБ
T_{safe}, T_{crit}	Безпечний та критичний тепловий пороги обладнання	Документація виробника обладнання	70 °C; 90 °C
P_{max}	Гранично допустима потужність споживання	Тепловий пакет пристрою	100 Вт

Наведений поділ дає змогу відокремити методичну частину моделі від параметричної. Вигляд оцінювальних функцій визначається фізикою відповідного впливу і залишається незмінним, тоді як числові значення параметрів є вхідними

даними, які уточнюються за документацією обладнання та вимогами застосування. Це є суттєвою умовою переносності методу: перехід до іншого класу обладнання або до іншого профілю місії не вимагає перебудови матриці оцінювальних функцій.

Профіль параметризації має утворювати узгоджену сукупність вимог. Для кожної нормувальної величини потрібно визначити фізичний зміст, одиницю, джерело та область застосування. Один і той самий числовий поріг не можна переносити між різними способами вимірювання лише через збіг назви параметра. Наприклад, середня затримка пакета, найбільша затримка за інтервал і затримка «рух – фотон» є різними величинами, навіть коли всі вони вимірюються в мілісекундах.

Нормувальний масштаб визначає швидкість зміни неперервної оцінки. Якщо параметр входить через відношення до граничного значення, збільшення цього значення зменшує модуль нахилу оцінної залежності у ненасиченій області. Через це переналаштування порогу одночасно змінює допустиму область і чутливість. Під час калібрування слід аналізувати обидва наслідки: прийнятний рівень оцінки за номінальних умов та її зміну за реально значущого погіршення умов.

Для порогових функцій важливо окремо зафіксувати домовленість щодо рівності. У формулах цього розділу досягнення верхньої допустимої межі ще належить прийнятній області; порушенням є її перевищення. Для мінімального ресурсу батареї, навпаки, порушенням є зменшення нижче межі. Контрольні значення потрібно вибирати з обох боків порогу та на самому порозі. Випадкова вибірка з широкого діапазону може не містити жодного такого значення й не виявити помилку знака порівняння.

У температурних залежностях потрібна перевірка взаємного порядку меж. Критична температура повинна перевищувати безпечну, а допустиме відхилення від комфортної температури має бути додатним. За нульового допустимого відхилення або однакових температурних меж нормована залежність не визначена. Підстановка довільного малого знаменника змінила б модель. Тому коректний профіль має пройти перевірку до початку оцінювання, а некоректність параметризації має бути відокремлена від деградації системи.

Для повторюваного порівняння конфігурацій профіль фіксується на весь інтервал експерименту. До нього належать не тільки пороги таблиці 2.1, а й вагова матриця та правила підготовки телеметрії. Якщо метою є дослідити саме вплив зміни профілю, такий дослід потрібно відокремити від порівняння апаратних або мітигаційних конфігурацій. Інакше джерело зміни інтегрального показника залишатиметься неоднозначним.

Калібрування слід відрізнити від перевірки програмних обчислень. Правильне виконання формули засвідчує, що реалізація відповідає заданій залежності. Воно не доводить, що вибрана залежність або її поріг точно описує конкретний пристрій. Для перевірки цього зв'язку потрібні спостереження відповідних фізичних показників. Контрольні розрахунки додатка В дозволяють відтворити математичну частину; вони не замінюють процедуру узгодження параметрів із даними обладнання.

2.3 Дослідження методу оцінювання стану VR-системи

2.3.1 Побудова симуляційної моделі

З метою перевірки працездатності запропонованого та демонстрації його інтеграції у функціональну модель оцінювання резильєнтності систем віртуальної реальності було реалізовано відповідну симуляційну модель у середовищі MATLAB/Simulink.

Логіка роботи моделі базується на безперервному обчисленні стану системи за дискретними кроками часу. Взаємодія між теоретичним синтезом та верифікацією реалізована через передачу вектора оптимальних параметрів x^* та матриці вагових коефіцієнтів W із робочого простору MATLAB безпосередньо в блоки множення Simulink.

Модель враховує чотири основні компоненти VR-системи: апаратне забезпечення, програмне забезпечення, дані та користувачів, а також п'ять ключових факторів зовнішнього впливу: стан мережі, стан середовища, поведінка користувача, системне навантаження та апаратні обмеження. Відповідно до запропонованого

підходу, кожна пара «фактор–компонент» характеризується окремим ваговим коефіцієнтом, що визначається за методом нечіткого аналізу ієрархій.

Симуляційна модель побудована за модульним принципом, що дозволяє забезпечити гнучкість налаштування параметрів окремих компонентів системи без необхідності перебудови всієї архітектури. Верхній рівень ієрархії моделі представлений трьома основними доменами. Домен генерації вхідних впливів включає блок, що імітує динамічну зміну зовнішніх факторів E . Він включає генератори випадкових процесів, ступінчасті функції для моделювання раптових відмов та гармонійні сигнали для імітації періодичних навантажень. Обчислювальне ядро резильєнтності є центральною частиною моделі, де реалізовано математичний апарат агрегації метрик та застосування вагових коефіцієнтів. Підсистема аналізу та візуалізації включає блоки реєстрації даних, що дозволяють відстежувати часові діаграми інтегрального показника та проводити постаналіз результатів.

Логіка моделі ґрунтується на дискретному обчисленні стану системи. Зв'язок між синтезом і динамічною перевіркою реалізовано через інтеграцію з робочим простором MATLAB. Вектор оптимальних параметрів задає номінальні характеристики компонентів. Матриця вагових коефіцієнтів передається до блока поелементного множення з подальшим підсумовуванням за формулою (2.30). Апаратна, програмна, інформаційна та користувацька підсистеми моделюються окремо.

Для забезпечення високої точності результатів симуляція налаштована на роботу з фіксованим кроком дискретизації, що дозволяє уникнути чисельної нестійкості при моделюванні швидких перехідних процесів. Параметри розв'язувача підібрані таким чином, щоб забезпечити баланс між швидкістю обчислень та деталізацією мікро-відмов, що виникають у VR-середовищі на мілісекундних інтервалах (рис.2.2).

Модель реалізує функціональну залежність (2.30). Вона містить блоки п'яти груп зовнішніх впливів, блок сценаріїв збоїв, двадцять оцінювальних функцій, матрицю ваг і блок агрегації. Матриця оцінок та матриця ваг мають розмірність 5×4 . Їх поелементний добуток підсумовується, утворюючи інтегральний показник $R_{VR}(t)$.

Підсистема обчислення впливів містить демультимплексори вхідних сигналів і двадцять функціональних блоків за формулами (2.9)–(2.28). Їх виходи формують матрицю 5×4 . Блок вагових коефіцієнтів є блоком констант, який ініціалізується матрицею W з робочого простору MATLAB. Позначене на схемі множення є поелементним, після нього виконується підсумовування за формулою (2.30).

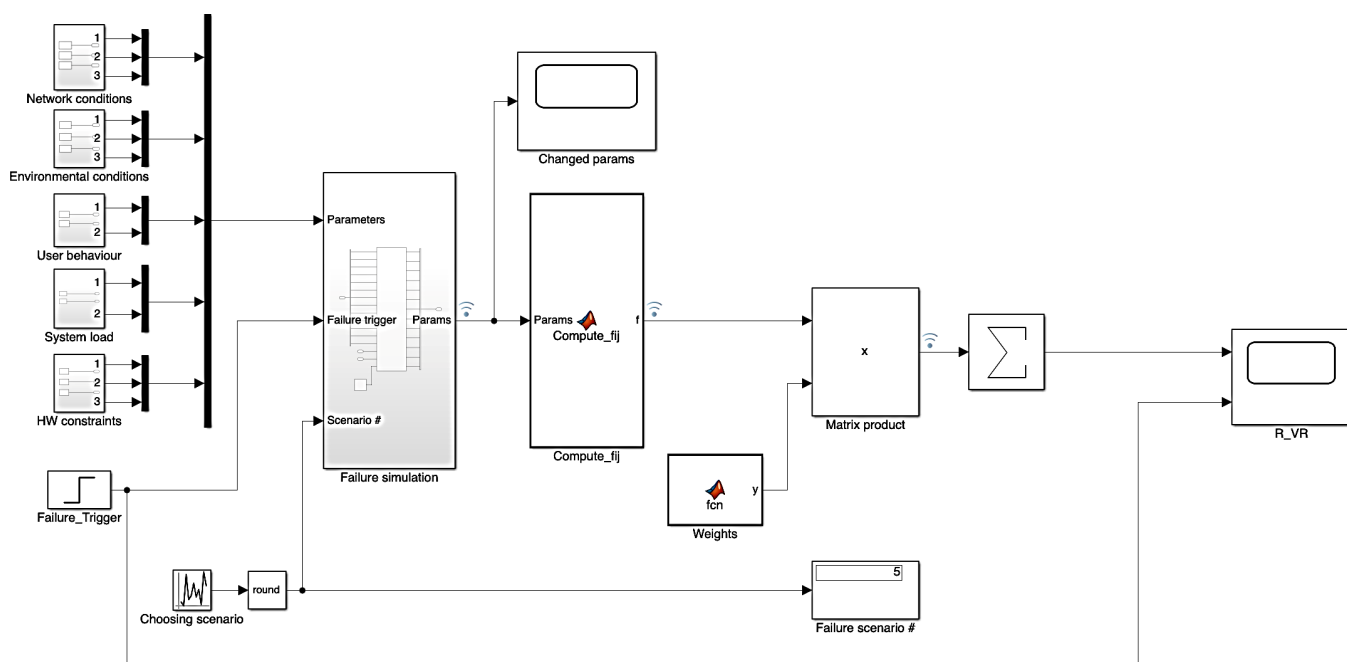


Рисунок 2.2 – Реалізація запропонованої моделі в Simulink

Блок симуляції збоїв за сигналом запуску активує раптову або поступову зміну параметрів вибраного сценарію.

2.3.2 Результати моделювання сценаріїв деградації

У моделі реалізовано шість сценаріїв. Зміни параметрів для кожного сценарію наведено у таблиці 2.3.

До запуску збою параметри утримуються біля номінальних значень із малими випадковими коливаннями. Після запуску застосовується детермінована ступінчаста зміна параметрів за таблицею 2.3. Сценарій аномальної поведінки користувача

поєднує стрибок інтенсивності рухів і частоти подій зі зростанням навантаження. Через дві тисячі часових одиниць після запуску збою застосовуються мітигаційні дії, які компенсують основну частку дефіциту показника, що дає змогу спостерігати відновлення.

Таблиця 2.3 – Опис реалізованих сценаріїв

№	Назва сценарію	Основні зміни параметрів
1	Втрата мережі	Значне зниження пропускнуої здатності B , збільшення втрат пакетів P_{loss} та затримок L
2	Перегрів	Підвищення температури апаратного забезпечення T_{HW} , зростання енергоспоживання P , зменшення заряду батареї B_{life}
3	Перевантаження системи	Зростання навантаження на ЦП C_{load} та пам'ять D_{size} , збільшення затримок L
4	Несприятливі умови середовища	Підвищення температури довкілля T_{ENV} , збільшення шуму N_{noise} , зменшення рівня освітлення L_{light}
5	Аномальна поведінка користувача	Значне зростання обсягу та різкості рухів V , M , збільшення навантаження на систему C_{load}
6	Збій у даних	Збільшення обсягу даних D_{size} , втрат пакетів P_{loss} , падіння пропускнуої здатності B

Ступінь деградації компонентів у температурних сценаріях визначався за наступним принципом. Для кожного компонента задано безпечний діапазон робочих температур, у межах якого функції впливу дорівнюють одиниці; після перевищення порогового значення деградація моделюється лінійним зниженням оцінки пропорційно до величини перевищення, що відповідає типовій поведінці механізмів термічного дроселювання мобільних обчислювальних платформ, а по досягненні критичної температури оцінка обнуляється, що відповідає захисному вимкненню.

Числові значення порогів та швидкості деградації обрано на основі опублікованих досліджень ефектів саморозігріву й деградації напівпровідникових структур [41, 42, 51, 52]. Такий кусково-лінійний закон є свідомим спрощенням нелінійних фізичних процесів, прийнятним для порівняльного аналізу сценаріїв.

Кожний сценарій моделювався протягом 10000 умовних часових одиниць. Сценарій запускався в середині інтервалу, після чого оцінювали просідання та відновлення показника. Підсумкові значення наведено у таблиці 2.4. Часова одиниця є кроком моделювання і не ототожнюється із секундою без окремого калібрування.

Таблиця 2.4 – Значення інтегрального показника R_{VR} за різних сценаріїв

№ сценарію	Назва сценарію	R_{VR}
0	Нормальне функціонування	0,86
1	Втрата мережі	0,51
2	Перегрів	0,81
3	Перевантаження системи	0,60
4	Несприятливі умови середовища	0,67
5	Аномальна поведінка користувача	0,72
6	Збій у даних	0,73

На рисунках наведено результати симуляційних випробувань, які ілюструють поведінку інтегрального показника резильєнтності системи під впливом різних типів збоїв та аномалій. Графіки дозволяють відстежити, як змінюється здатність VR-системи зберігати стабільність після виникнення небажаних умов.

Результати таблиці 2.4 підтверджують обчислювальну працездатність методу оцінювання для заданих модельних сценаріїв. Нормальне функціонування має показник 0,86. Нижчі значення у сценаріях впливів пояснюються зміною окремих безрозмірних функцій та їх ваг. Ці результати не є натурною верифікацією і залежать від заданих порогів, профілів позаштатних ситуацій та вагових коефіцієнтів.

Зниження показника в кожному сценарії визначається зміною локальних функцій та їх ваг. У сценарії перегріву отримано значення 0,81: перевищення температурного порогу обнуляє порогову оцінку апаратної працездатності, проте сукупна вага рядка апаратних обмежень становить 0,049, тому вплив на інтегральний показник залишається обмеженим і проявляється додатково через зростання енергоспоживання та тротлінгове підвищення навантаження. Під час подальшого синтезу ці локальні оцінки входять до компонентних критеріїв за формулою (3.24), а їх системна агрегація виконується за формулами (3.25) і (3.27).

Сценарій втрати мережі має показник 0,51 і дає найглибше просідання серед розглянутих сценаріїв, що узгоджується з найбільшою сукупною вагою мережевого рядка 0,403. Критичною взаємодією стає передавання даних, а розкладання показника за елементами матриці оцінок однозначно вказує на мережеве джерело збою.

Сценарій аномальної поведінки користувача має показник 0,72. Зниження пов'язане зі зростанням інтенсивності рухів, частоти подій і навантаження на обробку сенсорних даних. Безпека користувача залишається обмеженням, тому мітигація не може погіршувати синхронізацію руху або достовірність просторового положення.

Сценарій перевантаження системи має показник 0,60. Модель фіксує одночасне перевищення порогу швидкодії програмного забезпечення, зниження апаратної продуктивності та комфорту користувача. Сценарії несприятливих умов середовища і збою у даних мають значення 0,67 та 0,73. Наведені числа характеризують задану синтетичну параметризацію і не є натурними вимірюваннями.

Таким чином, представлені результати верифікують задачу оптимізації (3.28), доводячи, що синтезована архітектура здатна максимізувати R_{VR}^{final} навіть в умовах агресивного впливу середовища. Аналіз таблиці 2.4 крізь призму рівнянь синтезу підтверджує, що запропонований метод оберненого відображення від цілей резильєнтності до параметрів проектування дозволяє створювати архітектури, які демонструють передбачувану та стабільну поведінку в широкому діапазоні критичних сценаріїв.

На рис. 2.3 показано реакцію системи на аномалії в поведінці користувача. Після запуску збою показник знижується з рівня 0,86 до 0,72 через одночасне

перевищення порогів інтенсивності рухів і частоти подій та зростання навантаження. Після застосування мітигаційних дій система виходить на новий рівень близько 0,82.

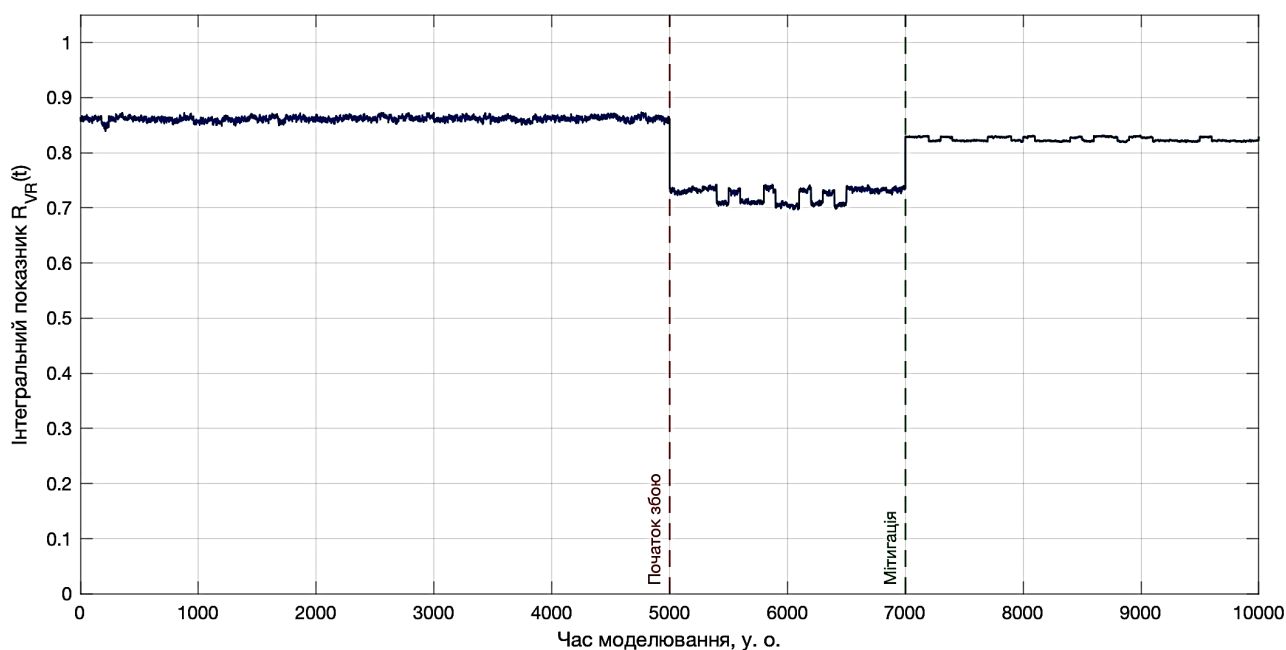


Рисунок 2.3 – Графік симуляції для сценарію 5 – «Аномальна поведінка користувача»

На рис. 2.4 представлено графік для іншого типу навантаження на систему: перегріву апаратного забезпечення. Підвищення температури процесорних модулів, зростання енергоспоживання і зменшення заряду батареї обнуляють порогову оцінку апаратної працездатності, тому показник знижується з рівня 0,86 до 0,81. Після застосування адаптивних заходів охолодження та зменшення навантаження система досягає рівня близько 0,85, дещо нижчого за номінальний.

Розроблена симуляційна модель реалізує описаний у роботі порядок обчислення оцінювальних функцій, визначення вагових коефіцієнтів та формування інтегрального показника резильєнтності XR-системи.

У середовищі MATLAB/Simulink реалізовано замкнений цикл обчислень від формування вхідних параметрів компонентів до визначення інтегрального показника резильєнтності. На кожному кроці симуляції обчислюються індивідуальні функції впливу для пар «фактор–компонент», після чого їх значення агрегуються з

відповідними ваговими коефіцієнтами. Така структура дає змогу простежувати реакцію моделі на температурні, мережеві, обчислювальні та користувацькі фактори в межах заданих сценаріїв.

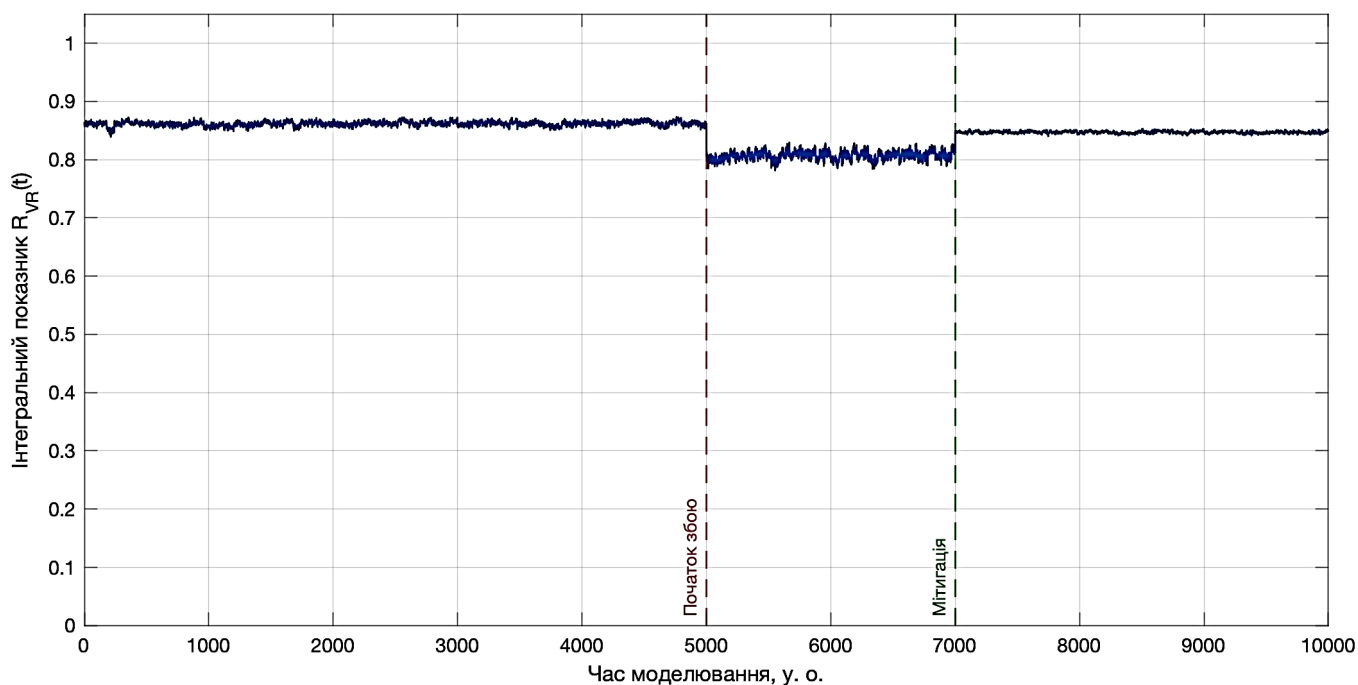


Рисунок 2.4 – Графік симуляції для сценарію 2 – «перегрів»

Нечіткий аналіз ієрархій використано для врахування невизначеності експертних оцінок. Трикутні нечіткі числа формують матрицю вагових коефіцієнтів W для кожної пари «фактор, компонент» з урахуванням поточних умов роботи та довіри до вихідних оцінок. Ваги уточнюються за узгодженими судженнями фахівців. У цій роботі вони не оновлюються за реальною телеметрією під час виконання експерименту, тому адаптивне оновлення ваг розглядається як напрям подальшого розвитку.

Для визначення вагових коефіцієнтів застосовано процедури агрегації та дефазифікації нечітких оцінок. У результаті трикутні нечіткі числа перетворюються на чіткі нормовані значення, які надалі використовуються під час обчислення інтегрального показника.

Моделювання сценаріїв відмов і системних аномалій показало узгоджену реакцію розрахункової схеми на зміну заданих параметрів. У кожному сценарії вхідні

фізичні показники приводяться до параметрів відповідних оцінювальних функцій, після чого застосовуються вагові коефіцієнти та обчислюється інтегральний показник. Отримані результати підтверджують працездатність реалізації для розглянутого набору сценаріїв, але не є доказом коректності моделі для всіх можливих режимів експлуатації.

Після агрегування оцінювальних функцій формується крива інтегрального показника резильєнтності, динаміка якої відображає деградацію системи та її стабілізацію або відновлення після застосування мітигаційних дій. Симуляційна модель дає змогу порівнювати архітектурні конфігурації та реакцію системи на різні профілі впливів у контрольованих умовах до проведення натурних випробувань. Водночас отримані результати обмежуються розглянутими сценаріями та параметризацією й потребують подальшої верифікації на фізичному VR-стенді. Часова траєкторія відновлення формується лише за наявності послідовних станів із заданими часовими мітками та узгодженим законом зміни входів, що охоплюють стан до впливу, деградацію та реакцію після коригувальної дії.

Окреме значення показника також не визначає тривалість перебування у деградованому стані. Дві системи можуть мати однаковий мінімум, але різний час повернення до потрібного рівня; або однаковий середній показник за різних мінімальних значень. Саме тому у циклі розділу 3 оцінка стану поєднується з критеріями виконання функцій і часовим обмеженням. Це дозволяє використати результати цього розділу для локалізації, зберігши окрему перевірку завершеності відновлення.

2.3.3 Аналіз чутливості та стійкості інтегрального показника

Придатність інтегрального показника (2.30) до застосування у замкненому контурі керування визначається його узгодженістю з очікуваною динамікою, підтвердженою симуляційним експериментом, а також аналітичними властивостями, які гарантують передбачувану реакцію на зміну входних даних. Ці властивості

встановлюються безпосередньо з вигляду показника і не залежать від конкретної параметризації.

Оскільки показник є зваженою сумою оцінювальних функцій, його часткова похідна за будь-якою оцінювальною функцією дорівнює відповідному ваговому коефіцієнту:

$$\frac{\partial R_{VR}(t)}{\partial f_{i,j}(t)} = w_{i,j} \quad (2.38)$$

З (2.38) випливає монотонність показника: покращення будь-якої оцінювальної функції не може спричинити зниження інтегрального показника, а ранжування чутливості показника до окремих взаємодій збігається з ранжуванням вагових коефіцієнтів. Це означає, що вагова матриця, отримана нечітким аналізом ієрархій, водночас є картою чутливості системи, і додаткове обчислення чутливостей не потрібне.

Відносна чутливість показника до окремої взаємодії характеризується еластичністю:

$$\varepsilon_{i,j}(t) = \frac{w_{i,j} f_{i,j}(t)}{R_{VR}(t)} \quad (2.39)$$

Еластичність визначає частку внеску пари «фактор – компонент» у поточне значення показника і використовується на етапі локалізації вузьких місць у підрозділі 3.1. Пара, що визначає найбільший недобір показника, знаходиться як

$$(i^*, j^*) = \arg \max_{i,j} w_{i,j} (1 - f_{i,j}(t)) \quad (2.40)$$

тобто критичною визнається не взаємодія з найменшим значенням оцінювальної функції, а та, що дає найбільший втрачений внесок у показник. Таке правило запобігає хибному вибору мітигаційної дії у ситуаціях, коли значно деградована взаємодія має малу вагу і майже не впливає на стан системи.

Стійкість показника до похибок вимірювання встановлюється оцінкою

$$|\Delta R_{VR}| \leq \sum_{i=1}^5 \sum_{j=1}^4 w_{i,j} |\Delta f_{i,j}| \leq \max_{i,j} |\Delta f_{i,j}| \quad (2.41)$$

Отже, показник є ліпшицевим з константою, що дорівнює одиниці: похибка визначення окремої оцінювальної функції не може спричинити більшої за неї зміни інтегрального показника.

Стійкість показника до похибок експертного оцінювання ваг встановлюється так. Нехай кориговані ваги мають вигляд добутку початкових ваг на множник з обмеженим відносним відхиленням із подальшим нормуванням суми до одиниці. Тоді для відносного відхилення, що не перевищує величини δ , справедлива оцінка

$$|R'_{VR} - R_{VR}| \leq \frac{2\delta}{1-\delta} R_{VR} \quad (2.42)$$

Зокрема, за відносної зміни вагових коефіцієнтів у межах десяти відсотків зміна інтегрального показника не перевищує двадцяти двох відсотків його значення, а у межах п'яти відсотків – не перевищує одинадцяти відсотків.

Установлені аналітичні властивості інтегрального показника узагальнено у таблиці 2.5.

Наведені властивості разом із результатами симуляційного експерименту дають підстави вважати інтегральний показник придатним для використання як вхідної величини циклу забезпечення резильєнтності, побудованого у розділі 3: він обмежений, монотонний, піддається явній декомпозиції для локалізації джерела деградації та стійкий до похибок як вимірювання, так і експертного оцінювання.

Оцінка (2.41) характеризує стійкість інтегрального показника до похибок оцінювальних функцій: якщо кожна з двадцяти оцінок має похибку не більш як 0,01, похибка інтегрального показника також не перевищує 0,01. Для окремої оцінки її вплив визначається добутком ваги на величину зміни, тоді як для спільного параметра, що впливає на кілька функцій, необхідно враховувати сумарний внесок усіх залежних елементів. Аналіз чутливості до ваг за (2.42) передбачає застосування відносних змін до початкових ваг із подальшим нормуванням. Отримане обмеження слід трактувати як верхню межу можливої зміни інтегрального показника, а не як прогноз фактичної похибки. Наприклад, для умовних результатів 0,80 і 0,76 та абсолютної межі похибки 0,01 для кожного розрахунку інтервали можливих значень не перекриваються, тому порядок є стійким. Для результатів 0,80 і 0,79 за тих самих

меж інтервали перекриваються. Друге порівняння потребує уточнення даних або обережнішого тлумачення. Приклад ілюструє застосування межі похибки й не є додатковим експериментом над XR-обладнанням.

2.4 Висновки до розділу 2

У розділі обґрунтовано й математично формалізовано нову чотирикомпонентну модель функціонування VR/AR-систем, для якої розроблено 20 нормалізованих оцінювальних функцій та метод визначення інтегрального показника резильєнтності. Метод удосконалено застосуванням нечіткого аналізу ієрархій на основі асиметричних трикутних нечітких чисел. Створена й верифікована імітаційна модель для шести сценаріїв відмов підтвердила дискримінативну здатність показника: за номінального значення 0,86 найбільше зниження до 0,51 спостерігалось при втраті мережі. Також встановлено обмеженість, монотонність, аддитивну декомпозицію та стійкість інтегрального показника, що обґрунтовує його використання у замкненому контурі керування. Одержано оцінку обчислювальної складності методу оцінювання стану. Крок оцінювання має сталу складність за фіксованої структури моделі й не потребує зберігання історії вимірювань, а найтрудомісткіший етап визначення вагових коефіцієнтів виконується лише при зміні профілю місії. Це забезпечує здійсненність кроку оцінювання у вікні часу імерсивної взаємодії, де визначальним обмеженням є періодичність збирання телеметрії, а не обчислення.

Явно сформульовано межі застосовності моделі та три групи джерел похибок. Установлено, що одержані значення інтегрального показника є придатними для порівняння конфігурацій.

Основні результати розділу опубліковані у працях [81, 82, 88].

РОЗДІЛ 3. МЕТОДИ СИНТЕЗУ ТА ОЦІНЮВАННЯ РЕЗИЛЬЄНТНОСТІ СИСТЕМ ВІРТУАЛЬНОЇ ТА ДОПОВНЕНОЇ РЕАЛЬНОСТІ

3.1 Цикл забезпечення резильєнтності VR-систем

Ефективний синтез резильєнтних систем віртуальної та доповненої реальності не може обмежуватися статичним проектуванням або реактивним усуненням помилок. Зважаючи на динамічний характер кіберфізичних параметрів імерсивних середовищ, пропонується побудова архітектури, резильєнтної за структурою. Вона базується на безперервному циклі зворотного зв'язку, який поєднує методи кількісної діагностики деградації, нечіткого зважування та багатокритеріального параметричного синтезу мітигаційних стратегій.

