International Science Group
ISG-KONF.COM

TECHNICAL SCIENCES AND COMPUTER SYSTEMS ENGINEERING: PROMISING DEVELOPMENTS, MODERN METHODS AND NEW TECHNOLOGIES

Collective monograph

ISBN 979-8-90214-596-7
DOI 10.46299/ISG.2026.MONO.TECH.2
BOSTON(USA)-2026

ISBN – 979-8-90214-596-7
DOI – 10.46299/ISG.2026.MONO.TECH.2

Technical sciences and computer systems engineering: promising developments, modern methods and new technologies

Collective monograph

Boston 2026

Library of Congress Cataloging-in-Publication Data

ISBN – 979-8-90214-596-7
DOI – 10.46299/ISG.2026.MONO.TECH.2

Authors – Дубовик Т., Мельникова Н.І., Марікуца У.Б., Пташинський М.О., Poliakov A., Руда Х., Кос І., Basyuk T., Simbirski G., Гузь Г., Косогов О.М.

REVIEWER

Ivan Katerynchuk – Doctor of Technical Sciences, Professor, Honoured Worker of Education of Ukraine, Laureate of the State Prize of Ukraine in Science and Technology, Professor of the Department of Telecommunication and Information Systems of Bohdan Khmelnytskyi National Academy of the State Border Guard Service of Ukraine.

Kostiantyn Dolia – Doctor of Engineering, Department of automobile and transport infrastructure, National Aerospace University "Kharkiv Aviation Institute".

Published by Primedia eLaunch
https://primediaelaunch.com/

Collection of scientific articles published is the scientific and practical publication, which contains scientific articles of students, graduate students, Candidates and Doctors of Sciences, research workers and practitioners from Europe and Ukraine. The articles contain the study, reflecting the processes and changes in the structure of modern science.

The recommended citation for this publication is:
Technical sciences and computer systems engineering: promising developments, modern methods and new technologies: collective monograph / Poliakov A – etc. – International Science Group. – Boston : Primedia eLaunch, 2026. 294 p. Available at : DOI – 10.46299/ISG.2026.MONO.TECH.2

TABLE OF CONTENTS

1.	COMPUTER ENGINEERING	
1.1	Дубовик Т.[1] КІБЕРФІЗИЧНІ МАНІПУЛЯТОРИ. РОЗРОБКА 3D МОДЕЛЕЙ NIRYO ONE ТА 7-DOF MANIPULATOR COPPELIASIM [1] кафедра комп'ютерно-інтегрованих технологій та робототехніки, Український державний університет науки і технологій ННІ УДХТУ, Дніпро, Україна	7
1.1.1	ЗАГАЛЬНІ ВІДОМОСТІ ПРО КІБЕРФІЗИЧНІ МАНІПУЛЯТОРИ	8
1.1.2	РОБОТ NIRYO ONE	15
1.1.3	7-DOF МАНІПУЛЯТОР	34
2.	COMPUTER SCIENCE	
2.1	Мельникова Н.І.[1,2] , Марікуца У.Б.[3], Пташинський М.О.[4] ОЦІНКА ПІДХОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ КЛАСИФІКАЦІЇ СТАДІЙ ХВОРОБИ АЛЬЦГЕЙМЕРА [1] Кафедри систем штучного інтелекту / Інститут комп'ютерних наук та інформаційних технологій, / Національний університет «Львівська політехніка», Львів, Україна [2] Кафедра метематики/ Університет сільського господарства Гуго Коллотая, Краків, Польща [3] Кафедри систем віртуальної реальності / Інститут комп'ютерних наук та інформаційних технологій, / Національний університет «Львівська політехніка», Львів, Україна [4] Кафедри систем автоматизованого проектування / Інститут комп'ютерних наук та інформаційних технологій, / Національний університет «Львівська політехніка», Львів, Україна	45
3.	CYBERSECURITY AND INFORMATION PROTECTION	
3.1	Poliakov A.[1,2] FEATURES OF TOKEN-BASED AUTHENTICATION AND AUTHORIZATION IN A WEB-ORIENTED MICROSERVICE ARCHITECTURE [1] Department of Information System, Educational and Scientific Institute of Information Technologies, Simon Kuznets Kharkiv National University of Economics [2] Department of Applied Mathematics, Kharkiv National University of Radio Electronics	59
3.2	Руда Х.[1], Кос І.[1] МУЛЬТИМОДАЛЬНІ БІОМЕТРИЧНІ СИСТЕМИ [1] кафедра захисту інформації, Інститут комп'ютерних технологій, автоматики та метрології, Національний Університет "Львівська політехніка", Україна	126
3.2.1	ВСТУП	126