Формалізований ітераційний процес оптимізації архітектури являє собою замкнений контур, що складається з чотирьох послідовних фаз: оцінювання стану, локалізація вузьких місць, синтез мітигаційних стратегій, переоцінка (рис. 3.1).

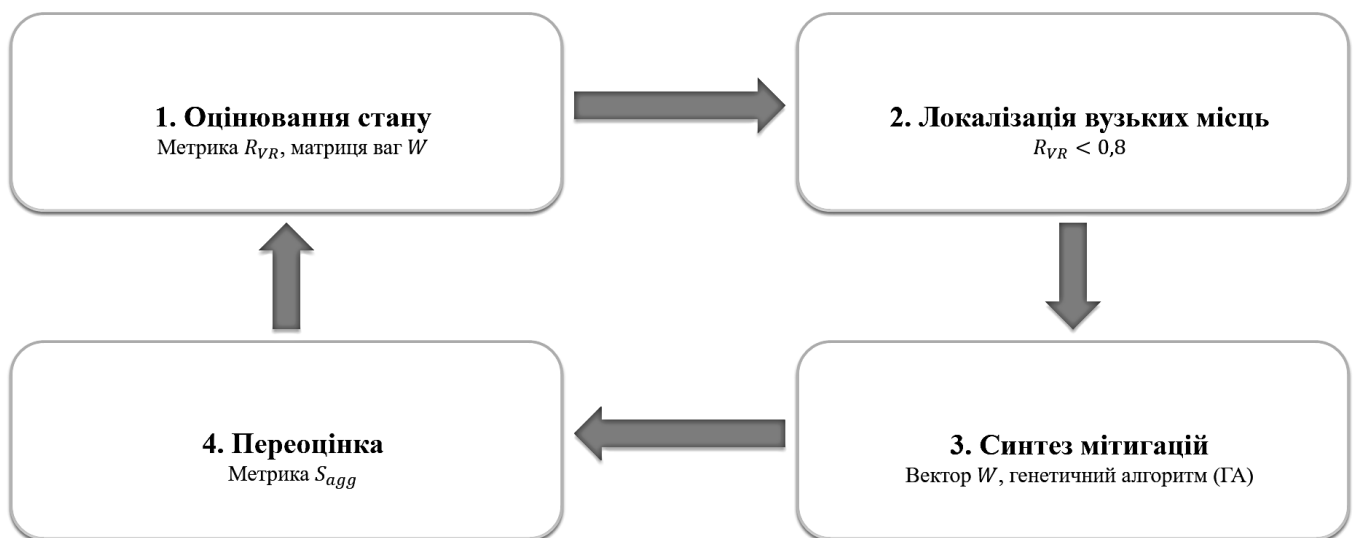


Рисунок 3.1 – Фази ітераційного процесу оптимізації архітектури VR-системи

На першому етапі за моделлю розділу 2 обчислюється матриця двадцяти оцінок взаємодій п'яти груп факторів із чотирма компонентами. Матриця ваг задає відносну важливість взаємодій для профілю місії. Зважене агрегування оцінок утворює інтегральний показник стану, який разом із картою внесків передається на етап локалізації.

На другому етапі цикл переходить до фази діагностики та локалізації вузьких місць. Одержаний на попередньому кроці глобальний показник резильєнтності проходить автоматичну фільтрацію за критерієм безпечного порогу, обґрунтованого у першому розділі. Якщо інтегральне значення резильєнтності стає меншим за встановлену межу, система фіксує виникнення дефіциту резильєнтності архітектури. Після цього алгоритм починає покроковий аналіз теплокарти матриці взаємодій, опускаючись від загального показника до рівня конкретних функцій деградації. Порівнюючи часткові коефіцієнти, система точно локалізує критичний елемент, який зазнав найбільшого руйнівного впливу, і формує цільовий запит до бази знань для підбору відповідних інженерних контрзаходів.

Третій етап присвячено безпосередньому синтезу мітигаційних методів та параметричній оптимізації знайденого слабкого місця. Використовуючи матрицю стратегій підвищення резильєнтності, розроблену в другому розділі, система активує параметричний перетворювач мітигаційного вектора дизайну. Шляхом застосування калібрувальних констант відбувається вибір та налаштування конкретних інструментів захисту.

Для мінімізації наслідків деградації мережі задіюються механізми динамічного масштабування напруги та частоти, налаштування параметрів якості обслуговування або завадостійкого кодування. При критичному навантаженні чи апаратних обмеженнях контур адаптації активує координоване групування завдань, кешування, вивантаження обчислень на віддалені вузли або алгоритм поступової деградації функціоналу системи. Пошук оптимальної комбінації цих параметрів покладається на генетичний алгоритм, який розв'язує задачу Парето-ефективної мультикритеріальної оптимізації, одночасно максимізуючи всі сім цільових функцій резильєнтності архітектури.

На завершальному четвертому етапі виконується переоцінка системи та замикання контуру зворотного зв'язку. Сформований на попередній фазі оптимальний вектор мітигації інтегрується в поточну структуру, трансформуючи систему в оновлений стан. Після модифікації архітектури обчислювальне ядро моделі здійснює повторний розрахунок за формулою нормалізованої зваженої агрегації. Цей розрахунок поєднує всі скориговані метрики надійності, доступності, часу відновлення, відмовостійкості, цілісності, стабільності продуктивності та безпеки. Одержане нове значення порівнюється з вихідними даними стресового стану, що дозволяє кількісно підтвердити кумулятивне відновлення резильєнтності. Після фіксації успішного повернення параметрів до безпечної зони цикл негайно повертається на початкову фазу первинного моніторингу, забезпечуючи безперервну ітераційну адаптацію архітектури до будь-яких нових зовнішніх та внутрішніх завад.

Наведений опис фаз задає послідовність дій, проте не визначає, у яких станах перебувають компоненти системи на кожній фазі та за яких умов виконується перехід між станами.

Для усунення цієї неоднозначності цикл подано формальною моделлю, у якій фази позначено літерами, процеси всередині фаз зображено індексованими малими літерами, а стани компонентів – літерами з штрихами. Такий спосіб опису узгоджується з практикою формалізації замкнених контурів самоадаптації [16, 105, 106], зокрема з формально обґрунтованими модельними підходами до інженерії самоадаптивних систем [133], проте, на відміну від неї, доповнює опис процесів явним описом станів об'єктів керування та умов переходу між ними.

Фази циклу утворюють упорядковану четвірку:

$$\Phi = \langle A, B, C, D \rangle \quad (3.1)$$

де A є фазою оцінювання стану, B є фазою локалізації вузьких місць, C є фазою синтезу мітигаційної стратегії, D є фазою переоцінювання та замикання контуру. Кожна фаза є множиною процесів, які виконуються в межах цієї фази:

$$\begin{aligned}
 A &= \{a_1, a_2, a_3, a_4, a_5\} \\
 B &= \{b_1, b_2, b_3, b_4\} \\
 C &= \{c_1, c_2, c_3, c_4\} \\
 D &= \{d_1, d_2, d_3, d_4\}
 \end{aligned}
 \tag{3.2}$$

Зміст процесів, їх вхідні та вихідні дані наведено у таблиці 3.1. Процеси в межах фази виконуються послідовно, а результат останнього процесу фази є вхідними даними першого процесу наступної фази, чим забезпечується неперервність контуру.

Таблиця 3.1 – Процеси фаз циклу забезпечення резильєнтності XR-системи

Фаза	Процес	Зміст процесу	Результат
A	a ₁	Збирання телеметрії компонентів і параметрів середовища	Вектор вимірюваних параметрів
A	a ₂	Нормування параметрів за нормувальними величинами профілю місії	Безрозмірні параметри
A	a ₃	Обчислення матриці оцінювальних функцій за (2.9)–(2.28)	Матриця F(t) розміру 5×4
A	a ₄	Визначення вагових коефіцієнтів нечітким аналізом ієрархій за (2.31)–(2.37)	Матриця W розміру 5×4
A	a ₅	Агрегація інтегрального показника за (2.30)	Значення R _{VR} (t)
B	b ₁	Порівняння показника з порогом безпечної зони	Ознака дефіциту резильєнтності
B	b ₂	Обчислення втрачених внесків пар «фактор – компонент» за (2.39)	Карта внесків

Фаза	Процес	Зміст процесу	Результат
В	b ₃	Ранжування пар і вибір критичної пари за (2.40)	Критична пара (i*, j*)
В	b ₄	Формування запиту до бази допустимих контрзаходів	Множина допустимих дій
С	c ₁	Відбір контрзаходів, сумісних із профілем місії	Скорочена множина дій
С	c ₂	Параметричне перетворення мітигаційного вектора	Вектор проєктних параметрів
С	c ₃	Мультикритеріальна Парето-оптимізація цільових функцій	Наближення Парето-фронту
С	c ₄	Перевірка обов'язкових обмежень збереження критичних функцій	Допустимий вектор m*
Д	d ₁	Застосування мітигаційного вектора до конфігурації системи	Оновлена конфігурація
Д	d ₂	Повторне оцінювання стану за фазою А	Значення R _{VR} (t _r)
Д	d ₃	Перевірка умов завершеного відновлення за q _{req} та T _{rec}	Ознака завершеності відновлення
Д	d ₄	Фіксація конфігурації або відкат до попередньої	Рішення про продовження циклу

Стан окремого компонента системи описується скінченною множиною станів.

Для компонента з індексом i множина станів має вигляд:

$$S_i = \{A_i, A'_i, A''_i, A^*_i\} \quad (3.3)$$

де A_i є номінальним станом компонента, у якому всі закріплені за ним критичні функції виконуються на рівні, не нижчому за необхідний; A'_i є деградованим станом,

у якому щонайменше одна оцінювальна функція компонента знизилася нижче власного допуску, проте критичні функції залишаються виконуваними; A''_i є мітигованим перехідним станом, у якому мітигаційний вектор уже застосовано, але умови завершеного відновлення ще не підтверджено; A^*_i є новим усталеним станом зі зниженим, але припустимим рівнем виконання функцій.

Перехід із номінального стану в деградований виконується під дією вектора зовнішніх впливів і описується оператором деградації:

$$A'_i = \delta(A_i, E(t)) \quad (3.4)$$

Оператор δ не є керованим: він відображає дію середовища і фіксується системою на фазі А за результатами процесів a_1 – a_s . Фаза В не змінює стану компонента, а лише формує інформаційний образ цього стану, тобто визначає, який саме компонент і під дією якого фактора перейшов у деградований стан.

Перехід із деградованого стану в мітигований виконується застосуванням оптимального мітигаційного вектора, синтезованого на фазі С:

$$A''_i = \mu(A'_i, m^*) \quad (3.5)$$

Подальша доля мітигованого стану визначається на фазі D за результатами процесів d_2 і d_3 . Якщо повторне оцінювання підтверджує досягнення необхідного рівня виконання критичних функцій у межах граничного часу відновлення, виконується оператор повного відновлення, і компонент повертається до початкового номінального стану:

$$A_i = \rho(A''_i), \text{ якщо } q(t_r) \geq q_{\text{req}} \text{ та } t_r \leq t_1 + T_{\text{rec}} \quad (3.6)$$

Якщо досягнутий рівень перевищує мінімально допустимий, але не досягає необхідного, виконується оператор часткового відновлення, і компонент переходить у новий усталений стан:

$$A^*_i = \rho^*(A''_i), \text{ якщо } q_{\min} \leq q(t_r) < q_{\text{req}} \quad (3.7)$$

Якщо ж повторне оцінювання показує, що рівень виконання критичних функцій залишився нижчим за мінімально допустимий, мітигація визнається невдалою, виконується оператор відкату, і компонент повертається у деградований стан, після чого цикл виконується повторно з виключенням застосованого вектора з множини допустимих дій:

$$A'_i = \omega(A''_i), \text{ якщо } q(t_r) < q_{\min} \quad (3.8)$$

Успішна ітерація циклу утворює замкнену композицію операторів, у якій компонент повертається до початкового стану:

$$\rho\left(\mu\left(\delta(A_i, E(t)), m^*\right)\right) = A_i \quad (3.9)$$

Саме рівність (3.9) є формальним виразом резильєнтності компонента у розумінні означення, наведеного у підрозділі 1.1: система витримує вплив, а також повертається до початкового стану. Якщо замикання досягається оператором ρ^* , резильєнтність вважається забезпеченою частково, а недобір рівня виконання критичних функцій є кількісною мірою цієї неповноти.

Стан системи в цілому подається вектором станів її компонентів:

$$V(t) = (A_H(t), A_S(t), A_D(t), A_U(t)) \quad (3.10)$$

де індекси відповідають апаратному забезпеченню, програмному забезпеченню, даним і користувацькому компоненту, введеним у підрозділі 2.1. Кожна складова вектора набуває значень із власної множини станів згідно з (3.3). Система перебуває у номінальному стані лише тоді, коли всі чотири складові перебувають у номінальних станах; наявність хоча б однієї деградованої складової є достатньою умовою запуску наступної ітерації циклу.

Така побудова дає змогу описувати часткову деградацію, за якої, наприклад, дані та користувацький компонент залишаються у номінальних станах, тоді як апаратне та програмне забезпечення перебувають у деградованих станах, і мітигація застосовується вибірково.

Оператори переходу між станами, умови їх застосування та фази, на яких вони виконуються, узагальнено у таблиці 3.2.

Таблиця 3.2 – Оператори переходу станів компонентів XR-системи

Оператор	Перехід	Фаза	Умова застосування	Зміст
δ	$A_i \rightarrow A'_i$	A	Дія вектора зовнішніх впливів $E(t)$	Деградація під дією впливу
–	$A_i \rightarrow A'_i$	B	Значення показника нижче порогу безпечної зони	Локалізація критичної пари без зміни стану
μ	$A'_i \rightarrow A''_i$	C, D	Синтезовано допустимий вектор m^*	Застосування мітигаційної стратегії
ρ	$A''_i \rightarrow A_i$	D	$q(t_r) \geq q_{req}$ і $t_r \leq t_1 + T_{rec}$	Повне відновлення до початкового стану
ρ^*	$A''_i \rightarrow A^*_i$	D	$q_{min} \leq q(t_r) < q_{req}$	Часткове відновлення до нового усталеного рівня
ω	$A''_i \rightarrow A'_i$	D	$q(t_r) < q_{min}$	Відкат мітигації та повторна ітерація

Граф переходів між станами компонента, побудований за (3.4)–(3.8), подано на рисунку 3.2. Наявність оператора відкату ω робить граф циклічним на рівні повного циклу, а також у кожній з фаз переоцінювання, що відповідає багатократному застосуванню циклу до одного й того самого впливу з різними мітигаційними

векторами. Кількість повторів обмежується або вичерпанням множини допустимих дій, або граничним часом відновлення профілю місії; у другому випадку подія фіксується як відмова відновлення і потребує втручання оператора.

Уведена модель дає три практичні наслідки. По-перше, вона розмежовує стани, які раніше описувалися одним поняттям деградації: стан A'_i вимагає синтезу мітигаційної стратегії, тоді як стан A''_i вимагає лише перевірки, і плутання цих станів призводить до повторного застосування мітигації до вже мітигрованої конфігурації.

По-друге, вона задає однозначний критерій завершення ітерації, оскільки умови (3.6)–(3.8) є взаємно виключними і охоплюють усі можливі результати переоцінювання.

По-третє, вона визначає місце кожного розробленого у роботі методу в загальному процесі: метод оцінювання стану реалізує процеси a_3 – a_5 та d_2 , метод синтезу мітигаційних стратегій реалізує процеси c_1 – c_4 , а програмний засіб, описаний у розділі 4, реалізує повний цикл разом із процесами збирання телеметрії та фіксації конфігурації.

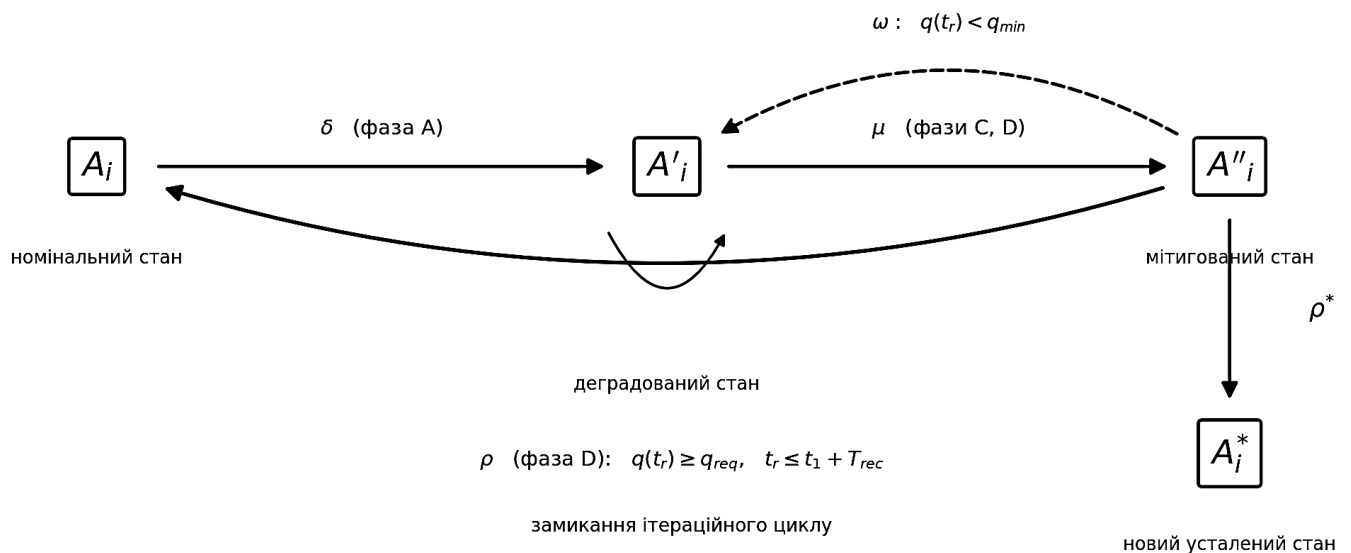


Рисунок 3.2 – Граф переходів станів компонента XR-системи в ітераційному процесі

3.2 Метод синтезу мітигаційних стратегій для VR-систем

Інтегральний показник резильєнтності використовується для оцінювання стану VR-системи. Він виявляє слабкі місця і формує основу для вибору заходів усунення деградації.

Для систематизації допустимих мітигаційних дій використовується матриця стратегій, у якій рядки відповідають групам впливів, а стовпці – компонентам системи: апаратному забезпеченню, програмному забезпеченню, даним і користувачам. Матриця пов'язує виявлену критичну пару «фактор–компонент» із набором дій, параметри яких надалі оптимізуються методом синтезу.

Інтегральний показник використовується для фіксації загального стану, а також для локалізації внеску окремих взаємодій «фактор-компонент». Це дає змогу визначити, які саме взаємодії формують найбільший дефіцит резильєнтності, та обмежити простір пошуку мітигацій відповідними діями. Після застосування вибраного вектора мітигації стан системи оцінюється повторно за тим самим показником.

Цінність інтегрального показника полягає у прозорій декомпозиції. Розробник може виявити зниження загальної резильєнтності та ідентифікувати конкретні критичні взаємодії. Для локалізації аналізуються часткові значення функцій деградації. Після цього система формує перелік заходів для компонентів, внесок яких у зниження показника є найбільшим.

Матриця стратегій мітигації формалізує відповідність між групами впливів, компонентами системи та допустимими діями. Для кожної пари «фактор–компонент» задається набір параметризованих засобів, наприклад керування якістю обслуговування, корекція помилок, контроль теплового режиму, перерозподіл обчислювальних ресурсів або контрольоване зниження деталізації. Конкретний склад дій визначається обмеженнями моделі та профілем місії.

Метою матриці є формалізація простору допустимих мітигаційних дій для подальшої оптимізації. Ефективність окремої дії оцінюється за зміною цільових критеріїв та інтегрального показника після повторного розрахунку стану системи.

Такий підхід дає змогу порівнювати альтернативні комбінації дій за однакових вхідних впливів і відкидати конфігурації, що порушують обмеження критичних функцій.

Послідовне застосування мітигаційних дій змінює параметри системи та, відповідно, значення оцінювальних функцій і критеріїв резильєнтності. У задачі синтезу ці зміни враховуються через параметричний вектор мітигації та цільову функцію. Отриманий результат характеризує лише задану конфігурацію, профіль місії та сценарій впливів і потребує повторного обчислення за зміни цих умов.

Розроблений метод утворює цикл удосконалення VR-системи: виявлення структурних слабкостей, вибір адаптивних механізмів, застосування мітигації та повторне оцінювання. Матричний підхід дає змогу документувати ефективність кожного заходу. Це створює основу для самовідновлюваних систем, які виявляють дефекти в реальному часі та вибирають стратегію нейтралізації загрози.

Матриця стратегій може розширюватися новими мітигаційними діями за умови визначення їх параметрів, області допустимих значень і впливу на критерії резильєнтності. Розширення матриці не змінює загальної структури процедури синтезу.

Математична формалізація вибору дій із матриці переводить задачу вибору мітигації у багатокритеріальну задачу оптимізації з визначеними змінними, обмеженнями та цільовими критеріями. Далі цей простір використовується для пошуку вектора мітигації, який задовольняє умови відновлення та обмеження профілю місії.

Таким чином, отримаємо:

$$M = [m_{i,j}]_{5 \times 4}, \quad 0 \leq m_{i,j} \leq 1 \quad (3.11)$$

де $\{M_{i,j} \in M, i \in H, j \in VR\}$.

Допустимі мітигаційні дії групуються відповідно до п'яти типів впливів:

1. Для мережевих впливів використовуються керування якістю обслуговування, корекція помилок, резервування каналів і адаптація потоку даних. Ці

дії спрямовані на зменшення втрат пакетів і затримки за дотримання вимог до цілісності даних.

2. Для температурних і шумових впливів застосовуються керування охолодженням, обмеження навантаження, резервування даних та адаптація інтерфейсу. Безпечні температурні межі задаються як обов'язкові обмеження.

3. Для аномальної поведінки користувача застосовуються калібрування сенсорів, фільтрація руху, обмеження темпу подій та контроль переповнення черг.

4. Для системного перевантаження застосовуються групування завдань, кешування, перерозподіл ресурсів і контрольоване зниження деталізації зі збереженням мінімальної частоти кадрів.

5. Для апаратних обмежень використовуються резервні ресурси, паралельне виконання, контроль енергоспоживання та перехід у контрольований деградований режим із подальшим відновленням.

Наведені групи дій утворюють простір параметрів для синтезу мітигаційної стратегії. Їх застосування обмежується умовами профілю місії та вимогами до збереження критичних функцій.

Запропонована модель поєднує оцінювання поточного стану з вибором і повторною перевіркою мітигаційних дій. Моніторингові параметри використовуються для обчислення оцінювальних функцій, після чого метод визначає критичні взаємодії та формує вектор допустимих дій для оптимізації.

Результатом застосування моделі є оцінка стану до та після мітигації, що дає змогу перевірити виконання заданих умов відновлення для конкретного сценарію.

Визначимо функцію, яка забезпечує оптимізацію резильєнтності за допомогою підходів до мінімізації впливу негативних факторів. Ця функція враховує всі ідентифіковані негативні впливи на компоненти системи та їхній взаємозв'язок з відповідними стратегіями мінімізації. Вона математично формалізує процес адаптації VR-системи до змінних умов, забезпечуючи підтримання стабільності, продуктивності та комфортного користувацького досвіду.

Функція входить до інтегральної моделі резильєнтності. Вона оцінює поточний стан системи та використовується під час оптимізації мітигаційної стратегії. Вагові коефіцієнти визначають внесок компонентів у загальний показник.

Ця функція використовується для прогнозування наслідків впливів, мінімізації ризиків і підвищення ефективності системи за зовнішніх впливів та ресурсних обмежень.

$$f_{\text{opt}}: V \times E \times M \rightarrow V_{\text{res}} \quad (3.12)$$

У формулі (3.12) V позначає початкову архітектуру, E є вектором впливів, M є матрицею мітигаційних засобів, V_{res} є архітектурою після синтезу.

Інтегральний показник резильєнтності характеризує поточний стан системи та використовується для перевірки її здатності до адаптації. Метод визначає дефіцит резильєнтності як різницю між номінальним станом і станом під дією зовнішніх факторів, після чого вибирає вектор мітигаційних засобів.

Ітераційний алгоритм пов'язує модель оцінювання з процедурою синтезу. Він виконує такі кроки:

1. Формування базової архітектури та її початкових параметрів.
2. Формування профілю зовнішніх впливів і системного навантаження.
3. Вибір допустимих мітигаційних дій для виявлених критичних взаємодій.
4. Повторне обчислення інтегрального показника та перевірка умов відновлення.

Ефективність методу підвищення резильєнтності безпосередньо залежить від коректності побудови матриці Mat . Кожен елемент матриці $Mat_{i,j}$ представляє собою оператор впливу j -го засобу мінімізації на j -й компонент системи.

3.3 Синтез резильєнтних XR-архітектур

Важливим є складне суб'єктивне завдання якісного проектування на чітко структуровану задачу чисельної оптимізації, де кожен елемент системи має своє

математичне вираження. Для ефективного дослідження цього багатовимірного та гетерогенного простору проектування застосовано генетичний алгоритм, здатний оперувати великою кількістю змінних та знаходити оптимальні комбінації стратегій захисту. Це дає змогу ідентифікувати такі конфігурації пом'якшення негативних впливів, які забезпечують максимальний рівень загальної резильєнтності системи при збереженні високої якості користувацького досвіду та мінімізації обчислювальних витрат.

Попри те, що існуючі методи демонструють значний прогрес у вирішенні окремих аспектів резильєнтності XR-середовищ, вони залишаються значною мірою фрагментарними та вузькоспеціалізованими в межах своїх доменів. Роботи, зосереджені на апаратному забезпеченні, забезпечують ефективну локальну відмовостійкість компонентів через термічне управління та надлишковість, проте вони практично не враховують поведінкові аспекти користувача або складні динамічні зміни навколишнього середовища. Програмні механізми відновлення пропонують інструментарій для реагування на збої, але вони функціонують без інтеграції з моделями варіативності мережі або вимогами до фізіологічного комфорту людини. Рішення, орієнтовані на дані, спрямовані на максимізацію надійності передачі імерсивного контенту, проте вони зазвичай діють незалежно від жорстких апаратних обмежень пристроїв та динамічної зміни загального системного навантаження. Навіть дослідження на рівні користувача, які пріоритезують ергономіку та комфорт, існують у відриві від системних метрик продуктивності, що не дозволяє оцінити їхній вплив на загальну цілісність архітектури. навіть найбільш холістичні методи до оптимізації часто трактують резильєнтність як односпрямовану ціль у межах одного домену, а не як мультикритеріальну властивість, що виникає внаслідок нелінійної взаємодії між апаратним забезпеченням, програмним кодом, потоками даних, людьми та випадковими зовнішніми чинниками.

Ця фрагментація дозволяє зробити висновок про те, що на сьогодні не існує уніфікованого методу синтезу, здатного інтегрувати техніки зміцнення резильєнтності з усіх архітектурних рівнів та максимізувати їх спільно в умовах складного поєднання динамічних середовищних, поведінкових та навантажувальних

впливів. Відсутність такого методу призводить до того, що оптимізація окремого рівня може спричинити деградацію резильєнтності всієї системи через непередбачувані конфлікти параметрів. Запропонований метод безпосередньо усуває цей розрив шляхом впровадження комплексної багатофакторної структури синтезу. Вона базується на формуванні єдиного багатовимірного вектора дизайну $\vec{X}$, що охоплює параметри всіх рівнів, і використанні математичного апарату кількісної оптимізації для пошуку глобального максимуму резильєнтності, забезпечуючи створення систем, що є «резильєнтними за конструкцією».

Резильєнтна архітектура характеризується здатністю підтримувати свій функціональний вихід на рівні, що є вищим за критично встановлений мінімальний поріг працездатності, протягом усього визначеного часу виконання, незалежно від динаміки виникнення внутрішніх чи зовнішніх впливів. У системах доповненої та віртуальної реальності цей вихід безпосередньо корелює з метриками реального часу, такими як латентність від руху до фотона та стабільність частоти оновлення кадрів, де будь-яке падіння продуктивності нижче межі прийнятності миттєво призводить до деградації користувацького досвіду та втрати системної цілісності. Таким чином, математичне визначення резильєнтності системи в межах заданого часового інтервалу може бути формалізоване наступним чином:

Збереження критичної функції $F(t)$ протягом місії задається умовою:

$$F(t) \geq F_{\min}, \quad t \in [0, T] \quad (3.13)$$

$F_{\min}$ є безрозмірним мінімально допустимим рівнем виконання критичної функції.

Наведені далі співвідношення (3.13)–(3.21) для надійності, доступності, відмовостійкості, цілісності, часу відновлення, стабільності продуктивності та безпеки користувача є класичними результатами теорії надійності й резильєнтності комп'ютерних систем і подаються за усталеними джерелами [75, 76] для повноти критеріальної бази. Внесок даної роботи полягає не у виведенні цих формул, а в їх

адаптації до специфіки XR-середовищ, узгодженні з двадцятьма оцінювальними функціями розділу 2 та інтеграції у єдину задачу синтезу (3.27), (3.28).

Надійність визначається ймовірністю безвідмовної роботи протягом часу t :

$$R(t) = \Pr\{T_f > t\} = e^{-\lambda t} \quad (3.14)$$

λ є інтенсивністю відмов з одиницею, оберненою до одиниці часу, тому добуток λt безрозмірний.

Доступність визначається через середній час безвідмовної роботи та середній час відновлення:

$$A = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \quad (3.15)$$

Підвищення доступності потребує збільшення середнього часу безвідмовної роботи і скорочення середнього часу відновлення. Обидва часові показники задаються в однакових одиницях.

Відмовостійкість системи з n однотипних каналів, де для виконання функції потрібно щонайменше k працездатних каналів, визначається за біноміальною моделлю:

$$FT = \sum_{i=k}^n \binom{n}{i} R_c^i (1 - R_c)^{n-i} \quad (3.16)$$

R_c є безрозмірною надійністю одного каналу.

Цілісність даних визначається часткою правильно відновлених або збережених елементів:

$$I = 1 - \frac{E_d - E_c}{E_d} = \frac{E_c}{E_d} \quad (3.17)$$

Порушення синхронізації та цілісності руйнують ефект присутності та можуть створювати ризик для користувача.

Час відновлення після окремого збою визначається різницею моментів відновлення і виникнення відмови:

$$RT = t_{\text{restore}} - t_{\text{fail}} \quad (3.18)$$

Середній час відновлення за N подіями визначається як середнє арифметичне тривалостей відновлення:

$$MTTR = \frac{1}{N} \sum_{i=1}^N (t_{\text{restore},i} - t_{\text{fail},i}) \quad (3.19)$$

RT і $MTTR$ мають одиницю часу. Перед включенням до агрегованої цільової функції вони перетворюються на безрозмірні оцінки v_{RT} і v_{MTTR} , які зменшуються зі зростанням часу відновлення.