3.2.2	ТЕОРЕТИЧНІ ЗАСАДИ МУЛЬТИМОДАЛЬНОЇ БІОМЕТРІЇ	127
3.2.2.1	ОСНОВНІ ПОНЯТТЯ ТА ВИЗНАЧЕННЯ	127
3.2.2.2	ПРИЧИНИ ПЕРЕХОДУ ДО МУЛЬТИМОДАЛЬНОСТІ	129
3.2.3	АРХІТЕКТУРА МУЛЬТИМОДАЛЬНИХ БІОМЕТРИЧНИХ СИСТЕМ	130
3.2.3.1	ЗАГАЛЬНА СТРУКТУРА СИСТЕМИ	130
3.2.3.2	РІВНІ ЗЛИТТЯ ДАНИХ	132
3.2.3.3	МЕТОДИ ЗЛИТТЯ	133
3.2.4	ОСНОВНІ КОМБІНАЦІЇ БІОМЕТРИЧНИХ МОДАЛЬНОСТЕЙ	134
3.2.4.1	КОМБІНАЦІЇ НА ОСНОВІ ЗОБРАЖЕНЬ	134
3.2.4.2	КОМБІНАЦІЇ З ПОВЕДІНКОВИМИ ОЗНАКАМИ	135
3.2.4.3	СИСТЕМИ З ТРЬОМА І БІЛЬШЕ МОДАЛЬНОСТЯМИ	137
3.2.5	МЕТОДИ ОЦІНЮВАННЯ ЯКОСТІ ТА ПРОДУКТИВНОСТІ	138
3.2.5.1	МЕТРИКИ ОЦІНЮВАННЯ	138
3.2.6	ВРАЗЛИВОСТІ ТА МЕТОДИ ЗАХИСТУ	140
3.2.6.1	ТИПИ АТАК	140
3.2.6.2	МЕХАНІЗМИ ЗАХИСТУ	141
3.2.7	ПРАКТИЧНІ ЗАСТОСУВАННЯ	143
3.2.8	ТЕНДЕНЦІЇ ТА ВІДКРИТІ ПРОБЛЕМИ	144
3.2.9	ВИСНОВКИ	145
4.	INFORMATION SYSTEMS AND TECHNOLOGIES	
4.1	Basyuk T.[1] CURRENT APPROACHES TO MULTIMODAL DATA STORAGE IN AI-DRIVEN INFORMATION SYSTEMS [1] Department of Information Systems and Networks, Lviv Polytechnic National University, Ukraine	147
4.2	Simbirski G.[1] ASSESSMENT OF CRITICAL COMPONENTS OF AUTONOMOUS SYSTEMS USING VARIABLE PARAMETRIC IDENTIFICATION [1] Department of Software Engineering, Faculty of Software Engineering and Business, Kharkiv National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine	163
4.3	Гузь Г.[1] ДЕКЛАРАТИВНІ МЕТОДИ ПОБУДОВИ АДАПТИВНИХ КОРИСТУВАЦЬКИХ ІНТЕРФЕЙСІВ [1] кафедра комп'ютерно-інтегрованих технологій та робототехніки, Український державний університет науки і технологій, ННІ «Український державний хіміко-технологічний університет», м. Дніпро, Україна	199

4.4	Косогов О.М.[1] МОДЕЛІ І ТЕХНОЛОГІЇ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ БЕЗПЕКИ АВІАЦІЙНИХ СОЦІОТЕХНІЧНИХ КОМПЛЕКСІВ [1] Кафедра авіваційного траспорту аерокосмічного факультету, Державний університет "Київський авіаційний інститут", м. Київ, Україна	235
4.4.1	СИСТЕМНИЙ АНАЛІЗ АВІАЦІЙНИХ СОЦІОТЕХНІЧНИХ КОМПЛЕКСІВ ЯК ОБ'ЄКТІВ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ	236
4.4.1.1	АВІАЦІЙНА СОЦІОТЕХНІЧНА СИСТЕМА ЯК ОБ'ЄКТ ДЕСТРУКТИВНОГО ІНФОРМАЦІЙНОГО ВПЛИВУ	236
4.4.1.2	ПРОБЛЕМА АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ЦІЛЕСПРЯМОВАНИХ ІНФОРМАЦІЙНИХ АТАК НА ІНФОРМАЦІЙНІ РЕСУРСИ АСТК І ШЛЯХИ ЇЇ ВИРІШЕННЯ	242
4.4.2	МАТЕМАТИЧНЕ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ ПРОЦЕСІВ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ В АВІАЦІЙНИХ СОЦІОТЕХНІЧНИХ СИСТЕМАХ	249
4.4.2.1	КОНЦЕПТУАЛЬНА МОДЕЛЬ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ КОМПЛЕКСНОЇ БЕЗПЕКИ АВІАЦІЙНИХ СОЦІОТЕХНІЧНИХ СИСТЕМ	249
4.4.2.2	МАТЕМАТИЧНА МОДЕЛЬ АНАЛІЗУ УРАЗЛИВОСТЕЙ СОЦІОТЕХНІЧНИХ СИСТЕМ ДО ВПЛИВІВ СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ НА ОСНОВІ НЕЧІТКИХ НАПРАВЛЕНИХ СОЦІАЛЬНИХ ГРАФІВ	256
4.4.2.3	МАТЕМАТИЧНІ МОДЕЛІ ОЦІНЮВАННЯ ПОТУЖНОСТІ ДЕСТРУКТИВНОГО ІНФОРМАЦІЙНОГО ВПЛИВУ В АСТК	261
4.4.2.3.1	ПЕРША МАТЕМАТИЧНА МОДЕЛЬ (ОЦІНКА ПОТЕНЦІАЛУ ТА ПРОНИКНОСТІ ДІВ)	261
4.4.2.3.2	ДРУГА МАТЕМАТИЧНА МОДЕЛЬ (ОЦІНКА ЕМЕРДЖЕНТНИХ НАСЛІДКІВ ТА ОПЕРАЦІЙНОГО РИЗИКУ)	263
4.4.2.4	МАТЕМАТИЧНА МОДЕЛЬ ОЦІНЮВАННЯ НЕЛІНІЙНОЇ СТІЙКОСТІ СОЦІОТЕХНІЧНОГО КОНТУРУ АСТК	264
4.4.2.5	ЕНТРОПІЙНА МОДЕЛЬ ОЦІНЮВАННЯ СТАНУ БЕЗПЕКИ ІНФОРМАЦІЙНОГО СЕРЕДОВИЩА АВІАПІДПРИЄМСТВА	267
4.4.3	АВТОМАТИЗОВАНІ ТЕХНОЛОГІЇ ТА АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ЛЮДИНО-ЦЕНТРИЧНОГО МОНІТОРИНГУ БЕЗПЕКИ АСТК	275
4.4.3.1	ІНФОРМАЦІЙНО-ТЕХНОЛОГІЧНА АРХІТЕКТУРА АВТОМАТИЗОВАНОЇ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ	275

4.4.3.2.1	АЛГОРИТМ ДИНАМІЧНОЇ ПОБУДОВИ ТА АНАЛІЗУ НЕЧІККОГО СОЦІОТЕХНІЧНОГО ГРАФА	277
4.4.3.2.2	АЛГОРИТМ РОЗРАХУНКУ ОПЕРАЦІЙНОГО РИЗИКУ ТА АКТИВАЦІЇ ТЕХНОЛОГІЧНИХ ПРОЦЕДУР ПРОТИДІЇ	278
4.4.3.3	ТЕХНОЛОГІЯ ПІДТРИМКИ ПРИЙНЯТТЯ УПРАВЛІНСЬКИХ РІШЕНЬ ТА ОЦІНКА ЇЇ ЕФЕКТИВНОСТІ	279
	REFERENCES	285

SECTION 1. COMPUTER ENGINEERING

DOI: 10.46299/ISG.2026.MONO.TECH.2.1.1

1.1 Кіберфізичні маніпулятори. Розробка 3D моделей Niryo One та 7-DOF Manipulator Coppeliasim

У сучасну епоху технологічних інновацій розробка кіберфізичних систем, таких як маніпулятори, набуває все більшої важливості. Кіберфізичні маніпулятори відіграють вирішальну роль у таких сферах, як промисловість, медицина, автоматизація виробництва та наукові дослідження. Вони відкривають нові можливості для виконання складних завдань, підвищують точність і швидкість операцій, знижують ризики для людей і забезпечують високу ефективність роботи.