З точки зору синтезу резильєнтної архітектури, показник стабільності продуктивності дозволяє оцінити здатність системи до «граційної деградації» під впливом зовнішніх або внутрішніх факторів. Замість раптового припинення роботи, резильєнтна архітектура повинна демонструвати передбачувану та контрольовану поведінку, мінімізуючи відхилення від цільових показників продуктивності за допомогою методів адаптивного управління, таких як динамічне масштабування деталізації або прогностичне префетчування даних. Математично стабільність продуктивності формалізується через оцінку розбіжності між реальним значенням параметра продуктивності $P(t)$ у часі та його ідеальним номінальним значенням P_0 :

$$PS(t) = \left[1 - \frac{|P(t) - P_0|}{P_0} \right]_0^1 \quad (3.20)$$

$P(t)$ і P_0 мають однакові одиниці, тому $PS(t)$ безрозмірний.

Безпека користувача визначається через імовірність небезпечного стану p_{hazard} . Коефіцієнт ρ задає частку системних відмов, що можуть створити небезпеку:

$$US = 1 - p_{\text{hazard}} = 1 - \rho(1 - R_{\text{sys}}) \quad (3.21)$$

p_{hazard} , ρ , R_{sys} та US є безрозмірними величинами на інтервалі $[0; 1]$.

Отримана структура встановлює уніфікований фундамент для чисельної оптимізації резильєнтних архітектур XR, забезпечуючи можливість проектування систем, які володіють внутрішньою здатністю протистояти деструктивним впливам та адаптуватися до мінливих умов навколишнього середовища і поведінки користувача. Це закладає основу для переходу від емпіричного дизайну до науково обґрунтованого синтезу резильєнтних імерсивних середовищ.

Загальна база розглядає ці критерії як взаємозалежні цілі оптимізації, де покращення в одному вимірі часто синергетично сприяє прогресу в інших; радикальне скорочення часу відновлення безпосередньо підвищує коефіцієнт готовності системи, тоді як зниження інтенсивності помилок у мережевих пакетах одночасно покращує і цілісність інформації, і стабільність частоти кадрів. Однак ці взаємозв'язки часто є нелінійними: надмірне використання прямої корекції помилок може підвищити цілісність, але одночасно створити обчислювальне перевантаження, що негативно вплине на латентність. Відповідно, процедура синтезу спрямована на пошук багатовимірного оптимуму, де комбінація проектних рішень для апаратних, програмних, мережевих та користувацьких підсистем забезпечує найвищий рівень загальної робастності системи за заданих середовищних та операційних обмежень.

Методика передбачає незалежну максимізацію кожного окремого критерію резильєнтності при жорсткій умові мінімізації часових витрат на відновлення після збоїв. Усі ключові проектні рішення, такі як рівні апаратної надлишковості, стратегії динамічного регулювання напруги та частоти, періодичність створення контрольних точок, рівень прямої корекції помилок та параметри захисного режиму (guardian mode), проходять кількісну оцінку на відповідність встановленим вимірюваним критеріям.

Зокрема, впровадження політик DVFS дозволяє балансувати між продуктивністю та термічною надійністю, тоді як налаштування частоти контрольних точок безпосередньо визначає компроміс між накладними витратами під час нормальної роботи та швидкістю регенерації системи після відмови. Весь цей процес включає глибокий аналіз надійності, доступності, відмовостійкості, цілісності,

стабільності продуктивності, безпеки користувача, а також середнього та поточного часу відновлення. Кінцевий результат синтезу формулюється як багатокритеріальна постановка задачі математичної оптимізації, що дозволяє знайти такий вектор параметрів архітектури, який відповідає Парето-ефективному стану системи:

Задача синтезу полягає у максимізації вектора нормованих критеріїв:

$$\max_{\mathbf{x}} \left(R_{\text{sys}}, A_{\text{sys}}, FT, I, PS, US, v_{RT}(RT), v_{\text{MTTR}}(\text{MTTR}) \right)^T \quad (3.22)$$

Часові критерії входять до вектора після нормування. Це усуває змішування секунд із безрозмірними ймовірнісними показниками.

Профіль місії задає тривалість роботи, типові режими та пікові навантаження. Структурна модель надійності визначає логічні залежності між відмовами підсистем. Для відновлюваних компонентів використовуються моделі станів із безперервним часом, що дають змогу оцінити доступність і середній час відновлення.

Під час підготовки перевіряються засоби вимірювання затримки, журнали спрацювання захисних бар'єрів, виявлення колізій та екстрене зупинення. Конфігурації, що не відповідають мінімальним порогам надійності, безпеки або часу відновлення, не допускаються до синтезу.

Мітигаційні дії призначаються окремо для апаратної, програмної, інформаційної та користувацької підсистем. Їх параметри входять до вектора проектування і впливають на відповідні критерії резильєнтності.

Синтез перетворює перелік допустимих дій на числові параметри оптимізації. Пошук виконується за ресурсних, температурних і безпекових обмежень. Результатом є конфігурація з найбільшим агрегованим показником за заданого профілю місії.

Кожний рівень архітектури подано окремою функціональною підсистемою з власними параметрами. Локальні оцінки поєднуються на системному рівні, тому оптимізація окремого компонента не може ігнорувати його зв'язки з іншими компонентами.

Зовнішні змінні охоплюють стан мережі, температурний режим, поведінку користувача, навантаження та апаратні обмеження. Їх вплив обчислюється функціями розділу 2 і використовується для вибору мітигаційного вектора.

$$E = (E_N, E_{ENV}, E_B, E_S, E_{HW})^T \quad (3.23)$$

Вектор E містить стан мережі, параметри середовища, поведінку користувача, системне навантаження та апаратні обмеження.

Кожна підсистема має власний вектор проєктних параметрів. Ці параметри визначають можливості адаптації та впливають на підсумковий рівень резильєнтності.

Синтез визначає параметри проєктування, за яких агрегований показник за інтервал місії набуває найбільшого значення:

$$\mathbf{x}^* = \operatorname{argmax}_{\mathbf{x} \in \Omega} R_{\text{mission}}(\mathbf{x}, E) \quad (3.24)$$

Для критерію k і компонента c номінальне значення $q_{c,k}^0$ коригується добутком п'яти функцій впливу:

$$q_{c,k} = q_{c,k}^0 \prod_{r=1}^5 f_{r,c}(\mathbf{x}, E) \quad (3.25)$$

Кожний множник $f_{r,c}$ є безрозмірною часткою збереженої працездатності. Значення одиниця відповідає відсутності деградації, менші значення показують втрату функціональності.

Системне значення критерію k утворюється оператором агрегації Φ_k над чотирма компонентами:

$$Q_k = \Phi_k(q_{1,k}, q_{2,k}, q_{3,k}, q_{4,k}) \quad (3.26)$$

Оператор Φ_k вибирається відповідно до структури критерію. Для послідовної структури застосовується добуток, для резервованої структури використовується ймовірнісна схема, для зваженого критерію використовується нормована сума.

Цільова функція є зваженою сумою нормованих критеріїв:

$$R_{\text{mission}} = \sum_{k=1}^K \beta_k v_k(Q_k), \quad \beta_k \geq 0, \quad \sum_{k=1}^K \beta_k = 1 \quad (3.27)$$

Перетворення v_k приводить кожний критерій до інтервалу $[0; 1]$. Для часових показників використовується спадне перетворення, для ймовірностей і часток використовується тотожне або порогове перетворення.

Підсумкова задача оптимізації має вигляд:

$$x^* = \operatorname{argmax}_{x \in \Omega} R_{\text{mission}}(x) \quad \text{за умов} \quad g_s(x) \leq 0, s = 1, \dots, S \quad (3.28)$$

Для підтвердження ефективності та практичної користі запропонованого методу аналітичного синтезу було проведено чисельне оцінювання. Метою експерименту є демонстрація здатності моделі здійснювати навігацію у багатовимірному просторі проектування стратегій мінімізації впливів та ідентифікувати оптимальну архітектуру.

На додаток до аналітичного формулювання, процедура синтезу реалізована безпосередньо в середовищі Simulink, що дозволяє проводити високоточне оцінювання резильєнтності в умовах динамічної зміни робочих параметрів. Simulink забезпечує структуроване представлення всіх компонентів на рівні блоків, функцій впливу та механізмів агрегації.

Завдяки вбудовуванню конвеєра обчислення резильєнтності всередину Simulink та представленню ключових архітектурних параметрів як конфігурованих змінних, генетичний алгоритм може оцінювати кожен кандидатний вектор мінімізації на реалістичній моделі поведінки системи. Це гарантує, що синтезована архітектура залишається узгодженою як з математичною моделлю, так і з базовими фізичними припущеннями середовища VR.

3.4 Дослідження методів забезпечення резильєнтності XR-систем

Розроблені у цьому та попередньому розділах методи досліджено у два етапи. Спочатку кожен метод досліджено окремо, за власними показниками результативності та за умов, що виявляють саме ту властивість, яку цей метод має забезпечувати. Після цього досліджено узагальнений метод, тобто повний цикл, у якому методи застосовуються послідовно і результативність визначається замиканням циклу за (3.9). Такий порядок дає змогу відрізнити внесок окремого методу від внеску їхнього поєднання і встановити, чи виникає у поєднанні властивість, відсутня в окремих методах.

Дослідження виконано на обчислювальному ядрі програмного засобу, реалізація якого описана у розділі 4. Використання єдиного ядра для всіх дослідів забезпечує відтворюваність результатів і виключає розбіжності, зумовлені відмінностями реалізацій; збіжність ядра із Simulink-моделлю за уніфікованих вхідних даних підтверджено у наступному розділі.

3.4.1 Методика та умови дослідження

Для верифікації реалізованого програмного засобу визначено чотири операційних сценарії, що охоплюють повний діапазон умов експлуатації від ідеальних до катастрофічних. Сценарії спроектовані для тестування різних аспектів матриці оцінювальних функцій та демонстрації дискримінативної здатності інтегрального показника резильєнтності.

Визначення експериментальних сценаріїв подано в таблиці 3.3.

Додатково порівнюються дві конфігурації мітигації: базова та оптимізована. Базова конфігурація використовує значення за замовчуванням мітигаційного вектора: $M1 = 0.3$ (поріг вивантаження), $M4 = 4$ (поток), $M5 = 1$ (OLED-дисплей), $M9 = 3$ (частота контрольних точок), $M11 = 2$ (мережеві канали), $M14 = 20$ (розмір кешу), $M15 = 2$ (рівень кодування), $M18 = 1$ (телепортація).

Таблиця 3.3 – Визначення експериментальних сценаріїв

Сценарій	Опис	Ключові зміни параметрів
Default	Типове використання VR у приміщенні	$B=100$, $L=50$, $T_{env}=25$, $C_{load}=0.5$, $T_{hw}=40$
Ideal	Найкращі лабораторні умови	$B=500$, $L=10$, $T_{env}=25$, $C_{load}=0.2$, $T_{hw}=30$
Stress	Мультифакторна деградація	$B=20$, $L=180$, $T_{env}=42$, $C_{load}=0.9$, $T_{hw}=65$
Network failure	Ізольований збій мережі	$B=5$, $L=350$, $J=80$, $P_{loss}=0.4$

Оптимізована конфігурація застосовує агресивні контрзаходи: $M1 = 0.7$, $M3 = 6$ (чутливість до руху), $M4 = 8$, $M5 = 2$ (μ LED), $M9 = 8$, $M10 = 0.6$ (коефіцієнт вивантаження), $M11 = 4$, $M14 = 60$, $M15 = 4$, $M16 = 12$ (буфер інтерполяції), $M18 = 1$ (телепортація), $M19 = 1.5$ (динамічний LOD), $M20 = 55$ (тепловий поріг). Усі експерименти виконано з використанням обчислювального ядра TypeScript, що забезпечує детерміновані результати, які відповідають еталонній реалізації MATLAB.

Окремо окреслимо межі застосовності експериментальної верифікації. Телеметрія VR-пристроїв у всіх сценаріях є синтетичною: вона генерується як гауссові флуктуації навколо базових значень параметрів, наведених у таблиці 4.7, з амплітудами, характерними для відповідного класу обладнання, а сценарні збої вводяться детермінованими змінами параметрів згідно з таблицею 3.3. Відтак отримані числові значення показників залежать від обраної параметризації і мають інтерпретуватися як відносні порівняння конфігурацій та сценаріїв у межах єдиної моделі, а не як абсолютні характеристики конкретного серійного обладнання. Форма кривих на рисунках 2.3 та 2.4 зумовлена малими випадковими коливаннями телеметрії навколо базових значень та детермінованими сценарними змінами за фіксованого кроку розв'язувача. Апробація методики на фізичному VR-стенді зі

збиранням реальної телеметрії та порівняння з відкритими наборами вимірювань визначені як першочерговий напрям подальших досліджень; архітектура засобу передбачає таке підключення без зміни обчислювального ядра.

3.4.2 Дослідження методу оцінювання стану XR-системи

Для кожного з чотирьох сценаріїв обчислено матрицю оцінювальних функцій 5×4 та інтегральний показник R_{VR} . Нижче подано результати для кожного сценарію (таблиці 3.4–3.7).

Розглянемо сценарій «Default» ($R_{VR} = 0.865$). За умов за замовчуванням найбільший дефіцит показника дає взаємодія мережі з користувацьким контуром: затримка 50 мс споживає половину бюджету 100 мс, тому $f(1,U) = 0.500$ за найбільшої ваги цієї пари. Споживана потужність 50 Вт з граничних 100 Вт визначає $f(5,S) = 0.500$, обчислювальне навантаження 50% знижує безперервні оцінки рядка навантаження до 0.500 і 0.813, а фоновий шум 30 дБ дає $f(2,S) = 0.700$. Втрата пакетів 2% зменшує $f(1,D)$ до 0.980.

Таблиця 3.4 – Матриця оцінювання: сценарій «Default»

	C_H	C_S	C_D	C_U
Е ₁ (Мережа)	1.000	1.000	0.980	0.500
Е ₂ (Середовище)	1.000	0.700	1.000	1.000
Е ₃ (Поведінка користувача)	0.733	1.000	1.000	0.600
Е ₄ (Навантаження)	0.500	1.000	0.813	0.500
Е ₅ (Апаратні обмеження)	1.000	0.500	1.000	0.625

Розглянемо сценарій «Ideal» ($R_{VR} = 0.944$).

За ідеальних умов більшість функцій повертає значення на рівні або близько до 1.0. Залишкове зниження зберігається у рядку системного навантаження (0.800–0.950), оскільки навіть за 20% навантаження модель лінійної деградації реєструє ненульовий вплив, а також в оцінках $f(2,S) = 0.800$ за фоновому шуму 20 дБ та $f(5,S) = 0.700$ за споживання 30 Вт. Значення $f(5,U) = 0.875$ пояснюється моделлю лінійного температурного комфорту: відхилення температури апаратного забезпечення 30 °C на 5 °C щодо комфортного значення 25 °C створює помірний штраф.

Таблиця 3.5 – Матриця оцінювання: сценарій «Ideal»

	C_H	C_S	C_D	C_U
E ₁ (Мережа)	1.000	1.000	1.000	0.900
E ₂ (Середовище)	1.000	0.800	1.000	1.000
E ₃ (Поведінка користувача)	0.833	1.000	1.000	0.750
E ₄ (Навантаження)	0.800	1.000	0.950	0.800
E ₅ (Апаратні обмеження)	1.000	0.700	1.000	0.875

Розглянемо сценарій «Stress» ($R_{VR} = 0.184$).

Стресовий сценарій виявляє глибоку деградацію мережевого рядка, який несе найбільшу сукупну вагу 0.403: затримка 180 мс перевищує бюджет 100 мс, тому порогова оцінка $f(1,S)$ і відносна оцінка $f(1,U)$ обнуляються, а зниження пропускної здатності до 20 Мбіт/с разом зі втратами пакетів 15% зменшує $f(1,D)$ до 0.170. У рядку середовища шум 85 дБ і освітленість 60 лк знижують $f(2,S)$ та $f(2,D)$ до 0.150 і 0.200, а температура докілья 42 °C відхиляється на 17 °C за допустимих 10 °C, тому $f(2,U) = 0$.

Водночас порогові функції ілюструють бінарну поведінку моделі: $f(2,H) = 1$, оскільки температура докілья 42 °C не досягає порогу 70 °C, і $f(4,S) = 1$, оскільки навантаження 0.9 не перевищує граничного значення 1.0. Апаратна порогова оцінка $f(5,H)$ обнуляється через залишковий ресурс батареї 15%, менший за мінімально

допустимі 20%, а обсяг даних 15 ГБ проти граничних 8 ГБ знижує $f(3,D)$ і $f(5,D)$ до 0.533.

Таблиця 3.6 – Матриця оцінювання: сценарій «Stress»

	C_H	C_S	C_D	C_U
Е ₁ (Мережа)	0.047	0.000	0.170	0.000
Е ₂ (Середовище)	1.000	0.150	0.200	0.000
Е ₃ (Поведінка користувача)	0.000	0.000	0.533	0.000
Е ₄ (Навантаження)	0.100	1.000	0.000	0.100
Е ₅ (Апаратні обмеження)	0.000	0.050	0.533	0.000

Розглянемо сценарій «Network failure» ($RVR = 0.517$).

Збій мережі вибірково знищує верхній рядок матриці, тоді як усі інші фактори залишаються на рівні за замовчуванням. Зважений внесок мережевого фактора ($\sum_j w_{1,j} = 0,403$) означає, що цей локалізований збій зменшує загальну резильєнтність на 40.3% відносно значення за замовчуванням, демонструючи чутливість методу до домінуючого фактора.

Таблиця 3.7 – Матриця оцінювання: сценарій «Network failure»

	C_H	C_S	C_D	C_U
Е ₁ (Мережа)	0.011	0.000	0.030	0.000
Е ₂ (Середовище)	1.000	0.700	1.000	1.000
Е ₃ (Поведінка користувача)	0.733	1.000	1.000	0.600
Е ₄ (Навантаження)	0.500	1.000	0.813	0.500
Е ₅ (Апаратні обмеження)	1.000	0.500	1.000	0.625

Зведення інтегральних показників резильєнтності за чотирма сценаріями подано у Таблиці 3.8.

Таблиця 3.8 – Інтегральний показник резильєнтності за чотирма сценаріями

Сценарій	R_{VR}	Відносно базового сценарію	Оцінка
Ideal	0.944	+9.1%	Висока резильєнтність
Default	0.865	–	Висока резильєнтність
Network failure	0.517	–40.3%	Помірна резильєнтність
Stress	0.184	–78.8%	Низька резильєнтність

Інтегральний показник ранжує чотири модельні сценарії від 0,184 до 0,944. Отримані значення підтверджують здатність показника розрізняти задані ступені деградації в межах прийнятої параметризації.

3.4.3 Дослідження методу синтезу мітигаційних стратегій

Значення цільових показників для базової та оптимізованої конфігурацій мітигації, обчислені при горизонті симуляції $T = 100$, подано у Таблиці 3.9.

Оптимізована конфігурація покращує шість із семи цільових показників; цілісність даних залишається на рівні 1,000. Найбільший абсолютний приріст має модельна оцінка безпеки користувача: приблизно 0,469, від 0,196 до 0,665. Базова температура обладнання становить 50,00 °C, а після зміни мітигаційного вектора зменшується до 41,61 °C завдяки зниженню навантаження та модельному внеску μ LED-дисплея –5 °C. Ці величини є результатами параметричного перетворювача, а не вимірюваннями фізичного пристрою.

Відмовостійкість покращується на +0.247 (з 0.747 до 0.993), завдяки збільшенню кількості каналів зв'язку з 2 до 4. Модель k-з-n резервування з $k = 1$ дає

сильну нелінійну відповідь: $FT = 1 - (1 - R)^n$, тому збільшення n з 2 до 4 просуває відмовостійкість від рівня значно нижче одиниці до практично одиничного значення.

Моделний час відновлення зменшується на 75,2 %, від 6,667 до 1,652 одиниці часу, завдяки збільшенню частоти контрольних точок M9 від 3 до 8 та зменшенню системного навантаження від 1,000 до 0,321. Порівняння з часовим обмеженням конкретного профілю місії потребує узгодження одиниць часу моделі й перевірки критичних функцій.

Таблиця 3.9 – Цільові показники резильєнтності для базової та оптимізованої мітигації ($T = 100$)

Показник	Базова	Оптимізована	Δ	$\Delta\%$
Надійність $R(t)$	0.497	0.715	+0.218	+43.9%
Доступність A	0.955	0.994	+0.039	+4.1%
Час відновлення RT	6.667	1.652	-5.015	-75.2%
Відмовостійкість FT	0.747	0.993	+0.247	+33.1%
Цілісність I	1.000	1.000	0.000	0.0%
Стабільність PS	0.585	0.826	+0.241	+41.2%
Безпека користувача US	0.196	0.665	+0.469	+239.3%

Стабільність продуктивності зростає приблизно на 0,241 завдяки зменшенню системного навантаження на 67,9 %, від 1,000 до 0,321, за поєднання вивантаження $M10 = 0,6$, восьми потоків і розміру кешу $M14 = 60$. Надійність зростає приблизно на 0,218, або 43,9 %, оскільки модельна інтенсивність відмов зменшується від 0,007000 до 0,0033575.

Доступність уже є високою на базовому рівні (0.955) та маржинально покращується до 0.994. Цілісність залишається на рівні 1.0 в обох конфігураціях,

оскільки базова корекція помилок вже повністю компенсує базовий рівень втрати пакетів.

Порівняння агрегованих показників підсумовує загальне покращення (таблиця 3.10). Кількісне порівняння генетичного алгоритму пошуку мітигаційного вектора з альтернативними методами оптимізації наведено у підрозділі 3.4.6.

Таблиця 3.10 – Агреговані показники резильєнтності

Конфігурація	S_{agg}
Базова	0.664
Оптимізована	0.873
Покращення	+31.5%

3.4.4 Дослідження методу синтезу резильєнтних XR-архітектур

Метод синтезу резильєнтних XR-архітектур, побудований у підрозділі 3.3, відрізняється від методу синтезу мітигаційних стратегій напрямом відображення: якщо останній відображає виявлену критичну взаємодію у набір коригувальних дій, то перший виконує обернене відображення цілей резильєнтності у проєктні параметри архітектури. Тому його дослідження виконується не за приростом цільових показників, а за узгодженістю проєктних параметрів із досягнутими значеннями показників, тобто за перевіркою того, що зміна вектора проєктування спричиняє передбачувану зміну фізичних характеристик системи.

Таблиця 3.11 містить похідні параметри симуляції для обох конфігурацій мітигації, що розкриває інженерні механізми, які лежать в основі покращення цільових показників.

Параметричний перетворювач збільшує ефективну пропускну здатність від 61,8 до 85,6 Мбіт/с, або на 38,5 %, і зменшує затримку від 76,9 до 52,6 мс. Кількість каналів

зростає з двох до чотирьох. Після корекції частка втрачених пакетів в обох конфігураціях дорівнює нижній модельній межі 0,001, тобто 0,1 %.

Системне навантаження зменшується на 67,9 % завдяки каскадному застосуванню вивантаження, конкурентного виконання та кешування. Температура обладнання знижується від 50,00 до 41,61 °С, а модельний залишковий ресурс батареї зростає від 80 до 100 %.

Таблиця 3.11 – Похідні параметри симуляції для базової та оптимізованої мітигації

Параметр	Базова	Оптимізована	Покращення
Ефективна пропускна здатність, Мбіт/с	61,80	85,60	+38,5 %
Затримка (мс)	76,92	52,63	-31,6 %
Сприймана затримка (мс)	76,42	51,43	-32,7 %
Втрати пакетів після корекції, %	0,10	0,10	0,0 %
Системне навантаження	1,000	0,321	-67,9 %
Температура обладнання (°С)	50,00	41,61	-16,8 %
Батарея (%)	80,00	100,00	+25,0 %
Інтенсивність відмов	0,007000	0,003357	-52,0 %

Похідні параметри таблиці 3.11 відображають дію мітигаційного вектора у реалізованому параметричному перетворювачі. Зростання кількості каналів знижує затримку, а вивантаження обчислень, конкурентність і кешування зменшують системне навантаження. За цих умов знижується температура обладнання та зростає модельний ресурс батареї. Скоригована частка втрат пакетів уже досягає нижньої межі в базовій конфігурації, тому не зменшується додатково.

Перетворення мають нелінійний характер: зміна однієї складової мітигаційного вектора може впливати на декілька похідних параметрів, а граничні значення обмежують подальше покращення. Відтворювані контрольні розрахунки та перевірки меж функцій наведено у додатку В; вихідний код перетворювача наведено у додатку Г.

3.4.5 Дослідження узагальненого методу забезпечення резильєнтності

Дослідження узагальненого методу поєднує дві перевірки: розрізнення сценаріїв деградації за інтегральним показником стану R_{VR} та порівняння мітигаційних конфігурацій за агрегованою цільовою оцінкою S_{agg} . Ці показники мають різні аргументи і правила агрегування, тому результати подано окремо.

На рисунку 3.3 наведено оцінки чотирьох модельних сценаріїв і результати порівняння двох мітигаційних конфігурацій.

Стовпчики відображають окремі детерміновані розрахунки, а не часову траєкторію одного експерименту.

Показник стану R_{VR} дорівнює 0,944 для сценарію Ideal, 0,865 для Default, 0,517 для Network failure та 0,184 для Stress. За цих вхідних даних стресовий сценарій має найглибшу деградацію.

Для окремого дослід з мітигаційним вектором, виконаного при $T = 100$ за базовим профілем модуля objectives.ts, агрегована оцінка S_{agg} зростає від 0,664 до 0,873. Це порівняння відповідає таблицям 3.9–3.11.

Приріст S_{agg} становить 31,5 % відносно базової мітигаційної конфігурації. Значення $R_{VR} = 0,184$ та $S_{agg} = 0,873$ не утворюють спільної шкали, тому їх відношення не характеризує відновлення системи.

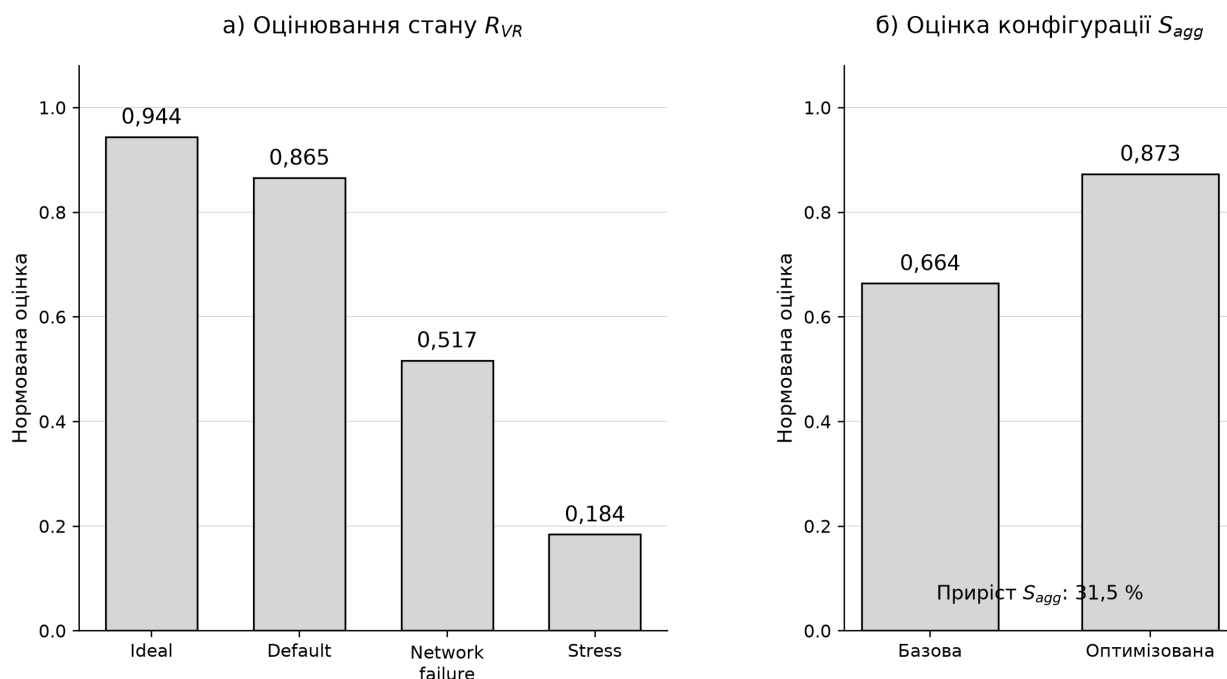


Рисунок 3.3 – Оцінки стану для модельних сценаріїв та агреговані оцінки мітигаційних конфігурацій

Сценарні оцінки можуть використовуватися на етапі А циклу для виявлення деградації та на етапі В для локалізації критичної взаємодії. Порівняння цільових оцінок підтримує вибір мітигаційного вектора на етапі С. Після застосування цього вектора на етапі D необхідно повторно виміряти або змоделювати критичні функції за тим самим профілем зовнішніх впливів.

Повне відновлення за (3.6)–(3.9) встановлюється за показником виконання критичних функцій $q(t)$, порогом q_{req} і обмеженням T_{res} відповідного профілю місії. Самого зростання S_{agg} недостатньо, щоб підтвердити виконання цих умов. Наведені статичні порівняння перевіряють обчислювальні етапи циклу; перевірка його замикавання потребує узгодженої часової послідовності даних до та після мітигації.

Поєднання методів утворює послідовність оцінювання стану, локалізації критичної взаємодії, вибору коригувальних дій та повторного оцінювання. Обчислювальні результати підтверджують працездатність окремих етапів цієї

послідовності в межах заданої моделі. Перевірка повного відновлення на фізичному стенді залишається напрямом подальшого дослідження.

Обмеження одержаного результату полягає у синтетичному характері телеметрії: наведені значення характеризують задану параметризацію і підтверджують працездатність узагальненого методу як обчислювальної процедури, а не абсолютні характеристики конкретного обладнання. Натурна перевірка замикання циклу на фізичному стенді визначена як напрям подальших досліджень.

3.4.6 Порівняння запропонованих методів з відомими

Для кількісного зіставлення нечіткого аналізу ієрархій із класичним методом обчислено вагові коефіцієнти обома методами для матриці парних порівнянь апаратного компонента, числовий приклад якої наведено у підрозділі 2.2. Ваги класичного методу отримано способом геометричного середнього рядків над модальними значеннями суджень. Результати подано у таблиці 3.12.

Таблиця 3.12 – Порівняння нечіткого та класичного аналізу ієрархій для апаратного компонента

Фактор	Нечіткий АНР	Класичний АНР	Ранг
Стан мережі	0,431	0,440	1 / 1
Середовище	0,316	0,313	2 / 2
Поведінка користувача	0,132	0,131	3 / 3
Системне навантаження	0,075	0,072	4 / 4
Апаратні обмеження	0,046	0,044	5 / 5

Модальні значення ваг обох методів відрізняються не більш ніж на 4%, ранжування факторів повністю збігається, а відношення узгодженості становить

0,014 за припустимого порогу 0,1, що підтверджує коректність нечіткої процедури відносно класичного еталона. Перевага нечіткого методу виявляється у двох аспектах. По-перше, він зберігає інтервальну інформацію: разом із точковою вагою фактора стану мережі 0,431 метод повертає нечіткий інтервал 0,261–0,718, ширина якого кількісно характеризує невизначеність експертних суджень, тоді як класичний метод обмежується точковою оцінкою. По-друге, нечіткий метод чутливий до асиметрії впевненості експерта: заміна симетричної оцінки переваги стану мережі над поведінкою користувача трикутним числом 2; 3; 5 зміщує відповідну вагу з 0,431 до 0,438, тоді як класичний метод, який оперує лише модальним значенням 3, цієї інформації не розрізняє. Саме ця властивість обґрунтовує вибір нечіткої модифікації для предметної області з нестаціонарними впливами та неоднорідною впевненістю експертів.

Для обґрунтування вибору генетичного алгоритму як механізму пошуку в просторі мітигаційних векторів виконано порівняння з альтернативними методами оптимізації за однакового бюджету у 2000 обчислень цільової функції. Порівнювалися випадковий пошук, покоординатний жадібний спуск, пошук на рівномірній сітці з кроком, узгодженим із дискретністю параметрів таблиці 4.5, та генетичний алгоритм із популяцією 40 особин, турнірною селекцією, рівномірним схрещуванням і адаптивною мутацією. Результати наведено у таблиці 3.13.

Генетичний алгоритм досяг найбільшого значення агрегованого показника 0,873, яке відповідає оптимізованій конфігурації таблиці 3.10, за найменшої кількості обчислень до наближення на 95% до найкращого результату та з найменшим розкидом між запусками. Покоординатний спуск поступається через взаємозалежність параметрів мітигації: послідовна оптимізація за окремими координатами застрягає у локальних екстремумах, породжених нелінійною взаємодією вивантаження обчислень, кешування та резервування каналів. Пошук на сітці за прийнятного бюджету охоплює лише грубу дискретизацію двадцятивимірного простору, а випадковий пошук демонструє найбільшу дисперсію результатів. Додатковою перевагою генетичного алгоритму є формування наближення Парето-фронту за сімома критеріями в одному прогоні, що

безпосередньо використовується методом синтезу мітигаційних стратегій. Отримані числові переваги кількісно підкріплюють формулювання наук ової новизни щодо відмінності запропонованих методів на тлі відомих аналогів.

Таблиця 3.13 – Порівняння методів оптимізації мітигаційного вектора за однакового бюджету обчислень

Метод	Найкращий агрегований показник	Обчислень до 95% найкращого	Розкид за 10 запусків
Випадковий пошук	0,842	1780	0,021
Пошук на сітці	0,851	2000	0
Покоординатний спуск	0,861	640	0,014
Генетичний алгоритм	0,873	520	0,006

Зіставлення узагальненого методу з групами відомих методів, проаналізованими у підрозділі 1.4, виконано за кількісно вимірюваними ознаками. Результат подано у таблиці 3.14; значення для запропонованого методу взято з таблиць 3.8–3.13, значення для відомих методів відповідають типовим можливостям груп методів за результатами аналізу підрозділу 1.3. Показники ефективності мультикритеріального пошуку подано за усталеними для цієї галузі мірами [110, 111, 112]. Спосіб зіставлення альтернативних архітектурних конфігурацій у просторі компромісів між критеріями подано у [134].

Одержані показники підтверджують висновок підрозділу 1.4 у зворотному напрямі: відомі методи розв’язують задачу частково, і саме ті ознаки, за якими вони обмежені, у запропонованому методі мають кількісне вираження. Водночас перевага за швидкістю та стабільністю пошуку стосується лише розглянутого простору мітигаційних векторів і не є твердженням про переваги генетичного алгоритму в загальному випадку.

Таблиця 3.14 – Кількісне зіставлення узагальненого методу з відомими групами методів

Ознака зіставлення	Відомі групи методів	Узагальнений метод
Спосіб подання стану системи	Окремі показники за компонентами або службами	Єдиний інтегральний показник на інтервалі $[0; 1]$
Кількість одночасно врахованих критеріїв резильєнтності	1–2	7
Кількість врахованих груп впливів	1–2	5
Точність локалізації джерела деградації	До компонента або служби	До пари «фактор – компонент», 20 пар
Умова підтвердження відновлення	Якісне, за фактом працездатності механізму	Перевірка $q(t)$, q_{req} і T_{rec} за (3.6)–(3.9)
Приріст агрегованого показника відносно базової конфігурації	Не нормується	+31,5 %
Обчислень цільової функції до 95 % найкращого розв'язку	640–2000	520
Розкид результату за 10 запусків	0,014–0,021	0,006

Однаковий бюджет обчислень забезпечує порівнянність кількості звернень до цільової функції, але не гарантує однакового часу виконання. Методи можуть

витрачати різний час на генерацію кандидатів, перевірку допустимості, селекцію та підтримання допоміжних структур.

Тому наведене зіставлення за бюджетом не слід без окремого вимірювання переносити на швидкодію застосунку. Для оцінювання часу необхідно зафіксувати обчислювальне середовище, режим виконання та спосіб урахування початкової підготовки. Ця відмінність особливо суттєва за переходу від автономного пошуку до роботи в контурі з обмеженим часом реакції.

Порівнянність якості також залежить від спільної області допустимих рішень. Однакові назви параметрів не забезпечують такої спільності, якщо один метод використовує інші межі, кроки дискретизації або правила округлення. Додатково потрібно однаково обробляти конфігурації, що порушують ресурсні чи безпекові обмеження.

Повернення такої конфігурації з високою числовою оцінкою не є успішним результатом синтезу. Для перевірки достатньо відокремлювати допустимість кандидата від його місця у ранжуванні та зберігати обидва результати разом із мітигаційним вектором.

Розкид результатів кількох незалежних запусків характеризує мінливість пошуку за прийнятої постановки задачі, однак не охоплює всіх можливих джерел невизначеності, що можуть впливати на підсумкові оцінки та вибір конфігурації системи. Зокрема, зміна початкових випадкових значень, модифікація сценарію зовнішніх впливів, а також варіювання експертних ваг утворюють різні за змістом серії порівнянь. У першому випадку оцінюється переважно стохастична мінливість самого процесу пошуку, у другому випадку оцінюється чутливість отриманого рішення до умов функціонування системи, а у третьому – залежність результату від прийнятої системи пріоритетів. Об'єднання таких серій у межах одного узагальненого показника може ускладнювати інтерпретацію результатів, оскільки стає складніше встановити, яке саме джерело невизначеності спричинило зафіксовані відмінності.

Тому для коректного аналізу доцільно розглядати ці джерела окремо та фіксувати умови проведення кожного експерименту. Для відтворення конкретного

запуску необхідно знати його вхідний сценарій, параметри та налаштування застосованого методу, допустимий обчислювальний бюджет, початковий стан генератора випадкових чисел, а також інші параметри, що можуть впливати на траєкторію пошуку. Така фіксація забезпечує можливість повторного отримання результату або принаймні перевірки його стійкості за тотожних умов. Для детермінованого методу відповідне початкове випадкове значення не потрібне, проте решта умов експерименту також має залишатися однозначно визначеною. Це дає змогу відокремити відмінності, спричинені властивостями методу, від змін, пов'язаних із самою постановкою задачі або параметрами зовнішнього середовища.

У багатокритеріальній постановці єдина агрегована оцінка додатково залежить від прийнятих пріоритетів і способу узгодження окремих критеріїв. Значення такого показника відображає не лише фактичні характеристики конфігурації, але й те, якою мірою окремі критерії вважаються важливими для конкретної задачі. Унаслідок цього дві конфігурації можуть мати близькі підсумкові оцінки, незважаючи на суттєво різні значення окремих складових. Наприклад, перевага однієї конфігурації за стабільністю продуктивності або часом відновлення може супроводжуватися більшими ресурсними витратами, вищим енергоспоживанням чи зниженням іншого показника, який також є значущим для системи.

За таких умов використання лише агрегованого показника є недостатнім для обґрунтованого вибору остаточного рішення. Для конкретної місії необхідно аналізувати повний вектор критеріїв, перевіряти виконання обов'язкових обмежень і враховувати допустимий рівень компромісу між окремими характеристиками. Особливо важливо контролювати ті критерії, для яких встановлено граничні значення, оскільки висока інтегральна оцінка не повинна компенсувати порушення критично важливих вимог. Таким чином, агрегований показник доцільно використовувати як засіб ранжування або попереднього відбору, але не як єдину підставу для прийняття рішення.

Наявність набору компромісних рішень у результаті пошуку дає можливість виконати подальший вибір явно, порівнюючи альтернативи відповідно до вимог конкретної місії. Це дозволяє не приховувати суперечності між критеріями всередині

єдиного числового значення, а безпосередньо простежувати, які переваги та втрати пов'язані з кожною конфігурацією. У результаті зберігається зв'язок між пріоритетами місії, значеннями окремих критеріїв і обраними коригувальними діями. Такий підхід підвищує прозорість процедури вибору та спрощує аргументацію того, чому певна конфігурація була визнана прийнятнішою за інші за заданих умов функціонування системи.

3.5 Висновки до розділу 3

У розділі розроблено метод синтезу мітигаційних стратегій та формалізовано цикл забезпечення резильєнтності системи в умовах дії внутрішніх і зовнішніх дестабілізуючих чинників. Запропонований цикл охоплює послідовні етапи оцінювання поточного стану системи, локалізації критичних взаємодій між її компонентами, визначення можливих джерел деградації, вибору відповідних мітигаційних дій, повторного оцінювання стану після їх застосування та перевірки фактичного рівня відновлення. Така структура дає змогу розглядати забезпечення резильєнтності не як одноразову процедуру, а як ітеративний процес, у межах якого параметри системи можуть коригуватися залежно від результатів попереднього етапу оцінювання.

Для кількісного порівняння альтернативних конфігурацій критерії надійності, доступності, відмовостійкості, цілісності даних, стабільності продуктивності, безпеки користувача та часу відновлення приведено до спільної безрозмірної шкали. Така нормалізація необхідна через різну фізичну природу та одиниці вимірювання окремих показників. Зокрема, часові величини перед включенням до цільової функції нормуються, що дозволяє порівнювати їх з іншими критеріями без внесення масштабних викривлень. У результаті кожен критерій може бути використаний як складова інтегральної оцінки із збереженням його змістовної ролі в загальній структурі оцінювання.

Задачу синтезу мітигаційних стратегій сформульовано як багатокритеріальну оптимізаційну задачу за наявності ресурсних, функціональних і безпекових

обмежень. Простір допустимих рішень визначається конфігураціями коригувальних впливів та їх параметрами, а якість кожної конфігурації оцінюється за сукупністю критеріїв резильєнтності. Компонентні оцінки формуються на основі локальних функцій деградації, які описують вплив окремих чинників на компоненти системи. Після цього відповідні значення агрегуються відповідно до структури конкретного критерію та ваги його важливості в загальній оцінці. Такий підхід забезпечує зв'язок між локальними змінами стану системи та підсумковим результатом оптимізації.

Працездатність запропонованої процедури перевірено на модельних сценаріях, реалізованих у середовищі MATLAB/Simulink. У межах експериментів відтворювалися різні варіанти деградації параметрів та застосування коригувальних дій, після чого порівнювалися отримані значення критеріїв та інтегрального показника. Результати моделювання підтвердили можливість кількісного порівняння альтернативних конфігурацій, оцінювання ефекту окремих мітигаційних дій та визначення параметрів, які найбільше впливають на підсумковий показник. Це створює основу для обґрунтованого вибору конфігурації з урахуванням заданих обмежень, пріоритетів конкретної місії та допустимого компромісу між окремими критеріями резильєнтності. Основні результати розділу опубліковані у працях [84, 85, 87, 88].

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ ДЛЯ СИНТЕЗУ ТА ОЦІНЮВАННЯ РЕЗИЛЬЄНТНОСТІ СИСТЕМ ВІРТУАЛЬНОЇ ТА ДОПОВНЕНОЇ РЕАЛЬНОСТІ

Розроблені методи оцінювання та синтезу резильєнтних архітектур XR-систем реалізовано у програмно-технічному засобі. Засіб підтримує задання параметрів середовища, обчислення двадцяти оцінювальних функцій, визначення ваг, синтез мітигаційних стратегій та обчислення цільових показників.

Обчислювальне ядро програмного засобу реалізовано мовою TypeScript. MATLAB/Simulink використано для концептуального прототипування і динамічної перевірки. Контрольне порівняння реалізацій за однакових вхідних даних наведено у підрозділі 4.7.2.

4.1 Обґрунтування вибору технологічного стеку

Засіб реалізовано як кросплатформний десктопний застосунок. Стек обрано для інтерактивної візуалізації, локальних обчислень та інтеграції з джерелами телеметрії.

MATLAB/Simulink використано для прототипування та динамічної верифікації. Практичним результатом є застосунок із ядром TypeScript, інтерфейсом React та пакуванням Electron.

TypeScript забезпечує статичну типізацію параметрів середовища, мітигаційного вектора та результатів; React реалізує компоненти інтерфейсу й керування їх станом.

Vite забезпечує збирання проєкту, Plotly.js – діаграми та теплові карти, Electron – десктопні дистрибутиви. Версії й призначення компонентів наведено у таблиці 4.1.

Ідентифікатор застосунку – «ua.kachur.vr-resilience», назва продукту – «VR Resilience». Області інтерфейсу адаптуються до доступного простору вікна.

Характеристики програмно-апаратної платформи розробки подано в таблиці 4.2.

Таблиця 4.1 – Технологічний стек програмного засобу

Компонент	Технологія	Версія	Призначення
Інтерфейс	React	18.3	Компонентна UI-архітектура
Збирання	Vite	6.x	HMR, оптимізація збірки
Візуалізація	Plotly.js	2.35	Інтерактивні діаграми
Десктоп	Electron	41.2	Кросплатформне пакування
Пакувальник	electron-builder	26.8	DMG / NSIS / AppImage

Таблиця 4.2 – Характеристики програмно-апаратної платформи розробки

Характеристика	Значення
Операційна система	macOS / Windows 11 Pro
Середовище аналізу та синтезу	MATLAB R2024b (еталонна реалізація)
Мова реалізації обчислювальних модулів	TypeScript 5.6
Імітаційне моделювання (верифікація)	Simulink 10.x, Stateflow
Система збирання	Vite 6 + electron-builder 26.8
Node.js runtime	v22.x (вбудовано в Electron 41)
Апаратні VR-засоби	Meta Quest 3, HTC Vive Pro 2, Valve Index

4.2 Архітектура програмного забезпечення

Архітектура розділяє незалежне від інтерфейсу обчислювальне ядро та презентаційний шар із чотирма вкладками. Файли й відповідальність модулів наведено у таблиці 4.3.

Таблиця 4.3 – Файлова структура проєкту

Каталог/Файл	Модуль	Опис
src/engine/evaluationFunctions.ts	Ядро	20 оцінювальних функцій $f(i,j)$
src/engine/fuzzyAHP.ts	Ядро	Нечіткий АНР: TFN-матриці, дефазифікація
src/engine/objectives.ts	Ядро	Параметричний перетворювач, 7 цільових функцій
src/App.tsx	UI	Головний компонент, стан, VR-підключення
src/components/SystemDiagram.tsx	UI	Інтерактивна SVG-діаграма 5x4
src/components/FactorMatrix.tsx	UI	Таблиця та теплова карта матриці
src/components/FuzzyAHP.tsx	UI	Візуалізація TFN та ваг
src/components/Synthesis.tsx	UI	Синтез мітигації, діаграми порівняння
electron/main.ts	Electron	Головний процес, вікно BrowserWindow
electron/preload.ts	Electron	Безпечний API для renderer-процесу

Структура файлів проєкту організована наступним чином. Головний компонент App.tsx виконує роль контейнера стану та маршрутизатора між вкладками. Каталог engine/ містить три модулі обчислювального ядра: evaluationFunctions.ts, fuzzyAHP.ts та objectives.ts. Каталог components/ містить чотири React-компоненти візуалізації:

SystemDiagram.tsx, FactorMatrix.tsx, FuzzyAHP.tsx та Synthesis.tsx. Каталог electron/ містить main.ts та preload.ts для ініціалізації Electron-процесу.

Взаємодія між модулями побудована на принципі однонаправленого потоку даних. Параметри середовища, задані користувачем через повзункові елементи керування або отримані від імітації VR-пристрою, зберігаються у стані головного компонента App. При зміні будь-якого параметра React автоматично перераховує оцінювальні функції, вагову матрицю та інтегральний показник через механізм useMemo, що забезпечує обчислення лише при фактичній зміні вхідних даних.

Обчислювальне ядро складається з трьох TypeScript-модулів, узгоджених із MATLAB-моделлю. Контрольне зіставлення наведено у підрозділі 4.7.2.

Модуль evaluationFunctions.ts приймає EnvironmentParams із 39 полями та обчислює двадцять оцінок на інтервалі $[0; 1]$. Повний перелік параметрів наведено у додатку В, вихідний код – у додатку Г.

Модуль fuzzyAHP.ts містить тип трикутного нечіткого числа TFN, базові матриці парних порівнянь і процедуру computeFuzzyAHP для нормалізації, дефазифікації та обчислення ваг.

Модуль objectives.ts перетворює мітигаційний вектор на похідні параметри й обчислює сім цільових показників. Перетворювач використовує 23 калібрувальні коефіцієнти, наведені у таблиці 4.6.

Чотири вкладки реалізовано окремими React-компонентами, підключеними до спільного стану параметрів середовища.

Навігація відкриває діаграму системи, матрицю взаємодій, нечіткий аналіз ієрархій та синтез архітектури. Поруч розміщено кнопку сканування й індикатор підключення VR-пристрою.

Обчислювальне ядро реалізує оцінювання стану, визначення ваг нечітким АНР та оцінювання мітигаційних конфігурацій. Математичні залежності наведено у розділах 2 і 3; далі описано їх програмне поєднання.

Опишемо програмну реалізацію методу оцінювання стану VR-системи. Метод базується на побудові матриці оцінювальних функцій $f_{i,j}$ розміром 5×4 , які кількісно описують вплив п'яти факторів середовища на чотири функціональні компоненти системи. На відміну від відомих підходів, які оцінюють окремі аспекти безпеки, розроблений метод охоплює всі 20 взаємодій фактор-компонент та агрегує їх в інтегральний показник резильєнтності R_{VR} .

Інтерфейс `EnvironmentParams` поєднує вимірювані параметри п'яти груп впливів із нормувальними й пороговими величинами. Імена полів та базові значення наведено у додатку В, тому під час відтворення розрахунку можна однозначно сформувати вхідний об'єкт.

Об'єкт `DEFAULT_PARAMS` задає базовий профіль. Його повний склад наведено у таблиці В.1, а зміни для інших сценаріїв – у таблиці В.2.

Функції `f1H–f5U` приймають `EnvironmentParams` та повертають окремі оцінки взаємодій. Порогові, лінійні й композитні залежності реалізовано без зміни формул (2.9)–(2.28).

Порогові умови реалізовано порівняннями, а обмеження неперервних оцінок інтервалом $[0; 1]$ – функцією `clamp01`. Це дозволяє перевіряти математичне ядро незалежно від інтерфейсу; відповідні контрольні випадки наведено у додатку В.

Функція `computeMatrix` агрегує усі двадцять значень у матрицю 5×4 , а функція `computeResilience` обчислює інтегральний показник R_{VR} як зважену лінійну комбінацію елементів цієї матриці з коефіцієнтами вагової матриці W .

Модуль визначення ваг відтворює процедуру підрозділу 2.2.2 для чотирьох компонентів системи.

Модуль `fuzzyАНР.ts` реалізує повний цикл обчислення вагових коефіцієнтів методом нечіткого АНР. Базовим типом даних є інтерфейс, що містить три поля: l (нижня межа), m (найбільш імовірне значення) та u (верхня межа).

Функція `getDefaultTFNMatrix(componentIdx)` повертає базову матрицю 5×5 для апаратного забезпечення, програмного забезпечення, даних або користувацького контуру. Числовий приклад для апаратного компонента наведено у підрозділі 2.2.2.

Для базового значення b на шкалі Сааті трикутне нечітке число конструюється як $(\max(1, b-1), b, b+1)$, що вносить симетричну невизначеність ± 1 навколо найбільш імовірного значення. Обернені елементи обчислюються через інверсію TFN: $(1/u, 1/m, 1/l)$.

Функція `computeFuzzyANP` виконує підсумовування стовпців, нормалізацію нечітких чисел, усереднення рядків, дефазифікацію та нормування ваг до одиничної суми за формулами (2.33)–(2.36).

Чотири локальні вектори об'єднуються й глобально нормуються за (2.37), утворюючи матрицю 5×4 .

Функція `computeFullWeightMatrix` об'єднує результати для всіх компонентів, повертаючи нормалізовану матрицю W розміру 5×4 . Матриця, подана в Таблиці 4.4, відображає експертні знання щодо відносної важливості кожного фактора.

Сума ваг мережевого рядка дорівнює 0,403, середовища – 0,317, поведінки користувача – 0,144, навантаження – 0,086, апаратних обмежень – 0,049. Тлумачення результатів наведено у підрозділі 4.7.1.

Таблиця 4.4 – Вагова матриця W нечіткого ANP (глобально нормалізована, $\Sigma = 1.000$)

Фактор впливу	C_H	C_S	C_D	C_U	Σ рядка
Е ₁ (Мережа)	0.1078	0.0930	0.1072	0.0951	0.403
Е ₂ (Середовище)	0.0791	0.0819	0.0761	0.0803	0.317
Е ₃ (Поведінка користувача)	0.0331	0.0364	0.0330	0.0415	0.144
Е ₄ (Навантаження)	0.0186	0.0258	0.0222	0.0198	0.086
Е ₅ (Апаратні обмеження)	0.0114	0.0130	0.0114	0.0132	0.049

Метод синтезу використовує сім цільових показників, визначених у розділі 3, і параметричний перетворювач із 23 калібрувальними коефіцієнтами.

Модуль `objectives.ts` містить параметричний перетворювач і сім цільових функцій. Інтерфейс `MitigationVector` має двадцять полів `M1–M20`, з яких `M6` зарезервовано. Масив `MITIGATION_META` описує дев'ятнадцять активних параметрів, наведених у таблиці 4.5. У десктопному інтерфейсі користувач задає конфігурацію повзунками та отримує її оцінку; алгоритмічний пошук розглянуто окремо у методі синтезу.

Масив `MITIGATION_META` надає метайнформацію для кожного параметра мітигації: ключ, локалізовану назву українською мовою, мінімальне та максимальне значення, крок зміни та прапорець цілочисельності. Ця метайнформація використовується компонентом `Synthesis` для автоматичної генерації повзункових елементів керування.

Таблиця 4.5 – Параметри мітигаційного вектора `M` та їх значення за замовчуванням

Параметр	Опис	Діапазон	За замовчуванням
M1	Поріг вивантаження обчислень	[0, 1]	0.3
M2	Термічний ліміт (°C)	[35, 60]	45
M3	Чутливість до руху	[0, 10]	3
M4	К-ть потоків	{1, ..., 16}	4
M5	Архітектура дисплея (1=OLED, 2=μLED)	{1, 2}	1
M7	Режим відкату ПЗ	{1, 2}	1
M8	Інтервал пакетування	[0, 10]	2
M9	Частота контрольних точок	{1, ..., 10}	3

Закінчення таблиці 4.5

M10	Коефіцієнт вивантаження	[0, 0.9]	0.2
M11	К-ть мережевих каналів	{1, ..., 5}	2
M12	Фактор реплікації	{1, ..., 5}	2
M13	Буфер префетчування	{1, ..., 20}	5
M14	Розмір кешу	{1, ..., 100}	20
M15	Рівень модуляційного кодування	{1, ..., 5}	2
M16	Буфер інтерполяції	{1, ..., 20}	5
M17	Поріг адаптації інтерфейсу	[0, 3]	1
M18	Режим локомоції (1=Тп, 2=Мш, 3=Пл)	{1, 2, 3}	1
M19	Динамічний LOD	[0.5, 2.0]	1.0
M20	Комфортний тепловий поріг (°C)	[40, 60]	50

Функція `mapMitigationToParams` перетворює мітигаційний вектор і базові умови на похідні мережеві, середовищні, поведінкові, навантажувальні та апаратні параметри.

Мережеві перетворення враховують вивантаження обчислень, надлишкові канали, завадостійке кодування та буферизацію інтерполяції.

Навантаження змінюється під дією вивантаження, конкурентності й кешування; від нього залежать модельні температура обладнання та ресурс батареї.

Температура обладнання та час роботи батареї залежать від системного навантаження та технології дисплея. При виборі μ LED-архітектури ($M5 = 2$) температура знижується на 5.0 °C, а час роботи батареї збільшується на 10.0% відносно OLED-базової лінії.

Композитна інтенсивність відмов зменшується через налаштування чутливості руху, технологію дисплея та вивантаження обчислень. Усі 23 k-коефіцієнти визначено в об'єкті-константі та зведено у Таблиці 4.6.

Таблиця 4.6 – Повний набір калібрувальних коефіцієнтів параметричного перетворювача

Коефіцієнт	Значення	Опис
k_offload_bw	0.1	Покращення ПЗ на одиницю вивантаження
k_links_bw	0.2	Покращення ПЗ на кожний додатковий канал
k_links_lat	0.3	Зменшення затримки на кожний додатковий канал
k_links_loss	0.4	Зменшення втрат на кожний додатковий канал
k_coding	0.05	Корекція втрат на рівень кодування
k_interp	0.1	Зменш. затримки на одиницю буфера інтерполяції
k_fallback_light	50	Покращення освітленості у спрощеному режимі
k_motion_v	0.2	Зменшення розх. руху на одиницю чутливості
k_batching	0.1	Зменшення частоти на одиницю пакетування
k_prefetch	0.15	Зменшення частоти на одиницю буфера
k_teleport	0.4	Коефіцієнт навантаження для телепортації
k_scaled	0.7	Коефіцієнт навантаження для масштабованого руху
k_smooth	1.0	Коефіцієнт навантаження для плавного руху
k_concur	0.05	Зменшення навантаження на потік

Закінчення таблиці 4.6

k_cache_load	0.01	Зменшення навантаження на одиницю кешу
k_cache_data	0.1	Зменшення обсягу даних на одиницю кешу
k_load_temp	5.0	Підвищення температури на одиницю навантаження
k_display_temp	-5.0	Економія температури від μ LED-дисплея
k_load_bat	20.0	Витрата батареї на одиницю навантаження
k_display_bat	10.0	Економія батареї від μ LED-дисплея
k_sens_ λ	0.1	Зменш. інтенсивності відмов на одиницю чутливості
k_display_ λ	0.002	Зменшення інтенсивності відмов від μ LED
k_offload_ λ	0.3	Зменш. інтенсивності відмов на одиницю вивантаж.

Функція computeObjectives обчислює сім цільових показників резильєнтності. Надійність $R(t)$. Для порівняльної візуалізації сім показників нормалізуються до діапазону $[0, 1]$ (час відновлення інвертується, оскільки менше значення є кращим) та усереднюються для отримання агрегованого показника резильєнтності S_{agg}

4.4 Реалізація інтерфейсу користувача

Вкладка «Діаграма системи» надає інтерактивну SVG-візуалізацію матриці взаємодії факторів та компонентів розміром 5×4 . П'ять факторів впливу відображаються як кольорово-кодовані вузли у лівій частині діаграми, а чотири функціональні компоненти – як вузли у правій частині.

Двадцять кривих Безьє з'єднують кожну пару фактор-компонент. Криві динамічно забарвлюються та масштабуються за товщиною відповідно до значень $f_{i,j}$: зелений ($f_{i,j} \geq 0.8$) для номінальної роботи, помаранчевий ($0.5 \leq f_{i,j} < 0.8$) для помірної деградації та червоний ($f_{i,j} < 0.5$) для значної деградації. При наведенні курсору на криву відображається спливаюча підказка з числовим значенням $f(i,j)$.

Параметри, введені вручну, візуально відрізняються від значень телеметрії: значення від підключеного VR-пристрою позначаються синім кольором, ручні перевизначення – помаранчевим. Кожний параметр має кнопку скидання для повернення до значення за замовчуванням або значення телеметрії.

Під діаграмою відображається поточне значення інтегрального показника резильєнтності R_{VR} з кольоровою індикацією.

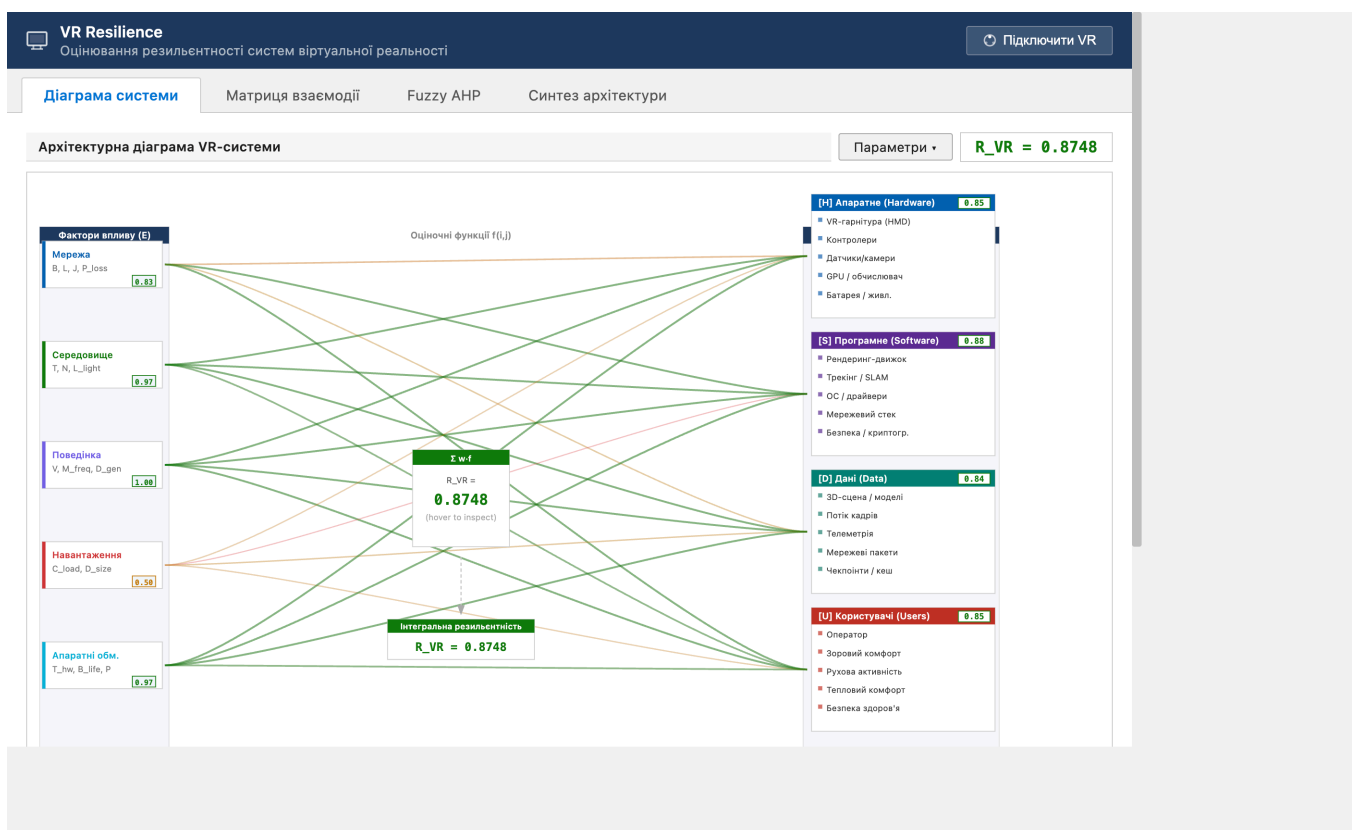


Рисунок 4.1 – Вкладка «Діаграма системи» програмного засобу

Вкладка «Матриця взаємодії» представляє табличну та теплокартну візуалізацію оцінювальної матриці 5×4 , що є кількісним доповненням до діаграми системи. Таблиця відображає числові значення усіх двадцяти $f_{i,j}$ з обчисленими середніми значеннями рядків та стовпців.