Нині існує широкий асортимент маніпуляторів, що різняться за своїм призначенням, конструкцією та функціональними можливостями. Розробка роботизованого маніпулятора вимагає детального розуміння технічних аспектів, таких як системи управління, сенсори та програмування алгоритмів руху. Для успішної реалізації таких систем необхідно забезпечити точність керування, високу якість взаємодії з оточенням і надійність роботи в різних умовах. Метою цієї роботи є розробка роботизованого маніпулятора, що включає аналіз та проектування технічних характеристик, систем управління та інтеграцію сучасних технологій. Актуальність цієї теми обумовлена необхідністю вдосконалення роботизованих систем і підвищення їх ефективності в умовах сучасної автоматизації

Розробка кіберфізичного маніпулятора є комплексним науково-технічним завданням, що об'єднує різні області знань, включаючи механіку, автоматизацію та системи управління. Проектування та розробки роботизованих систем - це аналіз і синтез механічних систем, розробку нових методів і алгоритмів керування маніпуляторами та їх впровадження у практичні умови [2].

1.1.1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО КІБЕРФІЗИЧНІ МАНІПУЛЯТОРИ

Кіберфізичні маніпулятори — це автоматизовані системи, які поєднують фізичні елементи, такі як механічні структури та рухливі частини, з цифровими компонентами, зокрема сенсорами, системами управління та програмним забезпеченням. Вони призначені для виконання складних завдань, зокрема для переміщення об'єктів, точного виконання операцій і взаємодії з навколишнім середовищем. Маніпулятори використовуються в різних галузях, таких як промисловість, медицина, наукові дослідження, автоматизація виробництва та логістика.

Модель використання роботизованих маніпуляторів — це набір методів і технологій, що реалізують певні функції або задачі в залежності від галузі застосування. Кожна модель визначає специфічні алгоритми та їхні властивості, які підтримують конкретні функції маніпулятора:

- Складальні роботи. Ця модель дозволяє маніпуляторам виконувати автоматизовані складальні процеси на виробничих лініях. Вона включає точне позиціонування елементів, з'єднання деталей і контроль якості складання.

- Логістика та складування. Використовуючи цю модель, маніпулятори автоматизують процеси переміщення вантажів на складах. Вони здатні виконувати завантаження, розвантаження та переміщення товарів між різними зонами складу, що значно підвищує ефективність логістичних операцій.

- Дослідження та аналіз. Ця модель забезпечує автоматизацію наукових експериментів, де маніпулятори виконують точні маніпуляції зі зразками. Вони можуть автоматично проводити аналізи, збирати дані та виконувати повторювані дії, що сприяє підвищенню точності досліджень.

- Медичні процедури. У цій моделі маніпулятори виконують хірургічні та медичні завдання, забезпечуючи точність та контроль під час операцій. Вони можуть використовуватися для транспортування інструментів та матеріалів під час хірургічних процедур.

- Сервісні послуги. Маніпулятори можуть використовуватися в обслуговуванні, наприклад, у готелях або ресторанах, де вони виконують завдання доставки їжі або предметів. Ця модель оптимізує процес обслуговування клієнтів та покращує загальну ефективність бізнесу.

Моделі використання роботизованих маніпуляторів демонструють їхню універсальність і здатність адаптуватися до різних потреб промисловості та служать основою для подальшого розвитку та впровадження нових технологій.

Ці системи складаються з механічних елементів, які можуть виконувати різноманітні рухи, такі як захоплення, транспортування та маніпулювання об'єктами. Вони оснащені інтелектуальними системами управління, які планують та контролюють їхні рухи, а також алгоритмами для оптимізації виконання завдань, забезпечуючи точність і швидкість роботи. Сенсори, які вбудовані в маніпулятори, дозволяють їм взаємодіяти з навколишнім середовищем, зчитуючи інформацію про об'єкти, що дозволяє адаптувати рухи в реальному часі.

Інтерфейси для користувачів дають змогу оператору керувати маніпулятором або налаштовувати його для виконання конкретних завдань. Такі маніпулятори мають значну перевагу, оскільки можуть виконувати складні й небезпечні завдання, що раніше вимагали людської участі, наприклад, працювати в умовах, де є ризик для здоров'я людини. Завдяки високій точності та можливості адаптації, вони значно підвищують ефективність і безпеку виробничих процесів. Удосконалення кіберфізичних маніпуляторів за допомогою новітніх технологій, таких як штучний інтелект і машинне навчання, дає можливість розширити їхні можливості й застосування в різних сферах.