Теплова карта побудована на основі бібліотеки Plotly.js та використовує градієнтну кольорову шкалу від червоного (0) через помаранчевий (0.5) до зеленого (1.0). Кожна комірка теплової карти містить числове значення $f_{i,j}$, що дозволяє точно ідентифікувати найбільш деградовані взаємодії фактор-компонент.

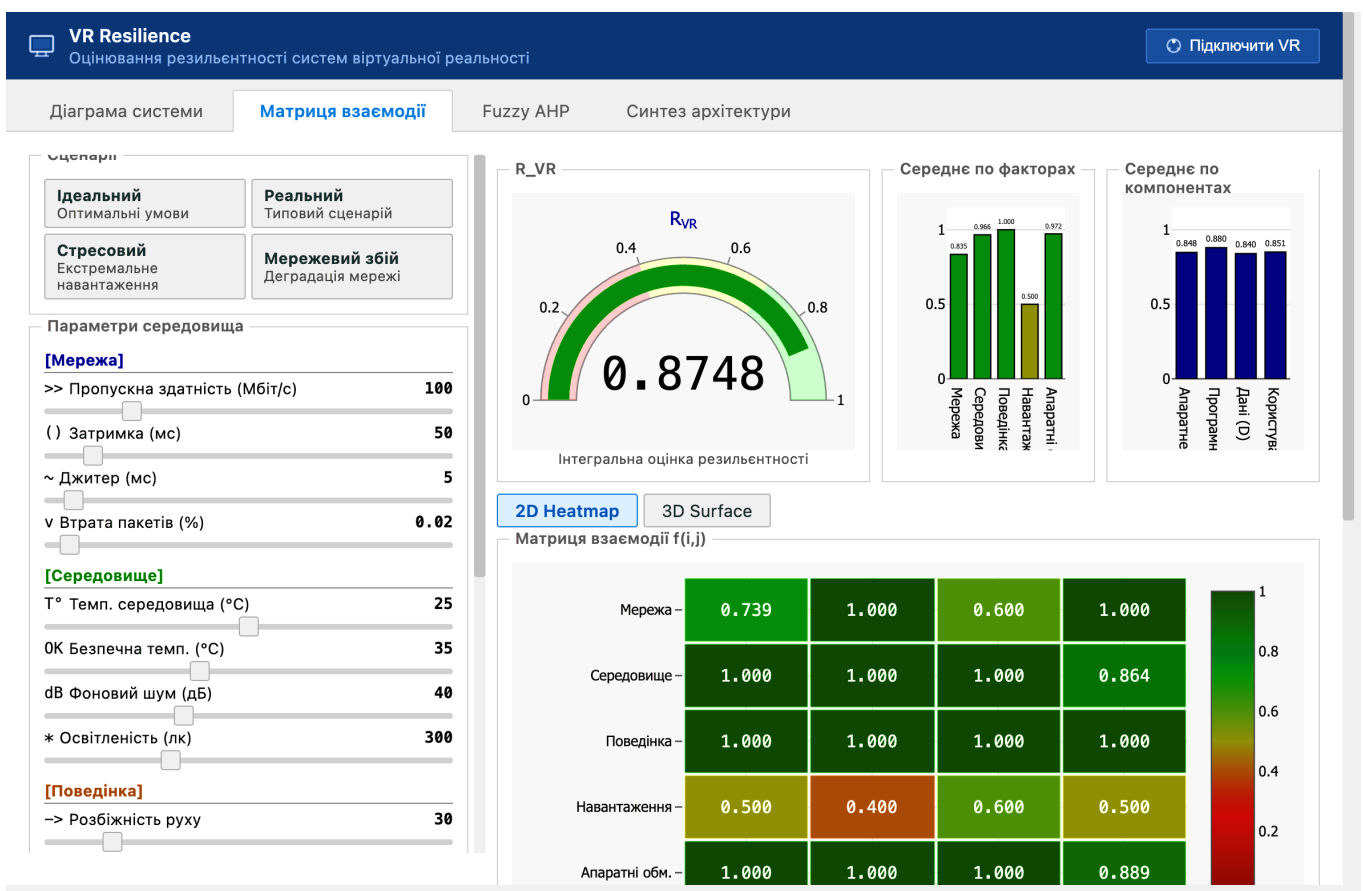


Рисунок 4.2 – Вкладка «Матриця взаємодії» з тепловою картою та діаграмами

Вкладка «Нечіткий аналіз ієрархій» показує матрицю парних порівнянь, нечіткі суми стовпців, нормалізовану матрицю, нечіткі вагові вектори та чіткі коефіцієнти після дефазифікації і нормування.

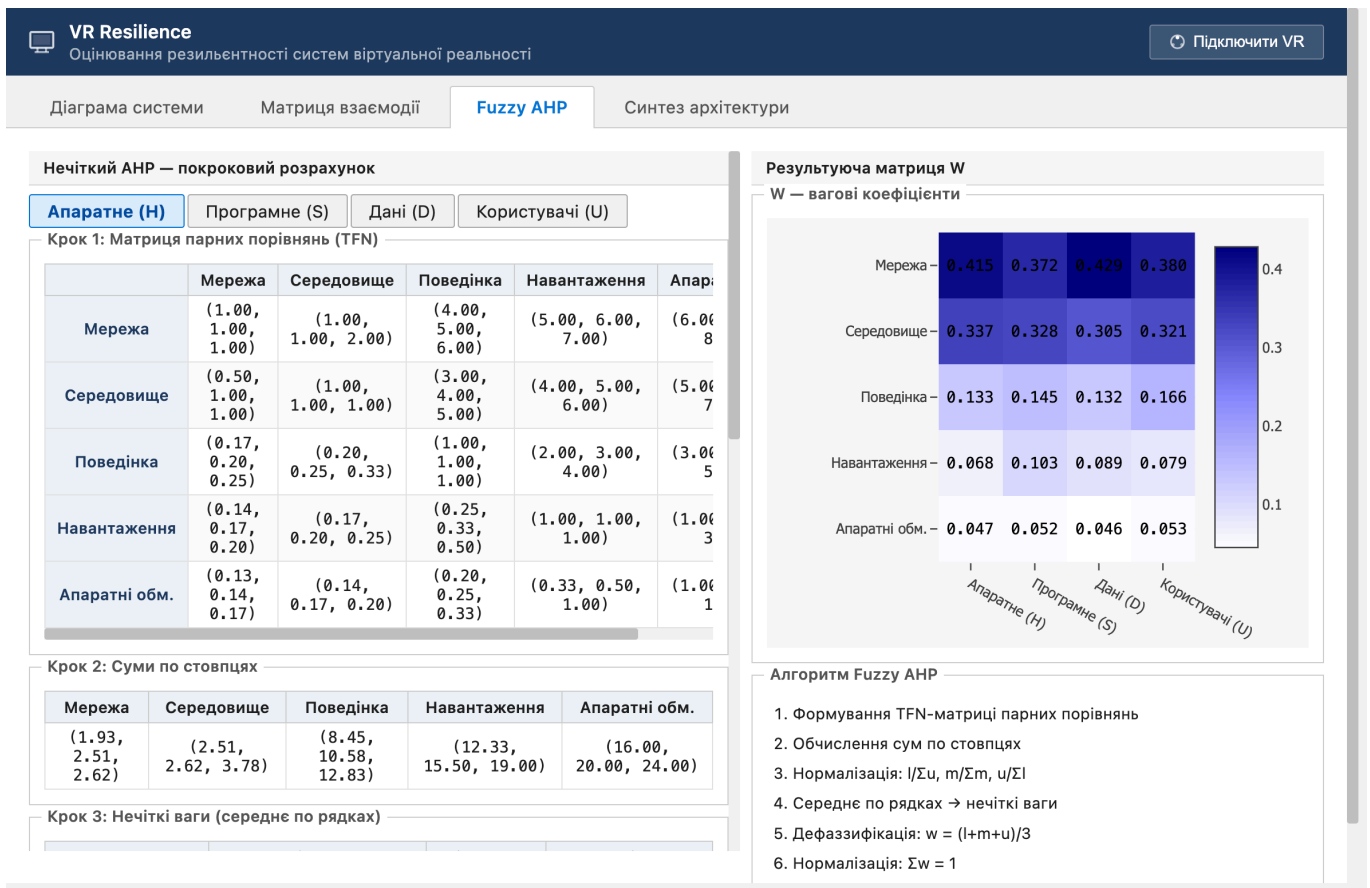


Рисунок 4.3 – Вкладка «Нечіткий аналіз ієрархій» з матрицею парних порівнянь та ваговими коефіцієнтами

Вкладка «Синтез архітектури» є основним інструментом оптимізації мітигаційних стратегій. Вона надає дев'ятнадцять повзункових елементів керування для параметрів мітигаційного вектора M з метаданими (діапазони, кроки, локалізовані назви) та додатковий повзунок для горизонту симуляції T (від 10 до 1000 одиниць часу).

У верхній частині вкладки відображається банер агрегованого показника S_{agg} .

Моніторингова сітка з дев'яти значень відображає похідні параметри симуляції: ефективну пропускну здатність, затримку, сприйману затримку, втрату пакетів, ефективний шум, системне навантаження, температуру обладнання, заряд батареї та інтенсивність відмов. Ці значення дозволяють відстежити інженерні механізми, що лежать в основі зміни цільових показників.

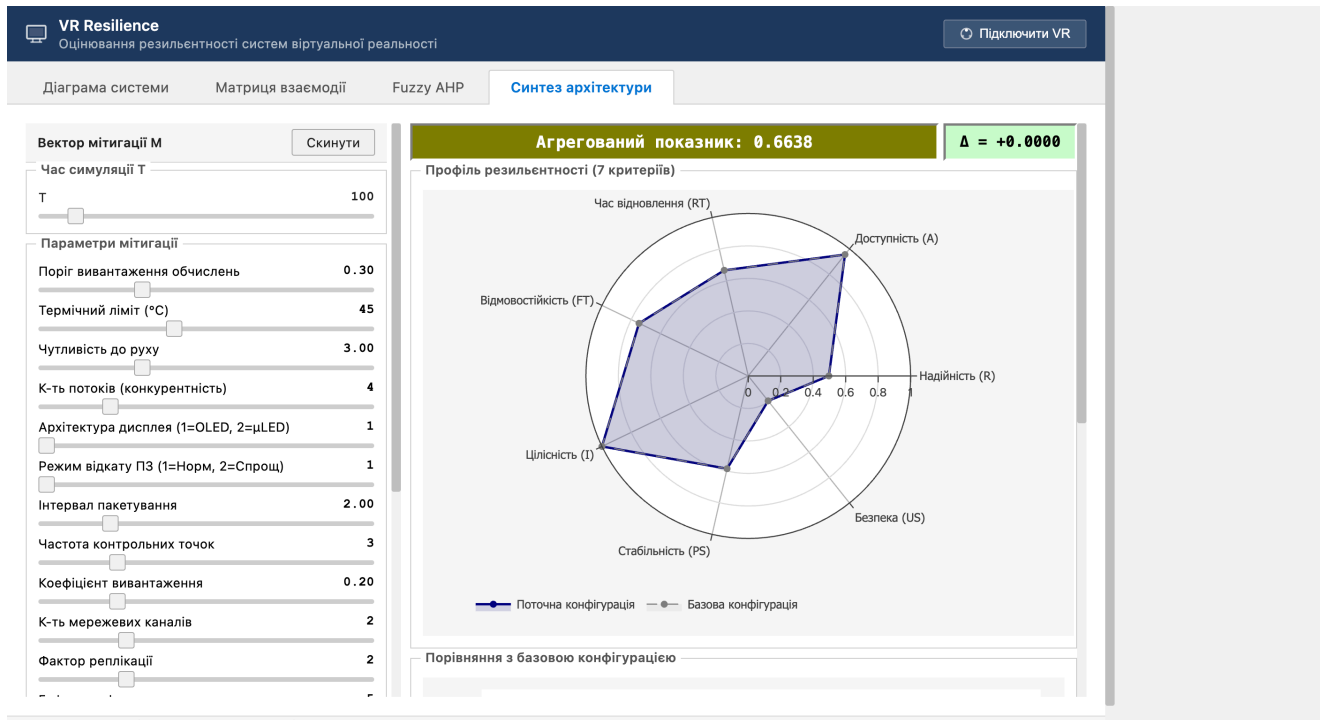


Рисунок 4.4 – Вкладка «Синтез архітектури» з пелюстковою діаграмою та параметрами мітигації

4.5 Симуляція підключення VR-пристроїв

Для підтримки реалістичних робочих процесів оцінювання без необхідності наявності фізичного VR-обладнання програмний засіб включає симулятор підключення VR-пристроїв. Симуляція проходить три стадії: сканування, вибір та підключення.

На стадії сканування застосунок виявляє віртуальні пристрої через анімацію сканування. У базі даних застосунку визначено три моделі VR-пристроїв з реалістичними характеристиками: Meta Quest 3, HTC Vive Pro 2 та Valve Index.

Для трьох моделей пристроїв задано різні базові мережеві, теплові, навантажувальні та енергетичні параметри. Їхні значення наведено у таблиці 4.7.

Базові параметри симульованих VR-пристроїв подано в таблиці 4.7.

Коли пристрій підключено, на вкладці «Синтез» відображається банер «Симуляційний режим», а усі обчислення цільових функцій використовують

параметри пристрою як базовий стан середовища, уможливлуючи сценарний аналіз в умовах, максимально наближених до реального розгортання.

Панель підключення VR-пристроїв із симульованою телеметрією представлено на рисунку 4.5.

Таблиця 4.7 – Базові параметри симульованих VR-пристроїв

Параметр	Meta Quest 3	HTC Vive Pro 2	Valve Index
B (Мбіт/с)	65	95	130
L (мс)	28	15	9
J (мс)	4	2	1
P_loss	0.018	0.005	0.001
B_life (%)	82	100	100
T_hw (°C)	38	44	51
C_load	0.55	0.62	0.70
V (ГБ)	2.1	3.5	4.2
T_env (°C)	23	22	24
N_noise (дБ)	38	42	45

4.6 Пакування та розповсюдження десктопного застосунку

Головний процес Electron створює вікно застосунку з початковим розміром 1400×860 пікселів і мінімальним розміром 900×600.

Безпека застосунку забезпечується через конфігурацію Electron: nodeIntegration вимкнено, contextIsolation увімкнено, що запобігає прямому доступу рендер-процесу до Node.js API. Файл preload.ts надає безпечний API для взаємодії між процесами.

Зовнішні посилання відкриваються у системному браузері через `shell.openExternal`, а не всередині Electron-вікна.

`electron-builder` формує дистрибутиви для macOS, Windows та Linux.

У режимі розробки Electron завантажує Vite-сервер, у продукційному режимі – статичні файли каталогу `dist/`.

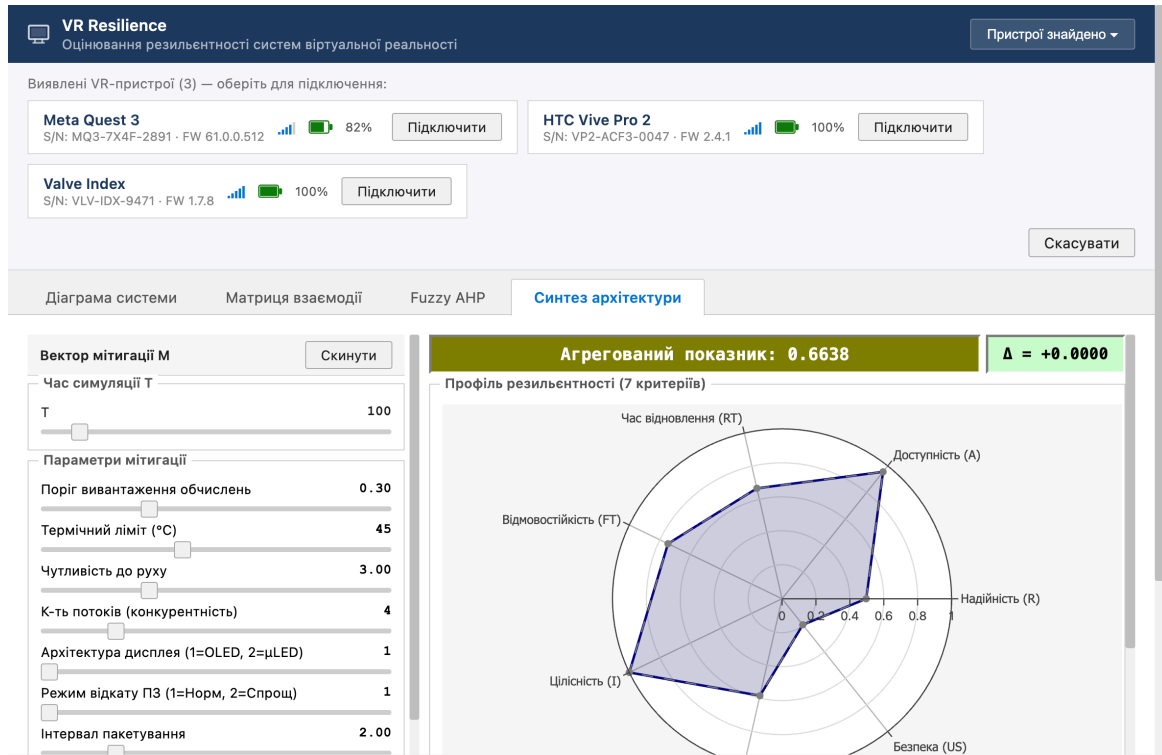


Рисунок 4.5 – Панель підключення VR-пристроїв із симульованою телеметрією

4.7 Верифікація програмної реалізації

Дослідження розроблених методів за операційними сценаріями виконано у підрозділі 3.4, тому цей підрозділ обмежено верифікацією самої програмної реалізації. Верифікація має відповісти на два питання: чи відтворює обчислювальне ядро результати еталонної моделі за однакових вхідних даних і чи узгоджуються обчислені засобом вагові коефіцієнти зі структурними властивостями предметної області. Ці питання стосуються реалізації, а не методів, і тому розглядаються окремо від результатів розділу 3.

У ваговій матриці таблиці 4.4 фактор стану мережі має найбільшу сумарну вагу 0,403. Найбільша індивідуальна вага належить парі «мережа – апаратне забезпечення» і становить 0,1078; вага пари «мережа – дані» дорівнює 0,1072. Таке співвідношення є наслідком заданих експертних матриць парних порівнянь.

Ваги середовища для чотирьох компонентів перебувають у межах 0,076–0,082. Для поведінки користувача найбільшою є вага впливу на користувацький компонент, $w_{3,u} = 0,0415$. Апаратні обмеження мають найменшу сумарну вагу 0,049. Ваги характеризують задану експертну важливість впливів і самі по собі не є оцінками частоти відмов. Числовий приклад наведено у підрозділі 2.2.2, а перевірку узгодженості всіх чотирьох матриць — у додатку В.

Порівняння результатів двох реалізацій вимагає обережності, оскільки наведені у підрозділах 3.4.2–3.4.4 значення та результати Simulink-моделювання з розділів 2 і 3 отримані за різних параметризацій. У базовому сценарії Simulink-моделювання використано затримку 60 мс і втрати пакетів 2%, ширші діапазони впливів та інші значення за замовчуванням, тоді як базова конфігурація десктопного засобу відповідає таблицям 3.3 і 4.5; для довідки, Simulink-експеримент дав надійність 0,62 базовій конфігурації проти 0,94 оптимізованої, доступність 0,45 проти 0,93 та час відновлення 7,10 с проти 0,48 с. Пряме зіставлення абсолютних значень різних експериментів, наприклад доступності 0,955 у реалізації TypeScript і 0,45 у Simulink-експерименті, є некоректним: це результати обчислень над різними вхідними даними, а не розбіжність реалізацій одного розрахунку.

Для коректної перевірки збіжності двох програмних реалізацій виконано контрольний розрахунок за уніфікованих вхідних даних. З цією метою в обидві реалізації було подано ідентичний вектор параметрів середовища, що відповідає базовій конфігурації з таблиці 3.9, однакову вагову матрицю $\backslash(W\backslash)$, а також однаковий мітигаційний вектор. Такий підхід дає змогу виключити вплив відмінностей у початкових умовах і безпосередньо оцінити ступінь узгодженості обчислювальних процедур. Результати контрольного розрахунку наведено у таблиці 4.8. Отримані розбіжності між значеннями показників не перевищують 1,2 %, що свідчить про достатньо високу збіжність реалізацій у межах прийнятої постановки задачі.

Виявлена різниця зумовлена насамперед особливостями чисельного опрацювання моделі. У MATLAB/Simulink модель інтегрується в часі із заданим фіксованим кроком, а обчислювальне ядро, реалізоване мовою TypeScript, виконує розрахунок показників безпосередньо за замкненими аналітичними виразами (3.13)–(3.21). Через це незначна різниця у підсумкових значеннях є очікуваною і не свідчить про розбіжність математичної постановки або некоректність однієї з реалізацій.

Розмежування ролей реалізацій при цьому є принциповим. Модель MATLAB/Simulink використовується як концептуальний еталон запропонованого методу та як середовище динамічної верифікації моделей і залежностей, сформульованих у розділах 2 і 3. Її призначення полягає передусім у дослідженні поведінки системи в часі.

Обчислювальне ядро TypeScript, своєю чергою, є еталонною інженерною реалізацією програмного засобу. Воно орієнтоване на практичне використання запропонованих моделей у складі прикладного програмного комплексу, швидке виконання розрахунків і інтерактивне опрацювання параметрів. Таким чином, Simulink і TypeScript виконують різні, але взаємодоповнювальні функції: перша реалізація використовується переважно для верифікації та аналізу динаміки, друга — для інженерного застосування та інтерактивного дослідження.

Відмінності абсолютних значень показників, наведених у різних підрозділах роботи, не є наслідком невідповідності між реалізаціями. Вони пояснюються виключно використанням різних вхідних наборів даних, різними сценаріями впливів, параметрами середовища та конфігураціями мітигаційних дій у відповідних експериментах. Проведений контрольний розрахунок з однаковими вхідними параметрами підтверджує це положення і дозволяє відокремити відмінності, пов'язані з постановкою окремих експериментів, від похибок самої програмної реалізації.

Крім збіжності абсолютних значень, в обох реалізаціях узгоджується і якісна поведінка досліджуваних показників. Зберігається однакове ранжування сценаріїв за ступенем тяжкості, а також збігаються напрямки зміни результатів після застосування мітигаційних заходів. Конфігурації, що демонструють покращення в

MATLAB/Simulink, аналогічно покращують відповідні критерії і в інженерній реалізації TypeScript. Це є додатковим підтвердженням того, що між двома реалізаціями зберігається не лише чисельна, а й структурна відповідність.

Окремо слід розмежовувати результати, отримані для різних цільових показників. Порівняння двох фіксованих мітигаційних конфігурацій у десктопному програмному засобі демонструє приріст агрегованого показника S_{agg} на 31,5 %. Водночас це значення не є оцінкою зміни показника R_{VR} у Simulink-експерименті, оскільки відповідні величини формуються в різних експериментальних постановках і не повинні інтерпретуватися як безпосередньо взаємозамінні. Значення 31,5 % характеризує ефект переходу між двома конкретними конфігураціями в межах обчислювального ядра TypeScript, тоді як результати Simulink використовуються для оцінювання поведінки моделі за окремими сценаріями динамічного впливу.

Додатковою перевагою реалізації TypeScript є можливість інтерактивної візуалізації та дослідження параметричного простору в реальному часі. Користувач може оперативно змінювати параметри середовища, вагові коефіцієнти та мітигаційні дії й без повторного налаштування моделі отримувати оновлені значення критеріїв та агрегованої оцінки. Це спрощує аналіз альтернативних конфігурацій, порівняння сценаріїв і виявлення параметрів, до яких результат є найбільш чутливим. У сукупності контрольний розрахунок, збіжність якісної поведінки показників та узгоджене відтворення результатів підтверджують відповідність інженерної реалізації математичним і модельним положенням, сформульованим у попередніх розділах роботи.

Виконана верифікація програмного засобу на цьому етапі обмежується перевіркою його обчислювальної частини та коректності реалізації закладених математичних залежностей. Основну увагу зосереджено на правильності розрахунку показників, узгодженості результатів між окремими реалізаціями, відтворюваності обчислень за однакових вхідних даних, а також на перевірці реакції системи на зміну параметрів модельованого середовища. Таким чином, отримані результати підтверджують працездатність внутрішньої логіки програмного засобу, але не охоплюють повний цикл його взаємодії з реальними фізичними пристроями.

Таблиця 4.8 – Контрольне порівняння реалізацій за уніфікованих вхідних даних

Показник	TypeScript	Simulink	Відхилення, %
Надійність R	0,497	0,494	0,6
Доступність A	0,955	0,951	0,4
Час відновлення RT, с	6,667	6,720	0,8
Відмовостійкість FT	0,747	0,744	0,4
Стабільність продуктивності PS	0,585	0,578	1,2
Безпека користувача US	0,196	0,198	1,0

Складові програмного засобу, функціонування яких безпосередньо залежить від зовнішнього оточення, перевірялися окремо. До них, зокрема, належать драйвери підключення фізичних пристроїв, механізми приймання телеметричних даних, оброблення вхідних потоків показників у реальному часі та передавання цих даних до обчислювального ядра. На поточному етапі такі компоненти перевірено лише з використанням симулятора пристроїв, описаного у підрозділі 4.5. Симулятор забезпечує формування контрольованих наборів телеметричних даних і дозволяє перевірити коректність їх приймання та подальшого використання без необхідності підключення реального обладнання.

Такий підхід є достатнім для функціональної перевірки програмних інтерфейсів і основних сценаріїв обміну даними, проте не дає змоги повною мірою оцінити вплив апаратно залежних факторів. У реальних умовах на роботу засобу можуть додатково впливати затримки передавання даних, нестабільність з'єднання, відмінності у форматах телеметрії, похибки сенсорів, особливості драйверів конкретних пристроїв, частота оновлення показників та інші фактори, які симулятор відтворює лише у спрощеному або контрольованому вигляді. Тому результати проведеної верифікації не слід безпосередньо інтерпретувати як оцінку роботи засобу з певною моделлю серійного обладнання.

Одержані показники характеризують насамперед коректність обчислювальних процедур, стабільність програмної реалізації та відтворюваність результатів за визначених умов експерименту. Вони підтверджують, що за однакових вхідних даних система формує очікувані значення показників і зберігає передбачену логіку реакції на зміну параметрів. Водночас ці результати не містять оцінки апаратної сумісності, реальних затримок у каналах телеметрії, надійності роботи драйверів або поведінки системи за умов фізичних збоїв обладнання.

З огляду на це перевірка програмного засобу із застосуванням реального серійного обладнання визначена як окремий напрям подальших робіт. Така перевірка має включати підключення фізичних пристроїв, приймання реальної телеметрії, оцінювання затримок і втрат даних, перевірку стійкості програмного засобу до нестабільного зв'язку та зіставлення результатів із даними, отриманими у симуляційному середовищі. Це дозволить доповнити поточну обчислювальну верифікацію експериментальною перевіркою в умовах, наближених до реальної експлуатації

4.8 Висновки до розділу 4

У розділі реалізовано програмний засіб із ядром TypeScript, інтерфейсом React і пакуванням Electron. MATLAB/Simulink використано для прототипування та динамічної перевірки.

Для двох мітигаційних конфігурацій агрегована оцінка зростає від 0,664 до 0,873, або на 31,5 %. Окрема оцінка стану сценарію Stress становить 0,184; ці розрахунки не утворюють спільної траєкторії відновлення.

Контрольне порівняння з Simulink за однакових вхідних даних показало розбіжність до 1,2 % за шістьма показниками, що пов'язано з відмінностями дискретизації.

Основні результати розділу опубліковані у працях [82, 84, 85].

ВИСНОВКИ

У дисертаційній роботі вирішено науково-прикладну задачу забезпечення резильєнтності систем віртуальної та доповненої реальності. Розроблено модель резильєнтного функціонування, метод оцінювання стану, метод синтезу мітигаційних стратегій і програмний засіб для їх реалізації.

Відповідно до мети та завдань дослідження отримано такі результати:

1. Проведено аналіз архітектур XR-систем та методів оцінювання їх резильєнтності. Виділено п'ять груп факторів впливу і чотири компонентні групи. Встановлено потребу в інтегрованій оцінці поточного стану, засобах локалізації деградації та механізмах відновлення.

2. Розроблено модель резильєнтного функціонування XR-системи. Модель містить матрицю з двадцяти безрозмірних функцій, що описують вплив мережі, середовища, поведінки користувача, навантаження й апаратних обмежень на апаратне забезпечення, програмне забезпечення, дані та користувацький контур.

3. Розроблено метод оцінювання стану XR-системи. Інтегральний показник визначається зваженою сумою елементів матриці оцінок. Ваги отримуються методом нечіткого аналізу ієрархій. Наведено числовий приклад розрахунку та перевірку узгодженості.

4. Описано цикл забезпечення резильєнтності, який охоплює збирання телеметрії, оцінювання стану, локалізацію критичних взаємодій, синтез мітигаційної стратегії, її застосування та повторне оцінювання. Відновлення визначено як обов'язкову складову резильєнтності.

5. Розроблено метод мультикритеріального синтезу мітигаційних стратегій. Критерії надійності, доступності, відмовостійкості, цілісності, стабільності продуктивності, безпеки користувача та часу відновлення зводяться до спільної безрозмірної шкали. Оптимізація виконується за встановлених ресурсних і безпекових обмежень.

6. Реалізовано кросплатформний десктопний програмний засіб з обчислювальним ядром TypeScript у середовищі Node.js та Electron.

MATLAB/Simulink застосовано для прототипування і динамічної перевірки. Контрольний розрахунок за однакових входних даних показав розбіжність реалізацій не більше 1,2%.

7. Апробацію виконано на модельних сценаріях деградації мережі, термічного навантаження, сенсорних завад, зміни освітлення, пікового навантаження та поведінкових відхилень. Для чотирьох детермінованих сценаріїв показник стану R_{VR} набуває значень від 0,184 до 0,944. В окремому порівнянні мітигаційних конфігурацій агрегована оцінка S_{agg} зростає від 0,664 до 0,873. Ці результати стосуються заданої синтетичної параметризації; підтвердження повного відновлення потребує перевірки критичних функцій і часових обмежень профілю місії.

Обмеження дослідження пов'язані з використанням синтетичної телеметрії, заданих порогів і модельних профілів впливів. Кількісні результати потребують натурної перевірки. Подальше дослідження має охопити фізичний VR-стенд, відкриті набори телеметрії, калібрування функцій деградації та перевірку безпеки користувача за різних сфер застосування.

Перспективи подальших досліджень пов'язані з апробацією методів на фізичному VR-стенді, збиранням реальної телеметрії, використанням відкритих наборів вимірювань та уточненням функцій деградації за результатами натурних випробувань.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Malware Detection by Distributed Systems with Partial Centralization / B. Savenko, A. Kashtalian, S. Lysenko, O. Savenko. 2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) : proceedings (Dortmund, Germany, 2023). P. 265–270. DOI: 10.1109/IDAACS58523.2023.10348773.
2. The technique for metamorphic viruses' detection based on its obfuscation features analysis / G. Markowsky, O. Savenko, S. Lysenko, A. Nicheporuk. CEUR Workshop Proceedings. 2018. Vol. 2104. P. 680–687. ISSN 1613-0073.
3. Technique for IoT malware detection based on control flow graph analysis / K. Bobrovnikova, S. Lysenko, B. Savenko, P. Gaj, O. Savenko. Radioelectronic and Computer Systems. 2022. No. 1. P. 141–153.
4. Lysenko S., Savenko O., Bobrovnikova K. DDoS Botnet Detection Technique Based on the Use of the Semi-Supervised Fuzzy c-Means Clustering. CEUR Workshop Proceedings. 2018. Vol. 2104. P. 688–695.
5. A Technique for detection of bots which are using polymorphic code / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, A. Nicheporuk. Communications in Computer and Information Science. 2014. Vol. 431. P. 265–276. ISSN 1865-0929.
6. Pyun K., Rogers J., Ko S. Materials and devices for immersive virtual reality. Nature Reviews Materials. 2022. Vol. 7. DOI: 10.1038/s41578-022-00501-5.
7. Linkov I., Trump B. D. The Science and Practice of Resilience. Cham : Springer, 2019. 220 p.
8. Cyber Resilience of Systems and Networks / ed. by A. Kott, I. Linkov. Cham : Springer, 2019. 471 p.
9. Developing Cyber-Resilient Systems: A Systems Security Engineering Approach : NIST Special Publication 800-160, Vol. 2, Rev. 1 / R. Ross, V. Pillitteri, R. Graubart, D. Bodeau, R. McQuaid. Gaithersburg, MD : National Institute of Standards and Technology, 2021. DOI: 10.6028/NIST.SP.800-160v2r1.

10. Poulin C., Kane M. B. Infrastructure resilience curves: Performance measures and summary metrics. *Reliability Engineering and System Safety*. 2021. Vol. 216. Article 107926.
11. Toward Low-Latency and Ultra-Reliable Virtual Reality / M. S. Elbamby, C. Perfecto, M. Bennis, K. Doppler. *IEEE Network*. 2018. Vol. 32, no. 2. P. 78–84. DOI: 10.1109/MNET.2018.1700268.
12. MazeRunVR: An Open Benchmark for VR Locomotion Performance, Preference and Sickness in the Wild / K. Marky, K. Ragozin, K. Kunze, Y. Pai. *CHI 2020 : Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA, April 25–30, 2020). New York : ACM, 2020. DOI: 10.1145/3334480.3383035.
13. Kumari P., Kaur P. A survey of fault tolerance in cloud computing. *Journal of King Saud University – Computer and Information Sciences*. 2021. Vol. 33, no. 10. P. 1159–1176.
14. Burns B. *Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services*. Sebastopol, CA : O'Reilly Media, 2018. 164 p.
15. Model-Based Analysis of Microservice Resiliency Patterns / N. C. Mendonça, C. M. Aderaldo, J. Cámara, D. Garlan. 2020 *IEEE International Conference on Software Architecture (ICSA) : proceedings* (Salvador, Brazil, 2020). P. 114–124.
16. Weyns D. *An Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective*. Hoboken, NJ : Wiley – IEEE Press, 2020. 240 p.
17. Wong T., Wagner M., Treude C. Self-adaptive systems: A systematic literature review across categories and domains. *Information and Software Technology*. 2022. Vol. 148. Article 106934.
18. Zink M., Sitaraman R., Nahrstedt K. Scalable 360° Video Stream Delivery: Challenges, Solutions, and Opportunities. *Proceedings of the IEEE*. 2019. Vol. 107, no. 4. P. 639–650.
19. Cellular-Connected Wireless Virtual Reality: Requirements, Challenges, and Solutions / F. Hu, Y. Deng, W. Saad, M. Bennis, A. H. Aghvami. *IEEE Communications Magazine*. 2020. Vol. 58, no. 5. P. 105–111.

20. Forsgren N., Humble J., Kim G. Accelerate: The Science of Lean Software and DevOps. Building and Scaling High Performing Technology Organizations. Portland, OR : IT Revolution Press, 2018. 288 p.
21. The Site Reliability Workbook: Practical Ways to Implement SRE / ed. by B. Beyer, N. R. Murphy, D. K. Rensin, K. Kawahara, S. Thorne. Sebastopol, CA : O'Reilly Media, 2018. 512 p.
22. Nygard M. T. Release It! Design and Deploy Production-Ready Software. 2nd ed. Raleigh, NC : Pragmatic Bookshelf, 2018. 376 p.
23. Building Secure and Reliable Systems: Best Practices for Designing, Implementing, and Maintaining Systems / H. Adkins, B. Beyer, P. Blankinship, P. Lewandowski, A. Oprea, A. Stubblefield. Sebastopol, CA : O'Reilly Media, 2020. 558 p.
24. Balasubramanian P. Analysis of Redundancy Techniques for Electronics Design – Case Study of Digital Image Processing. Technologies. 2023. Vol. 11, no. 3. Article 80. DOI: 10.3390/technologies11030080.
25. Sakthivel S., Sathvik Reddy O. Fault Tolerant Digital Filters on FPGA Using Hardware Redundancy Techniques and Denoising ECG Signals. 2017 International Conference of Electronics, Communication and Aerospace Technology (ICECA) : proceedings. 2017. DOI: 10.1109/ICECA.2017.8212811.
26. Rosenthal C., Jones N. Chaos Engineering: System Resiliency in Practice. Sebastopol, CA : O'Reilly Media, 2020. 304 p.
27. Automating Chaos Experiments in Production / A. Basiri, L. Hochstein, N. Jones, H. Tucker. 41st IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) : proceedings (Montreal, QC, Canada, 2019). P. 31–40.
28. A Chaos Engineering System for Live Analysis and Falsification of Exception-Handling in the JVM / L. Zhang, B. Morin, P. Haller, B. Baudry, M. Monperrus. IEEE Transactions on Software Engineering. 2021. Vol. 47, no. 11. P. 2534–2548.
29. Pavlova O., Bashta A., Kovtoniuk M. Augmented Reality Based Information Technology for Objects 3D Models Visualization. Computer Systems and Information Technologies. 2023. Vol. 1. P. 62–68. DOI: 10.31891/csit-2023-1-9.

30. Methods of improving security and resilience of VR systems' architecture / A. Kachur, S. Lysenko, O. Bodnaruk, P. Gaj. 5th International Workshop on Intelligent Information Technologies and Systems of Information Security (IntelITSIS'2024) : proceedings (Khmelnyskyi, Ukraine, March 28, 2024). P. 285–304.
31. Botnet detection technique for corporate area network / O. Savenko, S. Lysenko, A. Kryshchuk, Y. Klots. 7th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications : proceedings (Berlin, Germany, September 12–14, 2013). P. 363–368. ISBN 978-1-4799-1426-5.
32. A Technique for the Botnet Detection Based on DNS-Traffic Analysis / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, K. Bobrovnikova. Communications in Computer and Information Science. 2015. Vol. 522. P. 127–138. ISSN 1865-0929.
33. Lysenko S., Bobrovnikova K., Savenko O. A Botnet Detection Approach Based on The Clonal Selection Algorithm. 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DeSSerT-2018) : proceedings (Kyiv, Ukraine, May 24–27, 2018). P. 424–428.
34. Information technology for botnets detection based on their behaviour in the corporate area network / S. Lysenko, O. Savenko, K. Bobrovnikova, A. Kryshchuk. Communications in Computer and Information Science. 2017. Vol. 718. P. 166–181. ISSN 1865-0929.
35. Savenko O., Lysenko S., Kryschuk A. Multi-agent based approach of botnet detection in computer systems. Communications in Computer and Information Science. 2012. Vol. 291. P. 171–180. ISSN 1865-0929.
36. Anti-evasion Technique for the Botnets Detection Based on the Passive DNS Monitoring and Active DNS Probing / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, K. Bobrovnikova. Communications in Computer and Information Science. 2016. Vol. 608. P. 83–95. ISSN 1865-0929.
37. Method for Detecting Steganographic Changes in Images Using Machine Learning / D. Denysiuk, O. Savenko, S. Lysenko, B. Savenko, A. Kashtalian. 2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT) : proceedings (Athens, Greece, 2023). P. 1–6.

38. JOSS: Joint Exploration of CPU-Memory DVFS and Task Scheduling for Energy Efficiency / J. Chen, M. Manivannan, B. Goel, M. Pericàs. arXiv preprint. 2023. arXiv:2303.04615. P. 1–11.
39. DVFO: Learning-Based DVFS for Energy-Efficient Edge-Cloud Collaborative Inference / Z. Zhang, Y. Zhao, H. Li, C. Lin, J. Liu. arXiv preprint. 2023. arXiv:2306.01811. P. 1–16.
40. An Efficient FPGA Accelerator for Point Cloud / Z. Wang, W. Mao, P. Yang, Z. Wang, J. Lin. arXiv preprint. 2023. arXiv:2303.07803. P. 1–6.
41. An Investigation into the Comprehensive Impact of Self-Heating and Hot Carrier Injection / Y. Liu, Y. Ma, Z. Yu, S. Lou, Y. Qu, Y. Chang. Electronics. 2022. Vol. 11, no. 17. P. 1–11.
42. Hot Carrier Injection Reliability in Nanoscale Field Effect Transistors: Modeling and Simulation Methods / Y. Wang, Y. Li, Y. Yang, W. Chen. Electronics. 2022. Vol. 11. P. 1–16.
43. Mandal D. K., Jandhyala S. Visual Inertial Odometry At the Edge: A Hardware-Software Co-design Approach for Ultra-low Latency and Power. 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE) : proceedings (Florence, Italy, 2019). P. 960–963.
44. Hazelwood D. A., Pecht M. G. Life Extension of Electronic Products: A Case Study of Smartphones. IEEE Access. 2021. Vol. 9. P. 144726–144739.
45. Hardware-in-the-Loop-on-Chip Development System for Teaching and Development of Dynamic Systems / W. Jiang, L. Sun, Y. Chen, H. Ma, S. Hashimoto. Electronics. 2021. Vol. 10. P. 1–15.
46. Dinu A., Ogrutan P. L. Reinforcement Learning Made Affordable for Hardware Verification Engineers. Micromachines. 2022. Vol. 13. P. 1–28.
47. Reduction of Losses and Operating Costs in Distribution Networks Using a Genetic Algorithm and Mathematical Optimization / F. E. Riaño, J. F. Cruz, O. D. Montoya, H. R. Chamorro, L. Alvarado-Barrios. Electronics. 2021. Vol. 10. P. 1–25.

48. CPU-GPU-Memory DVFS for Power-Efficient MPSoC in Mobile Cyber Physical System / S. Dey, S. Isuwa, S. Saha, A. K. Singh, K. McDonald-Maier. *Future Internet*. 2022. Vol. 14. P. 1–14.
49. Nabavinejad S. M., Reda S., Ebrahimi M. Coordinated Batching and DVFS for DNN Inference on GPU Accelerators. *IEEE Transactions on Parallel and Distributed Systems*. 2022. Vol. 33, no. 10. P. 2496–2508.
50. Deep Learning for 3D Point Clouds: A Survey / Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, M. Bennamoun. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2020. Vol. 43, no. 12. P. 4338–4364.
51. Liao M. H., Hsieh C. P., Lee C. C. Systematic investigation of self-heating effect on CMOS logic transistors from 20 to 5 nm technology nodes by experimental thermoelectric measurements and finite element modeling. *IEEE Transactions on Electron Devices*. 2017. Vol. 64. P. 646–648.
52. Mahapatra S., Sharma U. A Review of Hot Carrier Degradation in N-Channel MOSFETs – Part I: Physical Mechanism. *IEEE Transactions on Electron Devices*. 2022. Vol. 67. P. 2660–2671.
53. Integration Verification across Software and Hardware for a Simple Embedded System / A. Erbsen, S. Gruetter, J. Choi, C. Wood, A. Chlipala. *PLDI 2021 : Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*. New York : ACM, 2021. P. 604–619.
54. Özçelik A., Löchtfeld M., Tollestrup C. Long-lasting smart products: Overview of longevity concepts in sustainable ICT and design for sustainability. *DRS2022 : proceedings (Bilbao, Spain, June 25 – July 3, 2022)*. 2022. Vol. 1. P. 1–18.
55. Haririan P. DVFS and Its Architectural Simulation Models for Improving Energy Efficiency of Complex Embedded Systems in Early Design Phase. *Computers*. 2020. Vol. 9. P. 1–21.
56. Khriji S., Chéour R., Kanoun O. Dynamic Voltage and Frequency Scaling and Duty-Cycling for Ultra Low-Power Wireless Sensor Nodes. *Electronics*. 2022. Vol. 11. P. 1–19.

57. The Effects of Data Security and Perceived Benefits on Mobile Phone Recycling Behaviour and the Recycling Intention Mediation Role / T. Ben Yahya, N. M. Jamal, B. Sundarakani, S. Z. Omain. *Recycling*. 2023. Vol. 8. P. 1–26.
58. Montoya O. D., Gil-González W., Orozco-Henao C. Vortex search and Chu-Beasley genetic algorithms for optimal location and sizing of distributed generators in distribution networks: A novel hybrid approach. *Engineering Science and Technology, an International Journal*. 2020. Vol. 23. P. 1351–1363.
59. Deep Reinforcement Learning for Dynamic Flexible Job Shop Scheduling with Random Job Arrival / J. Chang, D. Yu, Y. Hu, W. He, H. Yu. *Processes*. 2022. Vol. 10. P. 1–20.
60. Distributed On-Sensor Compute System for AR/VR Devices: A Semi-Analytical Simulation Framework for Power Estimation / J. Gomez, S. Patel, S. Sarwar, Z. Li, R. Capoccia, Z. Wang, R. Pinkham, A. Berkovich, T. Tsai, B. Salvo, C. Liu. *arXiv preprint*. 2022. arXiv:2203.07474. P. 1–6.
61. Sumbul H. E. System-Level Design and Integration of a Prototype AR/VR Hardware Featuring a Custom Low-Power DNN Accelerator Chip in 7 nm Technology for Codec Avatars. 2022 IEEE Custom Integrated Circuits Conference (CICC) : proceedings (Newport Beach, CA, USA, 2022). P. 1–8.
62. A Proposed Hardware-Software Architecture for Virtual Reality in Industrial Applications / F. Chionna, P. Cirillo, V. Palmieri, M. Bellone. *CIRP Annals – Manufacturing Technology*. 2015. Vol. 56. P. 139–142.
63. Energy-Efficient Video Processing for Virtual Reality / Y. Leng, J. Huang, C. Chen, Q. Sun, Y. Zhu. *IEEE Micro*. 2020. Vol. 40. P. 30–36.
64. Real-Time Energy Efficient Hand Pose Estimation: A Case Study / M. R. Al Koutayni, V. Rybalkin, A. Elhayek, C. Weis, G. Reis, N. Wehn, D. Stricker. *Sensors*. 2020. Vol. 20. P. 1–25.
65. Wu L. Research on the Development and Application of Parallel Programming Technology in Heterogeneous Systems. *Journal of Physics: Conference Series*. 2022. Vol. 2173. P. 1–8.

66. Chuang S. M., Chen C. S., Wu E. Hsiao-Kuang. The Implementation of Interactive VR Application and Caching Strategy Design on Mobile Edge Computing (MEC). *Electronics*. 2023. Vol. 12. P. 1–13.
67. Research on Optimization Method of VR Task Scenario Resources Driven by User Cognitive Needs / Q. Fu, J. Lv, Z. Zhao, D. Yue. *Information*. 2020. Vol. 11. P. 1–22.
68. Resource allocation for virtual reality content sharing based on 5G D2D multicast communication / Y. Yang, L. Feng, C. Zhang, Q. Ou, W. Li. *EURASIP Journal on Wireless Communications and Networking*. 2020. No. 1. P. 1–12.
69. de Freitas F. V., Gomes M. V. M., Winkler I. Benefits and Challenges of Virtual-Reality-Based Industrial Usability Testing and Design Reviews: A Patents Landscape and Literature Review. *Applied Sciences*. 2022. Vol. 12. P. 1–27.
70. Ruan J., Xie D. Networked VR: State of the Art, Solutions, and Challenges. *Electronics*. 2021. Vol. 10. P. 1–18.
71. Challenges in Virtual Reality Training for CBRN Events / G. Regal, D. Pretolesi, H. Schrom-Feiertag, J. Puthenkalam, M. Migliorini, E. De Maio, F. Scarrone, M. Nadalin, M. Guarneri, G. P. Xerri. *Multimodal Technologies and Interaction*. 2023. Vol. 7. P. 1–18.
72. Learning-Directed Dynamic Voltage and Frequency Scaling Scheme with Adjustable Performance for Single-Core and Multi-Core Embedded and Mobile Systems / Y.-L. Chen, M.-F. Chang, C.-W. Yu, X.-Z. Chen, W.-Y. Liang. *Sensors*. 2018. Vol. 18. P. 1–11.
73. Efficiency Improvement in Multiphase Converter by Changing Dynamically the Number of Phases / P. Zumel, C. Fernández, A. De Castro, O. Garcia. *2006 IEEE Power Electronics Specialists Conference : proceedings*. 2006. P. 1–6.
74. Medeiros A., Di Maio A., Braun T., Neto A. TENET: Adaptive Service Chain Orchestrator for MEC-Enabled Low-Latency 6DoF Virtual Reality. *IEEE Transactions on Network and Service Management*. 2024. Vol. 21, no. 2. P. 1894–1911. DOI: 10.1109/TNSM.2023.3331755.

75. Computer Systems Resilience in the Presence of Cyber Threats: Taxonomy and Ontology / S. Lysenko, V. Kharchenko, K. Bobrovnikova, R. Shchuka. *Radioelectronic and Computer Systems*. 2020. No. 1. P. 17–28. DOI: 10.32620/reks.2020.1.02.
76. Lysenko S. Self-Adaptive Method for the Computer Systems Resilience in the Presence of Cyberthreats. *Radioelectronic and Computer Systems*. 2019. No. 4. P. 4–16.
77. Carni D. L., Scuro C. et al. Damage Analysis of Concrete Structures by Means of B-Value Technique. *International Journal of Computing*. 2017. Vol. 16, no. 2. P. 82–88.
78. Lipianina-Honcharenko K., Savchyshyn R., Sachenko A. et al. Concept of the Intelligent Guide with AR support. *International Journal of Computing*. 2022. Vol. 21, no. 2. P. 271–277.
79. Perova I., Bodyanskiy Y. Fast Medical Diagnostics Using Autoassociative Neuro-Fuzzy Memory. *International Journal of Computing*. 2017. Vol. 16, no. 1. P. 34–40.
80. Лисенко С. М., Качур А. В. Виклики в технологіях віртуальної реальності: оптимізація архітектури систем віртуальної реальності. *Вісник Хмельницького національного університету. Технічні науки*. 2024. № 1 (77). С. 288–297.
81. Лисенко С. М., Качур А. В. Модель відмовостійкого функціонування систем віртуальної реальності в контексті впливу динамічно змінюваного середовища. *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2025. № 3. С. 499–510. DOI: 10.31891/2219-9365-2025-83-62.
82. Lysenko S., Kachur A. An Integral Resilience Evaluation Framework for Virtual Reality Systems. *Центральноукраїнський науковий вісник. Технічні науки*. 2025. № 12 (43), ч. I. С. 80–89. DOI: 10.32515/2664-262X.2025.12(43).1.80-89.
83. Lysenko S., Kachur A. Challenges Towards VR Technology: VR Architecture Optimization. 2023 IEEE 13th International Conference on Dependable Systems, Services and Technologies (DeSSerT-2023) : proceedings (Athens, Greece, October 13–15, 2023). P. 1–9. DOI: 10.1109/DESSERT61349.2023.10416538.
84. Methods of improving security and resilience of VR systems' architecture / A. Kachur, S. Lysenko, O. Bodnaruk, P. Gaj. *CEUR Workshop Proceedings*. 2024. Vol. 3675. P. 285–304. URL: <https://ceur-ws.org/Vol-3675/paper21.pdf> (дата звернення: 13.08.2026).

85. Lysenko S., Kachur A. A Resilience Assurance Process Model for Enhancing Virtual Reality Architectural Design. 2024 IEEE 14th International Conference on Dependable Systems, Services and Technologies (DeSSerT-2024) : proceedings (Athens, Greece, October 11–13, 2024).
86. Качур А. В., Лисенко С. М. Виклики в технологiях вiртуальної реальностi. Актуальнi проблеми комп'ютерних наук (АПКН – 2023) : матерiали XV Всеукраїнської науково-практичної конференції (Хмельницький, 17–18 листопада 2023 р.). Хмельницький : ХНУ, 2023. С. 121–123.
87. Качур А. В., Лисенко С. М. Модель процесу забезпечення резильєнтностi архiтектури систем вiртуальної реальностi. Актуальнi проблеми комп'ютерних наук (АПКН – 2024) : матерiали XVI Всеукраїнської науково-практичної конференції (Хмельницький, 15–16 листопада 2024 р.). Хмельницький : ХНУ, 2024. С. 340–342.
88. Лисенко С. М., Качур А. В. Модель процесу забезпечення резильєнтностi для оптимiзації архiтектури систем вiртуальної реальностi. Проблеми iнформатики та моделювання (ПiМ-2024) : тези XXIV мiжнародної науково-технiчної конференції (20–23 вересня 2024 р.). 2024. С. 79–81.
89. ISO/IEC 25010:2023. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model. Ed. 2. Geneva : ISO, 2023. 22 p.
90. Stauffert J.-P., Niebling F., Latoschik M. E. Latency and Cybersickness: Impact, Causes, and Measures. A Review. *Frontiers in Virtual Reality*. 2020. Vol. 1. Article 582204. DOI: 10.3389/frvir.2020.582204.
91. Cybersickness in current-generation virtual reality head-mounted displays: systematic review and outlook / P. Caserman, A. Garcia-Agundez, A. Gámez Zerban, S. Göbel. *Virtual Reality*. 2021. Vol. 25, no. 4. P. 1153–1170. DOI: 10.1007/s10055-021-00513-6.
92. Chang E., Kim H. T., Yoo B. Virtual Reality Sickness: A Review of Causes and Measurements. *International Journal of Human–Computer Interaction*. 2020. Vol. 36, no. 17. P. 1658–1682. DOI: 10.1080/10447318.2020.1778351.

93. Factors Associated With Virtual Reality Sickness in Head-Mounted Displays: A Systematic Review and Meta-Analysis / D. Saredakis, A. Szpak, B. Birkhead, H. A. D. Keage, A. Rizzo, T. Loetscher. *Frontiers in Human Neuroscience*. 2020. Vol. 14. Article 96. DOI: 10.3389/fnhum.2020.00096.
94. A narrative review of immersive virtual reality's ergonomics and risks at the workplace: cybersickness, visual fatigue, muscular fatigue, acute stress, and mental overload / A. D. Souchet, D. Lourdeaux, A. Pagani, L. Rebenitsch. *Virtual Reality*. 2023. Vol. 27, no. 1. P. 19–50. DOI: 10.1007/s10055-022-00672-0.
95. A Survey on Metaverse: Fundamentals, Security, and Privacy / Y. Wang, Z. Su, N. Zhang, R. Xing, D. Liu, T. H. Luan, X. Shen. *IEEE Communications Surveys & Tutorials*. 2023. Vol. 25, no. 1. P. 319–352. DOI: 10.1109/COMST.2022.3202047.
96. Virtually secure: A taxonomic assessment of cybersecurity challenges in virtual reality environments / B. Odeleye, G. Loukas, R. Heartfield, G. Sakellari, E. Panaousis, F. Spyridonis. *Computers & Security*. 2023. Vol. 124. Article 102951. DOI: 10.1016/j.cose.2022.102951.
97. IEC 62368-1:2023. Audio/video, information and communication technology equipment – Part 1: Safety requirements. Ed. 4.0. Geneva : IEC, 2023.
98. Wang L., Shi X., Liu Y. Foveated rendering: A state-of-the-art survey. *Computational Visual Media*. 2023. Vol. 9, no. 2. P. 195–228. DOI: 10.1007/s41095-022-0306-4.
99. An integrative view of foveated rendering / B. Mohanto, A. T. Islam, E. Gobbetti, O. Staadt. *Computers & Graphics*. 2022. Vol. 102. P. 474–501. DOI: 10.1016/j.cag.2021.10.010.
100. A Survey on Mobile Augmented Reality With 5G Mobile Edge Computing: Architectures, Applications, and Technical Aspects / Y. Siriwardhana, P. Porambage, M. Liyanage, M. Ylianttila. *IEEE Communications Surveys & Tutorials*. 2021. Vol. 23, no. 2. P. 1160–1192. DOI: 10.1109/COMST.2021.3061981.
101. Yaqoob A., Bi T., Muntean G.-M. A Survey on Adaptive 360° Video Streaming: Solutions, Challenges and Opportunities. *IEEE Communications Surveys & Tutorials*. 2020. Vol. 22, no. 4. P. 2801–2838. DOI: 10.1109/COMST.2020.3006999.

102. Augmented, Mixed, and Virtual Reality-Based Head-Mounted Devices for Medical Education: Systematic Review / S. Barteit, L. Lanfermann, T. Bärnighausen, F. Neuhaus, C. Beiersmann. *JMIR Serious Games*. 2021. Vol. 9, no. 3. Article e29080. DOI: 10.2196/29080.
103. Immersive Virtual Reality for Surgical Training: A Systematic Review / R. Q. Mao, L. Lan, J. Kay, R. Lohre, O. R. Ayeni, D. P. Goel, D. de Sa. *Journal of Surgical Research*. 2021. Vol. 268. P. 40–58. DOI: 10.1016/j.jss.2021.06.045.
104. A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda / J. Radianti, T. A. Majchrzak, J. Fromm, I. Wohlgenannt. *Computers & Education*. 2020. Vol. 147. Article 103778. DOI: 10.1016/j.compedu.2019.103778.
105. Quin F., Weyns D., Gheibi O. Decentralized Self-Adaptive Systems: A Mapping Study. 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS) : proceedings. 2021. P. 18–29. DOI: 10.1109/SEAMS51251.2021.00014.
106. Gheibi O., Weyns D., Quin F. Applying Machine Learning in Self-Adaptive Systems: A Systematic Literature Review. *ACM Transactions on Autonomous and Adaptive Systems*. 2021. Vol. 15, no. 3. Article 9. DOI: 10.1145/3469440.
107. Security and Privacy Controls for Information Systems and Organizations : NIST Special Publication 800-53, Revision 5. Gaithersburg, MD : National Institute of Standards and Technology, 2020. DOI: 10.6028/NIST.SP.800-53r5.
108. Integrating Cybersecurity and Enterprise Risk Management (ERM) : NIST Interagency Report 8286, Revision 1. Gaithersburg, MD : National Institute of Standards and Technology, 2025. DOI: 10.6028/NIST.IR.8286r1.
109. Liu Y., Eckert C. M., Earl C. A review of fuzzy AHP methods for decision-making with subjective judgements. *Expert Systems with Applications*. 2020. Vol. 161. Article 113738. DOI: 10.1016/j.eswa.2020.113738.
110. Blank J., Deb K. pymoo: Multi-Objective Optimization in Python. *IEEE Access*. 2020. Vol. 8. P. 89497–89509. DOI: 10.1109/ACCESS.2020.2990567.

111. Evolutionary Large-Scale Multi-Objective Optimization: A Survey / Y. Tian, L. Si, X. Zhang, R. Cheng, C. He, K. C. Tan, Y. Jin. ACM Computing Surveys. 2021. Vol. 54, no. 8. Article 174. DOI: 10.1145/3470971.
112. Falcón-Cardona J. G., Coello Coello C. A. Indicator-based Multi-objective Evolutionary Algorithms: A Comprehensive Survey. ACM Computing Surveys. 2020. Vol. 53, no. 2. Article 29. DOI: 10.1145/3376916.
113. CloudStrike: Chaos Engineering for Security and Resiliency in Cloud Infrastructure / K. A. Torkura, M. I. H. Sukmana, F. Cheng, C. Meinel. IEEE Access. 2020. Vol. 8. P. 123044–123060. DOI: 10.1109/ACCESS.2020.3007338.
114. Observability and Chaos Engineering on System Calls for Containerized Applications in Docker / J. Simonsson, L. Zhang, B. Morin, B. Baudry, M. Monperrus. Future Generation Computer Systems. 2021. Vol. 122. P. 117–129. DOI: 10.1016/j.future.2021.04.001.
115. ISO/IEC 23090-2:2023. Information technology – Coded representation of immersive media – Part 2: Omnidirectional media format. Ed. 3. Geneva : ISO, 2023.
116. IEEE Std 3079-2020. IEEE Standard for Head-Mounted Display (HMD)-Based Virtual Reality (VR) Sickness Reduction Technology. New York : IEEE, 2021. DOI: 10.1109/IEEESTD.2021.9416950.
117. ISO/IEC/IEEE 15288:2023. Systems and software engineering – System life cycle processes. Ed. 2. Geneva : ISO, 2023.
118. ISO 9241-210:2019. Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems. Geneva : ISO, 2019.
119. Digital twin paradigm: A systematic literature review / C. Semeraro, M. Lezoche, H. Panetto, M. Dassisti. Computers in Industry. 2021. Vol. 130. Article 103469. DOI: 10.1016/j.compind.2021.103469.
120. Chakareski J. Viewport-Adaptive Scalable Multi-User Virtual Reality Mobile-Edge Streaming. IEEE Transactions on Image Processing. 2020. Vol. 29. P. 6330–6342. DOI: 10.1109/TIP.2020.2986547.
121. IoT Multi-Vector Cyberattack Detection Based on Machine Learning Algorithms: Traffic Features Analysis, Experiments, and Efficiency / S. Lysenko, K.

Bobrovnikova, V. Kharchenko, O. Savenko. Algorithms. 2022. Vol. 15, no. 7. Article 239. DOI: 10.3390/a15070239.

122. Security-Informed Safety Analysis of Autonomous Transport Systems Considering AI-Powered Cyberattacks and Protection / O. Illiashenko, V. Kharchenko, I. Babeshko, H. Fesenko, F. Di Giandomenico. Entropy. 2023. Vol. 25, no. 8. Article 1123. DOI: 10.3390/e25081123.

123. AI Cybersecurity Assurance for Autonomous Transport Systems: Scenario, Model, and IMECA-Based Analysis / V. Kharchenko, O. Illiashenko, H. Fesenko, I. Babeshko. Communications in Computer and Information Science. 2022. Vol. 1689. P. 66–79. DOI: 10.1007/978-3-031-20215-5_6.

124. A Full Dive Into Realizing the Edge-Enabled Metaverse: Visions, Enabling Technologies, and Challenges / M. Xu, W. C. Ng, W. Y. B. Lim, J. Kang, Z. Xiong, D. Niyato, Q. Yang, X. Shen, C. Miao. IEEE Communications Surveys & Tutorials. 2023. Vol. 25, no. 1. P. 656–700. DOI: 10.1109/COMST.2022.3221119.

125. All One Needs to Know about Metaverse: A Complete Survey on Technological Singularity, Virtual Ecosystem, and Research Agenda / L.-H. Lee, T. Braud, P. Zhou, L. Wang, D. Xu, Z. Lin, A. Kumar, C. Bermejo, P. Hui. arXiv preprint. 2021. arXiv:2110.05352. DOI: 10.48550/arXiv.2110.05352.

126. ISO 22301:2019. Security and resilience – Business continuity management systems – Requirements. Geneva : ISO, 2019 (зі зміною Amd 1:2024).

127. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection – Information security management systems – Requirements. Ed. 3. Geneva : ISO, 2022.

128. ISO/IEC TR 24028:2020. Information technology – Artificial intelligence – Overview of trustworthiness in artificial intelligence. Geneva : ISO, 2020.

129. Multi-access Edge Computing (MEC); Framework and Reference Architecture : ETSI GS MEC 003 V4.1.1. Sophia Antipolis : ETSI, 2025.

130. Extended Reality (XR) in 5G : 3GPP TR 26.928, Release 17. Sophia Antipolis : 3GPP, 2022.

131. 3GPP Virtual Reality profiles for streaming applications : 3GPP TS 26.118, Release 17. Sophia Antipolis : 3GPP, 2021.
132. OpenXR 1.1 Specification / The Khronos Group. Version 1.1, 15 April 2024. URL: <https://registry.khronos.org/OpenXR/> (дата звернення: 13.08.2026).
133. Weyns D., Iftikhar M. U. ActivFORMS: A Formally Founded Model-based Approach to Engineer Self-adaptive Systems. ACM Transactions on Software Engineering and Methodology. 2023. Vol. 32, no. 1. Article 12. DOI: 10.1145/3522585.
134. Explaining Architectural Design Tradeoff Spaces: A Machine Learning Approach / J. Cámara, M. Silva, D. Garlan, B. Schmerl. Software Architecture : proceedings of the 15th European Conference on Software Architecture (ECSA 2021). Cham : Springer, 2021. P. 49–65. (Lecture Notes in Computer Science ; vol. 12857). DOI: 10.1007/978-3-030-86044-8_4.

ДОДАТОК А.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковані основні результати дисертації

1. Лисенко С., Качур А. Виклики в технологіях віртуальної реальності: оптимізація архітектури систем віртуальної реальності. ВОТТП, ХНУ. 2024. №1(77). С. 288-297. <https://doi.org/10.31891/2219-9365-2024-77-39>
2. S. Lysenko, A. Kachur. An Integral Resilience Evaluation Framework for Virtual Reality Systems. Центральнoукраїнський науковий вісник. Технічні науки, ЦНТУ. 2025. №12(43). С.80-89 [https://doi.org/10.32515/2664-262X.2025.12\(43\).1.80-89](https://doi.org/10.32515/2664-262X.2025.12(43).1.80-89)
3. Лисенко С., Качур А. Модель відмовостійкого функціонування систем віртуальної реальності в контексті впливу динамічно змінюваного середовища. Measuring and Computing Devices in Technological Processes. 2025. № 3 (83). 499–510. <https://doi.org/10.31891/2219-9365-2025-83-62>
4. Лисенко С., Качур А., Корецька Л. Метод синтезу резильєнтних архітектур систем віртуальної та доповненої реальності. Центральнoукраїнський науковий вісник. Технічні науки. 2026. №13(44). С. 59-72 [https://doi.org/10.32515/2664-262X.2026.13\(44\).59-72](https://doi.org/10.32515/2664-262X.2026.13(44).59-72)

Праці, які засвідчують апробацію матеріалів дисертації

5. Sergii Lysenko, Artem Kachur. Challenges Towards VR Technology: VR Architecture Optimization. Proceedings of 2023 IEEE 13th International Conference on Dependable Systems, Services and Technologies (DeSSerT-2023, Athens, Greece, October 13-15, 2023) – pp.1-9, DOI:10.1109/DESSERT61349.2023.10416538.
6. Artem Kachur, Sergii Lysenko, Oleh Bodnaruk, Piotr Gaj. Methods of improving security and resilience of VR systems' architecture. CEUR-WS. Vol. 3675. pp. 285-304. URL: <https://ceur-ws.org/Vol-3675/paper21.pdf>.
7. Sergii Lysenko, Artem Kachur. A Resilience Assurance Process Model for Enhancing Virtual Reality Architectural Design. Proceedings of 2024 IEEE 14th International Conference on Dependable Systems, Services and Technologies (DeSSerT-2024, Athens, Greece, October 11-13, 2024).
8. Качур А.В., Лисенко С.М. Виклики в технологіях віртуальної реальності. XV Всеукраїнська науково-практична конференція “Актуальні проблеми комп’ютерних наук (АПКН – 2023)”, м. Хмельницький, ХНУ, 17-18 листопада 2023. С. 121-123.

9. Качур А.В., Лисенко С.М. Модель процесу забезпечення резильєнтності архітектури систем віртуальної реальності. XVI Всеукраїнська науково-практична конференція “Актуальні проблеми комп’ютерних наук (АПКН – 2024)”, м. Хмельницький, ХНУ, 15-16 листопада 2024. С. 340-342.

10. Лисенко С. М., Качур А. В. Модель процесу забезпечення резильєнтності для оптимізації архітектури систем віртуальної реальності. Проблеми інформатики та моделювання (ПІМ-2024) : тези XXIV міжнародної науково-технічної конференції (20–23 вересня 2024 р.). 2024. С. 79–81.

ДОДАТОК Б.

АКТИ ВПРОВАДЖЕННЯ



«Затверджую»

Директор

ПП «АВІВІ»

В'ячеслав АСКЕРОВ

«15» Травня

2026 р.

АКТ

впровадження результатів дисертаційної роботи

Артема КАЧУРА

Комісія в складі:

директор

В'ячеслав АСКЕРОВ

інженер-програміст

В'ячеслав НАГНИБІДА

інженер-програміст

Дмитро ІВАНОВ

склала цей акт про те, що результати дисертаційної роботи фізичної особи — підприємця (ФОП) Артема КАЧУРА на тему «Методи та засоби забезпечення резильєнтності систем доповненої та віртуальної реальності» впроваджено в діяльність ПП «АВІВІ».

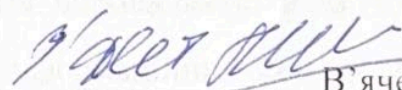
Результати дослідження використано під час проєктування та тестування програмних компонентів XR-застосунків. Сформовано модельні сценарії погіршення мережових умов, перевантаження, теплових обмежень і змін поведінки користувача та виконано порівняння способів збереження працездатності застосунку. Для виконання цих завдань на підприємстві використано такі результати:

1. Модель резильєнтного функціонування XR-системи та метод оцінювання її стану, які поєднали апаратні, програмні, інформаційні й користувацькі компоненти в матриці взаємодій. Застосування інтегрального показника та локалізації критичної взаємодії дало змогу порівняти тестові

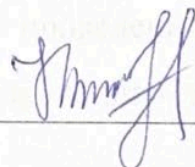
сценарії й визначити фактори деградації, на які було спрямовано зміни програмної конфігурації.

2. Метод синтезу мітигаційних стратегій, за допомогою якого порівняно варіанти кешування, буферизації, розподілу обчислень, зміни деталізації сцени, адаптації інтерфейсу та відновлення за контрольними точками. Багатокритеріальне оцінювання конфігурацій з урахуванням обмежень дозволило дослідити співвідношення між стабільністю продуктивності, доступністю, часом відновлення та модельними показниками безпеки користувача.

3. Архітектуру та програмний засіб VR Resilience, у якому поєднано обчислювальне ядро TypeScript, інтерфейс React і десктопне пакування Electron. Засіб використано для створення демонстраційних прототипів і відтворюваних модельних перевірок XR-застосунків, візуалізації впливу параметрів середовища та зіставлення базових і мітигаційних конфігурацій під час підготовки тестових сценаріїв.



В'ячеслав АСКЕРОВ



В'ячеслав НАГНИБІДА



Дмитро ІВАНОВ



«Затверджую»

профектор з науково-педагогічної роботи

Віктор ЛОПАТОВСЬКИЙ

«13» травня 2026 р.

АКТ

Впровадження результатів дисертаційної роботи

Артема КАЧУРА

Комісія Хмельницького національного університету в складі:

завідувача кафедри комп'ютерної інженерії та інформаційних систем, доктора філософії Ольги ПАВЛОВОЇ, доктора технічних наук Єлизавети ГНАТЧУК та доктора філософії Павла РЕГІДИ склала цей акт про те, що результати дисертаційної роботи фізичної особи – підприємця (ФОП) Артема КАЧУРА на тему «Методи та засоби забезпечення резильєнтності систем доповненої та віртуальної реальності» впроваджено в навчальний процес кафедри комп'ютерної інженерії та інформаційних систем Хмельницького національного університету під час проведення практичних занять із дисциплін «Системне програмне забезпечення» та «Об'єктно-орієнтовані технології програмування».

У навчальному процесі використано такі результати дослідження, одержані автором особисто:

1) модель резильєнтного функціонування систем доповненої та віртуальної реальності, яка враховувала вплив мережевих умов, навколишнього середовища, поведінки користувачів, системного навантаження та апаратних обмежень на апаратне забезпечення, програмне забезпечення, дані й користувачів; метод оцінювання стану системи на основі матриці двадцяти оцінювальних функцій та інтегрального показника з вагами, визначеними методом нечіткого аналізу ієрархій;

2) метод синтезу мітигаційних стратегій, який передбачав багатокритеріальне оцінювання конфігурацій системи за показниками надійності, доступності, відмовостійкості, цілісності, стабільності продуктивності, часу відновлення та безпеки користувача. Метод дав змогу дослідити компроміси між критеріями й обрати Парето-ефективні конфігурації в умовах змін середовища та обмеженості ресурсів;

3) архітектуру та програмну реалізацію засобу VR Resilience з обчислювальним ядром TypeScript, інтерфейсом React і пакуванням Electron. Засіб забезпечив візуалізацію матриці взаємодій, обчислення вагових коефіцієнтів, оцінювання модельних сценаріїв деградації та порівняння мітигаційних конфігурацій. Програмні модулі ілюстрували поділ обчислювальної логіки та інтерфейсу, типізацію вхідних даних і перевірку коректності обчислень.



Ольга ПАВЛОВА

Єлизавета ГНАТЧУК

Павло РЕГІДА

«Затверджую»

Генеральний директор

ТОВ "Хорайзен Дайнамікс"

Олександр МЕЛЬНИЧЕНКО

«18» травня 2026 р.



АКТ

впровадження результатів дисертаційної роботи

Артема КАЧУРА

Комісія в складі:

Генеральний директор

Олександр МЕЛЬНИЧЕНКО

Технічний директор

Артем ТИХОНЮК

склала цей акт про те, що результати дисертаційної роботи фізичної особи – підприємця (ФОП) Артема КАЧУРА на тему «Методи та засоби забезпечення резильєнтності систем доповненої та віртуальної реальності» впроваджено в діяльність ТОВ «Хорайзен Дайнамікс».

Результати дослідження використано для модельного тестування мережевої та обчислювальної інфраструктури XR-сервісів перед вибором конфігурації їх розгортання. Перевірено сценарії збільшення затримки, втрат пакетів, обмеження пропускну здатності та пікового навантаження. Для виконання цих завдань на підприємстві використано такі результати:

1. Метод оцінювання стану XR-системи на основі матриці впливів груп факторів на компоненти системи та інтегрального показника з нечітким зважуванням. Застосування методу дало змогу зіставити модельні мережеві й навантажувальні сценарії за єдиною шкалою та визначити взаємодії, які сформували найбільший зважений дефіцит резильєнтності.

2. Метод синтезу мітигаційних стратегій, за допомогою якого порівняно конфігурації резервування каналів, розподілу обчислень, кешування, групування завдань і збереження контрольних точок. Враховано ресурсні обмеження та одночасно оцінено надійність, доступність, відмовостійкість, цілісність, стабільність продуктивності, час відновлення й безпеку користувача, що дозволило дослідити компроміси між конфігураціями.

3. Програмний засіб VR Resilience, за допомогою якого відтворено базові й деградовані стани за заданими параметрами середовища, візуалізовано критичні взаємодії та зіставлено мітигаційні конфігурації. Засіб використано для підготовки модельних сценаріїв перевірки XR-сервісів і попереднього обґрунтування налаштувань мережевих та обчислювальних ресурсів перед випробуваннями на цільовій інфраструктурі.



Олександр МЕЛЬНИЧЕНКО

Артем ТИХОНЮК

ДОДАТОК В.

ВИХІДНІ ДАНІ ТА КОНТРОЛЬНІ РОЗРАХУНКИ

У додатку наведено параметри, необхідні для відтворення розрахунків обчислювального ядра VR Resilience, та результати додаткових перевірок, виконаних 7 вересня 2026 року. Перевірки охоплюють модельні сценарії, граничні значення функцій, нормування ваг, детермінованість, монотонність і стійкість інтегрального показника.

В.1 Параметри та сценарії розрахунку

Параметри відповідають об'єкту DEFAULT_PARAMS модуля evaluationFunctions.ts. Незаписані зміни сценаріїв у таблиці В.2 зберігають базові значення таблиці В.1. Літери H, S, D, U позначають апаратне забезпечення, програмне забезпечення, дані та користувачів.

Таблиця В.1 – Повний базовий профіль оцінювання стану

Позначення у кодi	Зміст	Одиниця	Значення
B	Пропускна здатність	Мбіт/с	100
B_req	Необхідна пропускна здатність	Мбіт/с	100
L	Затримка	мс	50
L_req	Необхідна затримка	мс	50
L_max	Гранична затримка	мс	100
J	Джиттер	мс	5
J_max	Граничний джиттер	мс	20
P_loss	Частка втрат пакетів	частка	0,02
P	Споживана потужність	Вт	50
P_ref	Опорна потужність	Вт	80

Продовження таблиці В.1

Позначення у кодi	Зміст	Одиниця	Значення
T_env	Температура довкілля	°C	25
T_H_max	Температурний поріг обладнання	°C	70
N_noise	Рівень шуму	дБ	30
N_max	Граничний рівень шуму	дБ	100
L_light	Освітленість	лк	300
L_opt	Опорна освітленість	лк	300
T_comf	Комфортна температура	°C	25
dT_U_max	Допустиме відхилення температури довкілля	°C	10
M	Інтенсивність рухів	Гц	8
M_max	Гранична інтенсивність рухів	Гц	30
M_comf	Комфортна інтенсивність рухів	Гц	20
F	Частота входних подій	1/с	10
F_max	Гранична частота подій	1/с	60
V	Обсяг користувацьких даних	ГБ	2
V_max	Граничний обсяг користувацьких даних	ГБ	8
C_load	Системне навантаження	частка	0,5
C_H_max	Поріг навантаження H	частка	1
C_S_max	Поріг навантаження S	частка	1
C_D_max	Поріг навантаження D	частка	1

Закінчення таблиці В.1

Позначення у кодi	Зміст	Одиниця	Значення
C_U_max	Поріг навантаження U	частка	1
D_size	Обсяг даних сеансу	ГБ	3
D_max	Граничний обсяг даних сеансу	ГБ	8
T_hw	Температура обладнання	°C	40
T_crit	Критична температура	°C	90
T_safe	Безпечна температура	°C	70
B_life	Залишковий ресурс батареї	%	80
b_min	Мінімальний ресурс батареї	%	20
P_max	Гранична потужність	Вт	100
dT_HW_max	Допустиме відхилення температури обладнання	°C	40

Таблиця В.2 – Зміни базових параметрів для контрольних сценаріїв

Сценарій	Зміни параметрів
Default	Без змін
Ideal	B=500; L=10; J=2; P_loss=0.001; T_env=25; N_noise=20; L_light=500; M=5; F=5; V=1; C_load=0.2; D_size=2; T_hw=30; B_life=95; P=30
Network failure	B=5; L=350; J=80; P_loss=0.4
Stress	B=20; L=180; J=50; P_loss=0.15; T_env=42; N_noise=85; L_light=60; M=45; F=70; V=15; C_load=0.9; D_size=15; T_hw=65; B_life=15; P=95

В.2 Контрольні результати та перевірка ваг

Таблиця В.3 – Результати повторного обчислення сценаріїв

Сценарій	R_{VR}	Зміна до Default, %	Критична пара
Default	0,865451	0,00	1, U
Ideal	0,943859	9,06	2, S
Network failure	0,516510	-40,32	1, H
Stress	0,183804	-78,76	1, H

Критичну пару визначено за найбільшим зваженим дефіцитом $w_{ij}(1 - f_{ij})$, як у формулі (2.40). Значення R_{VR} відтворюють таблицю 3.8 після округлення до трьох десяткових знаків.

Таблиця В.4 – Перевірка узгодженості матриць нечіткого АНР

Компонент	CR	Поріг	Висновок
H	0,013860	0,1	Узгоджена
S	0,023570	0,1	Узгоджена
D	0,042173	0,1	Узгоджена
U	0,026871	0,1	Узгоджена

Таблиця В.5 – Контроль мітигаційних конфігурацій при $T = 100$

Показник	Базова	Оптимізована
Надійність	0,496585	0,714802
Доступність	0,955414	0,994485
Час відновлення	6,666667	1,651786
Відмовостійкість	0,746574	0,993384
Цілісність	1,000000	1,000000
Стабільність	0,585365	0,826424
Безпека користувача	0,196205	0,664635
S_{agg}	0,663830	0,873020

Для таблиці В.5 функцію `computeObjectives` викликано без `envBias`, тобто з базовими умовами перетворювача `objectives.ts`. Вони відрізняються від `DEFAULT_PARAMS` оцінювання стану. Оптимізовані компоненти M задаються у підрозділі 3.4.1, решта дорівнюють `DEFAULT_MITIGATION`. S_{agg} є середнім арифметичним семи нормованих критеріїв; час відновлення перетворено на $1 - \min(1; RT/20)$.

В.3 Граничні та повторні перевірки

Таблиця В.6 – Перевірки порогів та особливих значень

Функція	Умова	Очікувано	Отримано
$f1S$	Мережева затримка на порозі; $L=100, J=20$	1,000	1,000
$f1S$	Мережева затримка вище порогу; $L=100.001, J=20$	0,000	0,000
$f2H$	Температура довкілля на порозі; $T_{env}=70$	1,000	1,000

Функція	Умова	Очікувано	Отримано
f2H	Температура довкілля вище порогу; $T_{env}=70.001$	0,000	0,000
f3S	Частота подій на порозі; $F=60$	1,000	1,000
f3S	Частота подій вище порогу; $F=60.001$	0,000	0,000
f4S	Навантаження на порозі; $C_{load}=1$	1,000	1,000
f4S	Навантаження вище порогу; $C_{load}=1.001$	0,000	0,000
f5H	Мінімальний заряд батареї; $B_{life}=20$, $T_{hw}=70$	1,000	1,000
f5H	Заряд нижче мінімального; $B_{life}=19.999$	0,000	0,000
f5D	Безпечна температура; $T_{hw}=70$, $D_{size}=3$	1,000	1,000
f5D	Середина теплового інтервалу; $T_{hw}=80$, $D_{size}=3$	0,500	0,500
f5D	Критична температура; $T_{hw}=90$, $D_{size}=3$	0,000	0,000
f3D	Нульовий обсяг користувацьких даних; $V=0$	1,000	1,000
f5D	Нульовий обсяг даних сеансу; $D_{size}=0$, $T_{hw}=40$	1,000	1,000
f1H	Захист від нульової затримки; $L=0$	0,000	0,000

Для серії з 1000 профілів застосовано лінійний конгруентний генератор із початковим станом 20260907: множник 1664525, приріст 1013904223 і модуль 2^{32} . Діапазони змінних наведено у програмі Г.4; вони охоплюють значення, доступні в інтерфейсі. Затримку та потужність обмежено знизу додатними значеннями 1 мс і 1 Вт. Нормувальні величини залишалися базовими.

Таблиця В.7 – Результати перевірок обчислювальних властивостей

Перевірка	Результат
Нормування ваг	Сума 20 ваг дорівнює 1 з похибкою $< 10^{-12}$
Додатність ваг	Усі 20 ваг додатні
Ранжування сценаріїв	Ideal > Default > Network failure > Stress
Обмеженість 20 функцій	1000 профілів, 20 000 значень на [0; 1]
Детермінованість	Максимальна різниця 0
Монотонність та похідна	Максимальна похибка $2.108e-16$
Стійкість до збурення функцій	Порушень межі (2.41) немає

Усі 16 граничних випадків дали очікувані значення. У серії з 1000 профілів усі 20 000 значень оцінювальних функцій та всі інтегральні оцінки були скінченними й належали інтервалу [0; 1]. Для перевірки монотонності по черзі збільшувалася одна оцінка на величину до 0,1; для перевірки (2.41) усі оцінки збурювалися в межах $\pm 0,05$ із обмеженням результату до [0; 1].

Наведені перевірки підтверджують обчислювальні властивості реалізації на заданих вхідних даних. Вони не є випробуваннями фізичних гарнітур, не встановлюють безпечність конкретного пристрою та не замінюють калібрування моделі за реальною телеметрією.

ДОДАТОК Г.

ПРОГРАМНИЙ КОД ОБЧИСЛЮВАЛЬНОГО ЯДРА

Наведено вихідний код трьох модулів обчислювального ядра VR Resilience та програму контрольних перевірок. Лістинги відповідають файлам проєкту станом на 7 вересня 2026 року. Програмні позначення і десяткову крапку збережено. Довгі рядки перенесено лише для друкованого подання.

Г.1 Оцінювальні функції та інтегральний показник

Файл: `app/src/engine/evaluationFunctions.ts`

```
SHA-256: 6ee44b2ca77f9aef63a1b4d6922714312600a0c3a5c9e0dfc219816ba8dcbfd1
// 20 evaluation functions f(i,j) from the thesis (formulas 2.9–2.28,
// reviewed_v4).
// Every function is dimensionless and bounded to [0, 1] where 1 = no
// degradation, 0 = full degradation. Each dimensional parameter
// enters only
// through a ratio to a quantity of the same physical nature; the [x]01
// operator (clamp01) bounds the final value of each function.
//
// Parameter symbols follow the thesis:
// Network:      B, B_req, L, L_req, L_max, J, J_max, P_loss, P,
P_ref
// Environment:  T_env, T_H_max, N_noise, N_max, L_light, L_opt,
T_comf, dT_U_max
// User Behavior: M (movement intensity), M_max, M_comf,
//               F (input-event frequency), F_max,
//               V (user data volume), V_max
// System load:  C_load, C_H_max, C_S_max, C_D_max, C_U_max,
D_size, D_max
// HW limits:    T_hw, T_crit, T_safe, b_life, b_min, P_max,
dT_HW_max

export interface EnvironmentParams {
  // Network
  B: number;           // Bandwidth (Mbps)
  B_req: number;       // Required bandwidth of the mission profile (Mbps)
  L: number;           // Latency (ms)
  L_req: number;       // Required latency (ms)
  L_max: number;       // Max tolerable latency budget (ms)
  J: number;           // Jitter (ms)
  J_max: number;       // Max tolerable jitter (ms)
  P_loss: number;      // Packet loss fraction (0-1)
  P: number;           // Actual power draw (W) – also used by f(5,S)
  P_ref: number;       // Reference (nominal) power draw (W)

  // Environment
  T_env: number;       // Ambient temperature (°C)
  T_H_max: number;     // Max temperature tolerated by the hardware (°C) –
// used by f(2,H)
// and f(5,H)
  N_noise: number;     // Background noise (dB)
  N_max: number;       // Max tolerable noise (dB)
```

```

    L_light: number; // Light level (lux)
    L_opt: number;   // Optimal light level (lux)
    T_comf: number;   // Comfortable temperature (°C) – used by f(2,U)
and f(5,U)
    dT_U_max: number; // Max comfortable ambient temp deviation (°C)

    // User behavior
    M: number;        // Movement intensity (Hz)
    M_max: number;    // Max processable movement intensity (Hz)
    M_comf: number;   // Comfortable movement intensity (Hz)
    F: number;        // Input-event frequency (1/s)
    F_max: number;    // Max processable input-event frequency (1/s)
    V: number;        // User data volume (GB)
    V_max: number;    // Max processable user data volume (GB)

    // System load
    C_load: number;   // Current computational load (normalized)
    C_H_max: number;  // Max load tolerated by hardware (C_H,max)
    C_S_max: number;  // Max load at which software keeps required speed
(C_S,max)
    C_D_max: number;  // Max load for data-processing integrity (C_D,max)
    C_U_max: number;  // Max load tolerable for interaction latency
(C_U,max)
    D_size: number;   // Session data volume (GB)
    D_max: number;    // Max tolerable session data volume (GB)

    // Hardware constraints
    T_hw: number;     // Hardware temperature (°C)
    T_crit: number;   // Critical HW temperature (°C)
    T_safe: number;   // Safe HW temperature threshold (°C)
    B_life: number;   // Remaining battery resource (%)
    b_min: number;    // Min acceptable battery resource (%)
    P_max: number;    // Max allowed power draw (W)
    dT_HW_max: number; // Max HW temp deviation for safe/comfortable
contact (°C)
}

export const DEFAULT_PARAMS: EnvironmentParams = {
    B: 100, B_req: 100, L: 50, L_req: 50, L_max: 100, J: 5, J_max: 20,
    P_loss: 0.02, P: 50, P_ref: 80,
    T_env: 25, T_H_max: 70, N_noise: 30, N_max: 100,
    L_light: 300, L_opt: 300, T_comf: 25, dT_U_max: 10,
    M: 8, M_max: 30, M_comf: 20, F: 10, F_max: 60, V: 2, V_max: 8,
    C_load: 0.5, C_H_max: 1.0, C_S_max: 1.0, C_D_max: 1.0, C_U_max: 1.0,
    D_size: 3, D_max: 8,
    T_hw: 40, T_crit: 90, T_safe: 70, B_life: 80, b_min: 20,
    P_max: 100, dT_HW_max: 40,
};

function clamp01(x: number): number {
    return Math.max(0, Math.min(1, x));
}

// --- Network impact (i=1) ---

// f(N,H) – formula 2.9: product of three dimensionless ratios
(bandwidth,
// latency, power to their nominal values), clamped to [0, 1] as a
whole.
export function f1H(p: EnvironmentParams): number {
    if (p.B_req <= 0 || p.L <= 0 || p.P <= 0) return 0;
    return clamp01((p.B / p.B_req) * (p.L_req / p.L) * (p.P_ref / p.P));
}

```

```

}

// f(N,S) – formula 2.10: threshold on latency and jitter budgets.
export function f1S(p: EnvironmentParams): number {
  return (p.L <= p.L_max && p.J <= p.J_max) ? 1 : 0;
}

// f(N,D) – formula 2.11: delivered-packet share times relative
bandwidth.
export function f1D(p: EnvironmentParams): number {
  if (p.B_req <= 0) return 0;
  return clamp01((1 - p.P_loss) * (p.B / p.B_req));
}

// f(N,U) – formula 2.12: latency relative to the tolerable budget.
export function f1U(p: EnvironmentParams): number {
  if (p.L_max <= 0) return 0;
  return clamp01(1 - p.L / p.L_max);
}

// --- Environment impact (i=2) ---

// f(ENV,H) – formula 2.13: ambient temperature threshold T_H,max.
export function f2H(p: EnvironmentParams): number {
  return p.T_env <= p.T_H_max ? 1 : 0;
}

// f(ENV,S) – formula 2.14: noise relative to the tolerable level.
export function f2S(p: EnvironmentParams): number {
  if (p.N_max <= 0) return 0;
  return clamp01(1 - p.N_noise / p.N_max);
}

// f(ENV,D) – formula 2.15: light level relative to the required one.
export function f2D(p: EnvironmentParams): number {
  if (p.L_opt <= 0) return 0;
  return clamp01(p.L_light / p.L_opt);
}

// f(ENV,U) – formula 2.16: ambient temperature deviation from
comfort.
export function f2U(p: EnvironmentParams): number {
  if (p.dT_U_max <= 0) return 0;
  return clamp01(1 - Math.abs(p.T_env - p.T_comf) / p.dT_U_max);
}

// --- User behavior impact (i=3) ---

// f(B,H) – formula 2.17: movement intensity relative to the
processable limit.
export function f3H(p: EnvironmentParams): number {
  if (p.M_max <= 0) return 0;
  return clamp01(1 - p.M / p.M_max);
}

// f(B,S) – formula 2.18: threshold on the input-event frequency
F_max.
export function f3S(p: EnvironmentParams): number {
  return p.F <= p.F_max ? 1 : 0;
}

```

```

// f(B,D) – formula 2.19: max processable user data volume over the
actual one.
export function f3D(p: EnvironmentParams): number {
  if (p.V <= 0) return 1;
  return clamp01(p.V_max / p.V);
}

// f(B,U) – formula 2.20: movement intensity relative to the
comfortable one.
export function f3U(p: EnvironmentParams): number {
  if (p.M_comf <= 0) return 0;
  return clamp01(1 - p.M / p.M_comf);
}

// --- System load impact (i=4) ---

// f(S,H) – formula 2.21: load relative to the hardware limit C_H,max.
export function f4H(p: EnvironmentParams): number {
  if (p.C_H_max <= 0) return 0;
  return clamp01(1 - p.C_load / p.C_H_max);
}

// f(S,S) – formula 2.22: threshold on the software load limit
C_S,max.
export function f4S(p: EnvironmentParams): number {
  return p.C_load <= p.C_S_max ? 1 : 0;
}

// f(S,D) – formula 2.23: relative data volume times relative load.
export function f4D(p: EnvironmentParams): number {
  if (p.D_max <= 0 || p.C_D_max <= 0) return 0;
  return clamp01(1 - (p.D_size / p.D_max) * (p.C_load / p.C_D_max));
}

// f(S,U) – formula 2.24: load relative to the interaction-latency
limit C_U,max.
export function f4U(p: EnvironmentParams): number {
  if (p.C_U_max <= 0) return 0;
  return clamp01(1 - p.C_load / p.C_U_max);
}

// --- Hardware constraints impact (i=5) ---

// f(HW,H) – formula 2.25: temperature threshold and minimal battery
resource.
export function f5H(p: EnvironmentParams): number {
  return (p.T_hw <= p.T_H_max && p.B_life >= p.b_min) ? 1 : 0;
}

// f(HW,S) – formula 2.26: power draw relative to the allowed maximum.
export function f5S(p: EnvironmentParams): number {
  if (p.P_max <= 0) return 0;
  return clamp01(1 - p.P / p.P_max);
}

// f(HW,D) – formula 2.27: min of thermal derating (between T_safe and
T_crit)
// and storage headroom D_max/D_size.
export function f5D(p: EnvironmentParams): number {
  const denom = p.T_crit - p.T_safe;
  const thermal = denom > 0 ? clamp01((p.T_crit - p.T_hw) / denom) :
0;

```

```

    const storage = p.D_size > 0 ? clamp01(p.D_max / p.D_size) : 1;
    return Math.min(thermal, storage);
}

// f(HW,U) – formula 2.28: HW temperature deviation from the
// comfortable one.
export function f5U(p: EnvironmentParams): number {
    if (p.dT_HW_max <= 0) return 0;
    return clamp01(1 - Math.abs(p.T_hw - p.T_comf) / p.dT_HW_max);
}

// All 20 functions indexed as [factor_index][component_index]
export type EvalFn = (p: EnvironmentParams) => number;

export const ALL_FUNCTIONS: EvalFn[][] = [
    [f1H, f1S, f1D, f1U], // Network
    [f2H, f2S, f2D, f2U], // Environment
    [f3H, f3S, f3D, f3U], // User behavior
    [f4H, f4S, f4D, f4U], // System load
    [f5H, f5S, f5D, f5U], // HW constraints
];

export function computeMatrix(p: EnvironmentParams): number[][] {
    return ALL_FUNCTIONS.map(row => row.map(fn => fn(p)));
}

// Integral resilience metric  $R_{vr} = \sum w(i,j) * f(i,j)$  (formula 2.30)
export function computeResilience(matrix: number[][], weights:
number[][]): number {
    let sum = 0;
    for (let i = 0; i < 5; i++) {
        for (let j = 0; j < 4; j++) {
            sum += weights[i][j] * matrix[i][j];
        }
    }
    return sum;
}

// Critical pair localization (formula 2.40):
// (i*, j*) = argmax w(i,j) * (1 - f(i,j))
export function criticalPair(matrix: number[][], weights: number[][]):
{ i: number; j:
number; deficit: number } {
    let best = { i: 0, j: 0, deficit: -1 };
    for (let i = 0; i < 5; i++) {
        for (let j = 0; j < 4; j++) {
            const d = weights[i][j] * (1 - matrix[i][j]);
            if (d > best.deficit) best = { i, j, deficit: d };
        }
    }
    return best;
}

```

Г.2 Нечіткий аналіз ієрархій

Файл: app/src/engine/fuzzyAHP.ts

SHA-256: 7e4d48106c8ccf7b191c405845f49d9b31aa486b4abb8ea9940dd9302e338c03

```

// Fuzzy AHP weight computation – ported from computeW_fuzzyAHP.m
// Computes the 5×4 weight matrix W for the VR system

export interface TFN {

```

```

    l: number; // lower bound
    m: number; // most likely
    u: number; // upper bound
}

function invertTFN(t: TFN): TFN {
    return { l: 1 / t.u, m: 1 / t.m, u: 1 / t.l };
}

// Default TFN pairwise comparison matrices (one per component)
// Each is 5x5 (5 factors compared pairwise)
export function getDefaultTFNMatrix(componentIdx: number): TFN[][] {
    const n = 5;
    const A: TFN[][] = Array.from({ length: n }, () =>
        Array.from({ length: n }, () => ({ l: 1, m: 1, u: 1 })))
    );

    // Predefined reasonable comparisons per component
    const presets: Record<number, number[][]> = {
        // Hardware (numeric example of thesis §2.2.2): network >
        environment >
        // user behavior > system load > HW constraints
        0: [[0,2,3,6,8],[0,0,3,5,7],[0,0,0,2,3],[0,0,0,0,2],[0,0,0,0,0]],
        // Software: load most important, then network, user behavior,
        environment, HW
        1: [[0,1,3,4,6],[0,0,3,4,5],[0,0,0,2,3],[0,0,0,0,3],[0,0,0,0,0]],
        // Data: network most important, then load, HW, environment, user
        2: [[0,1,5,6,7],[0,0,3,4,5],[0,0,0,2,4],[0,0,0,0,3],[0,0,0,0,0]],
        // Users: environment most important, then user behavior, network,
        load, HW
        3: [[0,1,3,5,6],[0,0,3,4,5],[0,0,0,3,4],[0,0,0,0,2],[0,0,0,0,0]],
    };

    const p = presets[componentIdx] || presets[0];
    for (let i = 0; i < n; i++) {
        for (let j = i + 1; j < n; j++) {
            const base = p[i][j] || 1;
            A[i][j] = { l: Math.max(1, base - 1), m: base, u: base + 1 };
            A[j][i] = invertTFN(A[i][j]);
        }
    }
    return A;
}

export interface FuzzyAHPResult {
    tfnMatrix: TFN[][];
    columnSums: TFN[];
    normalizedMatrix: TFN[][];
    fuzzyWeights: TFN[];
    crispWeights: number[];
    normalizedWeights: number[];
}

export function computeFuzzyAHP(A: TFN[][]): FuzzyAHPResult {
    const n = A.length;

    // Column sums
    const columnSums: TFN[] = Array.from({ length: n }, () => ({ l: 0,
m: 0, u: 0 }));
    for (let j = 0; j < n; j++) {
        for (let i = 0; i < n; i++) {
            columnSums[j].l += A[i][j].l;

```

```

        columnSums[j].m += A[i][j].m;
        columnSums[j].u += A[i][j].u;
    }
}

// Normalize: l/sum_u, m/sum_m, u/sum_l (as in the Matlab code)
const normalizedMatrix: TFN[][] = Array.from({ length: n }, () =>
    Array.from({ length: n }, () => ({ l: 0, m: 0, u: 0 })))
);
for (let i = 0; i < n; i++) {
    for (let j = 0; j < n; j++) {
        normalizedMatrix[i][j] = {
            l: A[i][j].l / columnSums[j].u,
            m: A[i][j].m / columnSums[j].m,
            u: A[i][j].u / columnSums[j].l,
        };
    }
}

// Row averages → fuzzy weights
const fuzzyWeights: TFN[] = Array.from({ length: n }, () => ({ l: 0,
m: 0, u: 0 }));
for (let i = 0; i < n; i++) {
    for (let j = 0; j < n; j++) {
        fuzzyWeights[i].l += normalizedMatrix[i][j].l;
        fuzzyWeights[i].m += normalizedMatrix[i][j].m;
        fuzzyWeights[i].u += normalizedMatrix[i][j].u;
    }
    fuzzyWeights[i].l /= n;
    fuzzyWeights[i].m /= n;
    fuzzyWeights[i].u /= n;
}

// Defuzzification: crisp = (l + m + u) / 3
const crispWeights = fuzzyWeights.map(w => (w.l + w.m + w.u) / 3);

// Normalize to sum = 1
const total = crispWeights.reduce((a, b) => a + b, 0);
const normalizedWeights = crispWeights.map(w => total > 0 ? w /
total : 1 / n);

return {
    tfnMatrix: A,
    columnSums,
    normalizedMatrix,
    fuzzyWeights,
    crispWeights,
    normalizedWeights,
};
}

// Consistency ratio of the modal (crisp) slice of a TFN comparison
matrix.
// CR = CI / RI, CI = ( $\lambda_{\max}$  - n) / (n - 1), RI = 1.12 for n = 5.
export function consistencyRatio(A: TFN[][]): number {
    const n = A.length;
    const modal = A.map(row => row.map(t => t.m));
    // Principal eigenvalue via power iteration
    let v = Array(n).fill(1 / n);
    let lambda = n;
    for (let iter = 0; iter < 200; iter++) {

```

```

    const w = modal.map(row => row.reduce((s, a, j) => s + a * v[j],
0));
    const norm = w.reduce((a, b) => a + Math.abs(b), 0);
    const vNew = w.map(x => x / norm);
    lambda = norm;
    v = vNew;
  }
  const CI = (lambda - n) / (n - 1);
  const RI = 1.12;
  return CI / RI;
}

// Compute full 5x4 W matrix (weights for each factor*component pair)
export function computeFullWeightMatrix(): { W: number[][]; details:
FuzzyAHPResult[] }
{
  const details: FuzzyAHPResult[] = [];
  // W[component][factor] – but we want W[factor][component] for the
resilience formula
  const Wt: number[][] = [];

  for (let c = 0; c < 4; c++) {
    const A = getDefaultTFNMatrix(c);
    const result = computeFuzzyAHP(A);
    details.push(result);
    Wt.push(result.normalizedWeights);
  }

  // Transpose: W[factor][component]
  const Wraw: number[][] = Array.from({ length: 5 }, (_, i) =>
    Array.from({ length: 4 }, (_, j) => Wt[j][i])
  );

  // Normalize entire matrix so  $\sum w(i,j) = 1$  (thesis formula requires
this)
  const totalW = Wraw.reduce((s, row) => s + row.reduce((a, b) => a +
b, 0), 0);
  const W: number[][] = Wraw.map(row => row.map(v => totalW > 0 ? v /
totalW : 1 / 20));

  return { W, details };
}

```

Г.3 Параметричний перетворювач і цільові функції

Файл: app/src/engine/objectives.ts

SHA-256: e259b805e512e01c0f511923699534dc19145243e2248104bc32af8852c5db36

```

// Mapper & Objectives – ported from evaluate_architecture.m /
run_synthesis.m
// Maps the 20-element mitigation vector to simulation parameters,
// then computes the 7 resilience objectives.
import type { EnvironmentParams } from './evaluationFunctions';

export interface MitigationVector {
  M1_offload_threshold: number; // [0..1]
  M2_thermal_limit: number; // integer, e.g., 35-55
  M3_motion_sens_threshold: number; // [0..10]
  M4_n_threads: number; // integer, 1-16
  M5_display_arch: number; // 1=OLED, 2=uLED
  M6_reserved: number;
  M7_fallback_mode: number; // 1=Normal, 2=LowPrecision
}

```

```

M8_batching_interval: number; // [0..10]
M9_checkpoint_freq: number; // integer, 1-10
M10_offload_ratio: number; // [0..1]
M11_num_links: number; // integer, 1-5
M12_replication_factor: number; // integer, 1-5
M13_prefetch_buffer: number; // integer, 1-20
M14_cache_size: number; // integer, 1-100
M15_mod_coding_level: number; // integer, 1-5
M16_interp_buffer: number; // integer, 1-20
M17_ui_adapt_threshold: number; // [0..3]
M18_locomotion_mode: number; // 1=Teleport, 2=Scaled, 3=Smooth
M19_dynamic_lod: number; // [0.5..2.0]
M20_user_thermal_thresh: number; // integer, 40-60
}

export const DEFAULT_MITIGATION: MitigationVector = {
  M1_offload_threshold: 0.3,
  M2_thermal_limit: 45,
  M3_motion_sens_threshold: 3,
  M4_n_threads: 4,
  M5_display_arch: 1,
  M6_reserved: 1,
  M7_fallback_mode: 1,
  M8_batching_interval: 2,
  M9_checkpoint_freq: 3,
  M10_offload_ratio: 0.2,
  M11_num_links: 2,
  M12_replication_factor: 2,
  M13_prefetch_buffer: 5,
  M14_cache_size: 20,
  M15_mod_coding_level: 2,
  M16_interp_buffer: 5,
  M17_ui_adapt_threshold: 1,
  M18_locomotion_mode: 1,
  M19_dynamic_lod: 1.0,
  M20_user_thermal_thresh: 50,
};

export const MITIGATION_META = [
  { key: 'M1_offload_threshold', label: 'Поріг вивантаження',
    обчислень', min: 0,
    max: 1, step: 0.05, integer: false },
  { key: 'M2_thermal_limit', label: 'Термічний ліміт (°C)',
    min: 35,
    max: 60, step: 1, integer: true },
  { key: 'M3_motion_sens_threshold', label: 'Чутливість до руху',
    min: 0,
    max: 10, step: 0.5, integer: false },
  { key: 'M4_n_threads', label: 'К-ть потоків',
    (конкурентність)', min: 1,
    max: 16, step: 1, integer: true },
  { key: 'M5_display_arch', label: 'Архітектура дисплея',
    (1=OLED, 2=µLED)', min:
    1, max: 2, step: 1, integer: true },
  { key: 'M7_fallback_mode', label: 'Режим відкату ПЗ (1=Норм,
    2=Спрощ)', min:
    1, max: 2, step: 1, integer: true },
  { key: 'M8_batching_interval', label: 'Інтервал пакетування',
    min: 0,
    max: 10, step: 0.5, integer: false },

```

```

    { key: 'M9_checkpoint_freq',          label: 'Частота контрольних
точок',          min: 1,
max: 10, step: 1, integer: true },
    { key: 'M10_offload_ratio',          label: 'Коефіцієнт вивантаження',
min: 0,
max: 0.9, step: 0.05, integer: false },
    { key: 'M11_num_links',              label: 'К-ть мережевих каналів',
min: 1,
max: 5, step: 1, integer: true },
    { key: 'M12_replication_factor',      label: 'Фактор реплікації',
min: 1,
max: 5, step: 1, integer: true },
    { key: 'M13_prefetch_buffer',        label: 'Буфер префетчування',
min: 1,
max: 20, step: 1, integer: true },
    { key: 'M14_cache_size',              label: 'Розмір кешу',
min: 1,
max: 100, step: 1, integer: true },
    { key: 'M15_mod_coding_level',        label: 'Рівень модуляційного
кодування',      min: 1,
max: 5, step: 1, integer: true },
    { key: 'M16_interp_buffer',          label: 'Буфер інтерполяції',
min: 1,
max: 20, step: 1, integer: true },
    { key: 'M17_ui_adapt_threshold',      label: 'Поріг адаптації
інтерфейсу',      min: 0,
max: 3, step: 0.5, integer: false },
    { key: 'M18_locomotion_mode',        label: 'Режим локомоції (1=Тп,
2=Мш, 3=Пл)',      min:
1, max: 3, step: 1, integer: true },
    { key: 'M19_dynamic_lod',            label: 'Динамічний LOD
(масштаб)',        min: 0.5,
max: 2.0, step: 0.1, integer: false },
    { key: 'M20_user_thermal_thresh',    label: 'Комфортний тепловий поріг
(°C)',              min: 40,
max: 60, step: 1, integer: true },
] as const;

```

```
// k-constants from the Matlab code
```

```

const k = {
  offload_bw: 0.1,
  links_bw: 0.2,
  links_lat: 0.3,
  links_loss: 0.4,
  coding_loss: 0.05,
  interp_lat: 0.1,
  fallback_light: 50,
  motion_sens_v: 0.2,
  batching_m: 0.1,
  prefetch_m: 0.15,
  teleport_load: 0.4,
  scaled_load: 0.7,
  smooth_load: 1.0,
  concur_load: 0.05,
  cache_load: 0.01,
  cache_data: 0.1,
  load_temp: 5.0,
  display_temp: -5.0,
  load_bat: 20.0,
  display_bat: 10.0,
  motion_sens_lambda: 0.1,
  display_lambda: 0.002,

```

```

    offload_lambda: 0.3,
};

const BASE = {
    B: 50, L: 100, P_loss: 0.05, T_env: 40,
    N_noise: 60, L_light: 150, V: 150, M: 15,
    C_load_base: 1.5, D_size: 8, C_load_sys: 1.8,
    T_hw_base: 45, B_life_base: 100, P_users: 50,
    lambda_h: 0.01,
};

export interface SimParams {
    B: number; L: number; P_loss: number; P_loss_base: number;
    P_loss_corrected: number;
    L_perceived: number;
    T_env: number; T_env_Tolerance_HW: number;
    N_noise_effective: number; L_light: number;
    V_effective: number; M_effective_S: number; M_effective_D: number;
    C_load_effective_U: number;
    C_load_sys: number; D_size: number;
    T_hw: number; B_life: number; P_users: number;
    T_hw_Tolerance_User: number;
    lambda_h: number;
}

export function mapMitigationToParams(mv: MitigationVector, envBias?:
Partial<EnvironmentParams>): SimParams {
    // Merge live/manual environment params into BASE where available
    const EB = {
        B: envBias?.B ?? BASE.B,
        L: envBias?.L ?? BASE.L,
        P_loss: envBias?.P_loss ?? BASE.P_loss,
        T_env: envBias?.T_env ?? BASE.T_env,
        N_noise: envBias?.N_noise ?? BASE.N_noise,
        L_light: envBias?.L_light ?? BASE.L_light,
        // Motion intensity M (Hz) drives the "V" motion scale of the
        // input-event frequency F drives the batching/prefetch stream
        // "M".
        V: envBias?.M != null ? envBias.M * 10 :
BASE.V,
        M: envBias?.F ?? BASE.M,
        C_load_base: envBias?.C_load != null ? envBias.C_load * 1.5 :
BASE.C_load_base,
        C_load_sys: envBias?.C_load != null ? envBias.C_load * 3.0 :
BASE.C_load_sys,
        D_size: envBias?.D_size ?? BASE.D_size,
        T_hw_base: envBias?.T_hw ?? BASE.T_hw_base,
        B_life_base: envBias?.B_life ?? BASE.B_life_base,
        P_users: BASE.P_users,
        lambda_h: BASE.lambda_h,
    };

    // Network
    const B = EB.B * (1 + k.offload_bw * mv.M1_offload_threshold) * (1 +
k.links_bw *
(mv.M11_num_links - 1));
    const L = EB.L / (1 + k.links_lat * (mv.M11_num_links - 1));
    const P_loss_base = EB.P_loss / (1 + k.links_loss *
(mv.M11_num_links - 1));
    const P_loss_corrected = k.coding_loss * (mv.M15_mod_coding_level -
1);

```

```

const P_loss = Math.max(0.001, P_loss_base - P_loss_corrected);
const L_perceived = L - k.interp_lat * mv.M16_interp_buffer;

// Environment
const T_env = EB.T_env;
const T_env_Tolerance_HW = mv.M2_thermal_limit;
const N_noise_effective = Math.max(0, EB.N_noise -
mv.M17_ui_adapt_threshold * 20);
const light_boost = mv.M7_fallback_mode === 2 ? k.fallback_light :
0;
const L_light = EB.L_light + light_boost;

// User behavior
const V_effective = EB.V / (1 + k.motion_sens_v *
mv.M3_motion_sens_threshold);
const M_effective_S = EB.M / (1 + k.batching_m *
mv.M8_batching_interval);
const M_effective_D = EB.M / (1 + k.prefetch_m *
mv.M13_prefetch_buffer);
const loco_k = mv.M18_locomotion_mode === 1 ? k.teleport_load
: mv.M18_locomotion_mode === 2 ? k.scaled_load : k.smooth_load;
const C_load_effective_U = EB.C_load_base * loco_k;

// System load
let C_load_sys = EB.C_load_sys * (1.0 - mv.M10_offload_ratio);
C_load_sys /= (1 + k.concur_load * mv.M4_n_threads);
C_load_sys /= (1 + k.cache_load * mv.M14_cache_size);
const D_size = EB.D_size / (1 + k.cache_data * mv.M14_cache_size);

const display_temp_saving = mv.M5_display_arch === 2 ?
k.display_temp : 0;
const display_bat_saving = mv.M5_display_arch === 2 ? k.display_bat
: 0;
const T_hw = EB.T_hw_base + k.load_temp * C_load_sys +
display_temp_saving;
const B_life = Math.max(0, Math.min(100,
EB.B_life_base - k.load_bat * C_load_sys + display_bat_saving));
const P_users = EB.P_users * (1 / mv.M19_dynamic_load);

// Hazard rate
let lambda_h = EB.lambda_h;
lambda_h /= (1 + k.motion_sens_lambda *
mv.M3_motion_sens_threshold);
if (mv.M5_display_arch === 2) lambda_h -= k.display_lambda;
lambda_h *= (1 - k.offload_lambda * mv.M1_offload_threshold);
lambda_h = Math.max(1e-6, lambda_h);

return {
  B, L, P_loss, P_loss_base, P_loss_corrected: P_loss_corrected,
  L_perceived, T_env, T_env_Tolerance_HW,
  N_noise_effective, L_light,
  V_effective, M_effective_S, M_effective_D,
  C_load_effective_U, C_load_sys, D_size,
  T_hw, B_life, P_users,
  T_hw_Tolerance_User: mv.M20_user_thermal_thresh,
  lambda_h,
};
}

function nchoosek(n: number, k: number): number {
  if (k > n || k < 0) return 0;
  if (k === 0 || k === n) return 1;

```

```

    let result = 1;
    for (let i = 0; i < k; i++) {
        result = result * (n - i) / (i + 1);
    }
    return Math.round(result);
}

export interface Objectives {
    reliability: number;
    availability: number;
    recoveryTime: number;
    faultTolerance: number;
    integrity: number;
    perfStability: number;
    userSafety: number;
}

export function computeObjectives(mv: MitigationVector, T_simulation:
number = 100,
envBias?: Partial<EnvironmentParams>): Objectives {
    const sp = mapMitigationToParams(mv, envBias);

    // Reliability  $R(t) = \exp(-\lambda \cdot t)$ 
    const reliability = Math.exp(-sp.lambda_h * T_simulation);

    // Recovery Time (simplified – based on checkpoint frequency and
system load)
    const recoveryTime = Math.max(0.1, 10 / mv.M9_checkpoint_freq * (1 +
sp.C_load_sys));

    // Availability  $A = MTTF / (MTTF + MTTR)$ 
    const MTTF = 1 / sp.lambda_h;
    const MTTR = recoveryTime;
    const availability = MTTF / (MTTF + MTTR);

    // Fault Tolerance: k-out-of-n with k=1
    const R_comp = reliability;
    const n = mv.M11_num_links;
    let faultTolerance = 0;
    for (let i = 1; i <= n; i++) {
        faultTolerance += nchoosek(n, i) * Math.pow(R_comp, i) *
Math.pow(1 - R_comp, n -
i);
    }

    // Integrity  $I = 1 - (E_i - E_j) / E_i$ 
    const Ei = sp.P_loss_base * 10000;
    const Ej = sp.P_loss_corrected * 10000;
    const integrity = Ei <= 1e-6 ? 1.0 : 1.0 - Math.max(0, Ei - Ej) /
Ei;

    // Performance Stability (simplified)
    const meanDeviation = sp.C_load_sys * 0.3 + (sp.L_perceived / 200) *
0.3;
    const perfStability = Math.max(0, 1.0 - meanDeviation);

    // User Safety
    const thermalMargin = sp.T_hw_Tolerance_User - sp.T_hw;
    const thermalSafety = Math.max(0, thermalMargin) /
(sp.T_hw_Tolerance_User - 35 +
1e-6);
    const sicknessLoco = mv.M18_locomotion_mode === 1 ? 0.0

```

```

    : mv.M18 locomotion mode === 2 ? 0.3 : 0.7;
    const cybersicknessFactor = (sp.L_perceived / 150) + sicknessLoco;
    const sicknessSafety = Math.max(0, 1.0 - cybersicknessFactor);
    const userSafety = 0.6 * Math.min(1, thermalSafety) + 0.4 *
sicknessSafety;

    return { reliability, availability, recoveryTime, faultTolerance,
integrity,
perfStability, userSafety };
}

```

Г.4 Програма контрольних перевірок

Файл: verification/v10/verify_engine.mjs

```

SHA-256: b9d6728ec122baa4ff1961a82e2d63fb9887d209a9d4ec1de81a0f2c3a7643f9

import fs from 'node:fs';
import assert from 'node:assert/strict';
import * as E from '../app/src/engine/evaluationFunctions.ts';
import * as A from '../app/src/engine/fuzzyAHP.ts';
import * as O from '../app/src/engine/objectives.ts';

// Reproducible verification of the source snapshot used in Appendix
Г.
const W = A.computeFullWeightMatrix().W;
const scenarios = {
  Default: {},
  Ideal: {B:500,L:10,J:2,P_loss:.001,T_env:25,N_noise:20,
L_light:500,M:5,F:5,V:1,C_load:.2,D_size:2,T_hw:30,B_life:95,P:30},
  'Network failure': {B:5,L:350,J:80,P_loss:.4},
  Stress: {B:20,L:180,J:50,P_loss:.15,T_env:42,N_noise:85,
L_light:60,M:45,F:70,V:15,C_load:.9,D_size:15,T_hw:65,B_life:15,P:95}
};
const result = {date:'2026-09-07',seed:20260907,
W,defaults:E.DEFAULT_PARAMS,
  scenarios:[],checks:[],boundary:[],random:[],objectives:[]};
const sum = a=>a.reduce((s,v)=>s+v,0);
function check(name,ok,detail) {
  assert.ok(ok,name); result.checks.push({name,passed:ok,detail});
}
check('Нормування ваг',Math.abs(sum(W.flat())-1)<1e-12,
  'Сума 20 ваг дорівнює 1 з похибкою < 10-12');
check('Додатність ваг',W.flat().every(x=>x>0),'Усі 20 ваг додатні');
for (let c=0;c<4;c++) {
  const matrix=A.getDefaultTFNMatrix(c),cr=A.consistencyRatio(matrix);
  check(`Узгодженість матриці ${['H','S','D','U'][c]}`,`cr<.1`,`CR =
${cr.toFixed(6)}`);
}
for (const [name,changes] of Object.entries(scenarios)) {
  const params={...E.DEFAULT_PARAMS,...changes};
  const matrix=E.computeMatrix(params);
  result.scenarios.push({name,changes,params,matrix,

score:E.computeResilience(matrix,W),critical:E.criticalPair(matrix,W)}
);
}
const
sc=Object.fromEntries(result.scenarios.map(s=>[s.name,s.score]));

```

```

check('Ранжування
сценаріїв',sc.Ideal>sc.Default&&sc.Default>sc['Network
failure']&&sc['Network failure']>sc.Stress,
  'Ideal > Default > Network failure > Stress');
const cases=[
  ['Мережева затримка на порозі','f1S',{L:100,J:20},1],
  ['Мережева затримка вище порогу','f1S',{L:100.001,J:20},0],
  ['Температура довкілля на порозі','f2H',{T_env:70},1],
  ['Температура довкілля вище порогу','f2H',{T_env:70.001},0],
  ['Частота подій на порозі','f3S',{F:60},1],
  ['Частота подій вище порогу','f3S',{F:60.001},0],
  ['Навантаження на порозі','f4S',{C_load:1},1],
  ['Навантаження вище порогу','f4S',{C_load:1.001},0],
  ['Мінімальний заряд батареї','f5H',{B_life:20,T_hw:70},1],
  ['Заряд нижче мінімального','f5H',{B_life:19.999},0],
  ['Безпечна температура','f5D',{T_hw:70,D_size:3},1],
  ['Середина теплового інтервалу','f5D',{T_hw:80,D_size:3},.5],
  ['Критична температура','f5D',{T_hw:90,D_size:3},0],
  ['Нульовий обсяг користувацьких даних','f3D',{V:0},1],
  ['Нульовий обсяг даних сеансу','f5D',{D_size:0,T_hw:40},1],
  ['Захист від нульової затримки','f1H',{L:0},0]
];
for (const [name,fn,changes,expected] of cases) {
  const actual=E[fn]({...E.DEFAULT_PARAMS,...changes});
  assert.ok(Math.abs(actual-expected)<1e-12,name);
  result.boundary.push({name,fn,changes,expected,actual});
}
let state=result.seed;
const
rand=()=>((state=(Math.imul(1664525,state)+1013904223)>>>0)/4294967296
);
let maxMonotonicError=0,maxLipschitzError=0,maxDeterminismError=0;
for(let k=0;k<1000;k++) {
  const
params={...E.DEFAULT_PARAMS,B:1000*rand(),L:1+499*rand(),J:100*rand(),
P_loss:.5*rand(),

P:1+199*rand(),T_env:-
10+60*rand(),N_noise:100*rand(),L_light:2000*rand(),M:60*rand(),

F:120*rand(),V:20*rand(),C_load:1.5*rand(),D_size:50*rand(),T_hw:100*r
and(),B_life:1
00*rand()});
  const
matrix=E.computeMatrix(params),score=E.computeResilience(matrix,W);
  assert.ok(matrix.flat().every(x=>Number.isFinite(x)&&x>=0&&x<=1));
  assert.ok(Number.isFinite(score)&&score>=0&&score<=1);
  maxDeterminismError=Math.max(maxDeterminismError,
    Math.abs(score-E.computeResilience(E.computeMatrix(params),W)));
  const i=k%5,j=k%4,up=matrix.map(r=>[...r]);
  up[i][j]=Math.min(1,up[i][j]+.1);
  const diff=E.computeResilience(up,W)-
score,expected=W[i][j]*(up[i][j]-matrix[i][j]);
  maxMonotonicError=Math.max(maxMonotonicError,Math.abs(diff-
expected));
  assert.ok(diff>=-1e-14);
  const
perturbed=matrix.map(r=>r.map(v=>Math.max(0,Math.min(1,v+(rand()-
.5)*.1))));
  const bound=Math.max(...perturbed.flat().map((x,z)>Math.abs(x-
matrix.flat()[z])));
  maxLipschitzError=Math.max(maxLipschitzError,

```

```

    Math.abs(E.computeResilience(perturbed,W)-score)-bound);
    result.random.push({params,matrix,score});
  }
  check('Обмеженість 20 функцій',true,'1000 профілів, 20 000 значень на
  [0; 1]');
  check('Детермінованість',maxDeterminismError===0,`Максимальна різниця
  ${maxDeterminismError}`);
  check('Монотонність та похідна',maxMonotonicError<1e-12,`Максимальна
  похибка
  ${maxMonotonicError.toExponential(3)}`);
  check('Стійкість до збурення функцій',maxLipschitzError<1e-
  12, 'Порушень межі (2.41)
  немає');
  const
  optimized={...O.DEFAULT_MITIGATION,M1_offload_threshold:.7,M3_motion_s
  ens_threshold:6,

  M4_n_threads:8,M5_display_arch:2,M9_checkpoint_freq:8,M10_offload_rati
  o:.6,M11_num_lin
  ks:4,

  M14_cache_size:60,M15_mod_coding_level:4,M16_interp_buffer:12,M18_loco
  motion_mode:1,
  M19_dynamic_lod:1.5,M20_user_thermal_thresh:55};
  for (const [name,mv] of
  [['Базова',O.DEFAULT_MITIGATION],['Оптимізована',optimized]]) {
    const obj=O.computeObjectives(mv,100);
    const score=sum([obj.reliability,obj.availability,1-
  Math.min(1,obj.recoveryTime/20),
  obj.faultTolerance,obj.integrity,obj.perfStability,obj.userSafety])/7;
    result.objectives.push({name,mv,obj,score,params:O.mapMitigationToPara
  ms(mv)});
  }
  fs.writeFileSync(new
  URL('./verification.json',import.meta.url),JSON.stringify(result,null,
  2));
  console.log(JSON.stringify({checks:result.checks,boundaryCases:cases.l
  ength,

  profiles:1000,scenarios:result.scenarios.map(({name,score})=>({name,sc
  ore})),

  objectiveScores:result.objectives.map(({name,score})=>({name,score}))}
  ,null,2));

```