Концепція та основні положення технології роботизованих маніпуляторів.

Кіберфізичні маніпулятори — це сучасні механічні системи, призначені для автоматизації різноманітних завдань у таких галузях, як промисловість, медицина та наукові дослідження. Вони можуть виконувати рухи з високою точністю та маніпулювати об'єктами різних форм і ваги, що забезпечує велику гнучкість і ефективність у роботі. Для їх використання необхідна система

керування, яка дозволяє програмувати дії маніпуляторів і контролювати їх рухи в реальному часі.

Області застосування кіберфізичних маніпуляторів

Роботизовані маніпулятори широко використовуються в різних галузях промисловості та науки, завдяки своїй гнучкості, точності та здатності виконувати складні завдання. Основні області їх застосування включають:

1. Промислове виробництво. Маніпулятори використовуються для автоматизації процесів складання, упаковки та обробки товарів. Вони можуть виконувати рутинні завдання, такі як зварювання, фарбування, монтаж та перевірка якості, що підвищує ефективність виробництва та знижує витрати [7].

2. Медицина. Роботизовані маніпулятори активно використовуються в хірургії для виконання точних операцій. Вони можуть допомагати лікарям під час складних процедур, забезпечуючи високу точність та зменшуючи ризики для пацієнтів. Також маніпулятори застосовуються для транспортування медичних інструментів у лікарнях.

3. Логістика та складування. Маніпулятори можуть автоматизувати процеси завантаження, розвантаження та переміщення товарів на складах. Вони здатні працювати в тісних просторах і виконувати завдання з високою швидкістю, що сприяє підвищенню продуктивності логістичних операцій.

4. Наукові дослідження. У лабораторіях роботизовані маніпулятори використовуються для автоматизації експериментів, збору даних та обробки зразків. Вони можуть виконувати рутинні завдання з високою точністю, що дозволяє вченим зосередитися на більш складних аспектах дослідження.

Завдяки своїй універсальності, кіберфізичні маніпулятори продовжують знаходити нові сфери застосування, адаптуючись до потреб сучасного суспільства [3].

Сучасні маніпулятори оснащені різними приводами — електричними, гідравлічними або пневматичними — та сенсорними системами, які забезпечують зворотний зв'язок для точного контролю. Це дозволяє маніпуляторам працювати в складних умовах, виконувати завдання з монтажу,

зварювання або збору компонентів, а також використовуватися в медичних операціях, де потрібна особлива точність.

Концепція маніпуляторів полягає в автоматизованому виконанні завдань, що імітують рухи людини або спеціалізовані операції, які складно або небезпечно виконувати вручну. Завдяки розвитку технологій управління та сенсорних систем, маніпулятори можуть інтегруватися з різними платформами керування, що дозволяє застосовувати їх у широкому спектрі різних задач [1].

Для забезпечення безпечної роботи кіберфізичних маніпуляторів застосовуються три основні рівні захисту:

Перший рівень — мінімальний рівень безпеки, де передача базових команд та даних здійснюється без шифрування. Це дозволяє швидко виконувати прості операції, але підходить лише для відкритих систем, де захист даних не є критично важливим.

Другий рівень — захист на рівні пристрою, коли кожен маніпулятор або контролер має унікальний ідентифікатор. Це обмежує доступ до системи тільки авторизованим пристроям, що допомагає запобігти несанкціонованому втручанню в роботу маніпулятора.

Третій рівень — захист на рівні сеансу керування, де для передачі команд та даних використовуються методи шифрування, наприклад, за допомогою 128-бітних ключів, що генеруються при встановленні з'єднання між контролером та маніпулятором. Цей підхід забезпечує високий рівень безпеки, запобігаючи перехопленню або підміні даних під час виконання операцій. Принципи побудови і функціонування роботизованих маніпуляторів

Кожен кіберфізичний маніпулятор складається з механічних елементів, системи управління та програмного забезпечення. Це програмне забезпечення включає в себе контролер руху, інтерфейс для управління і моніторингу, а також алгоритми, які виконують необхідні функції. Взаємодія між різними компонентами маніпулятора забезпечується через інтерфейси, такі як USB, UART або CAN.

Роботизований маніпулятор здатний виконувати як прості, так і складні завдання, взаємодіючи з сенсорами та іншими пристроями. Комунікація з контролером здійснюється через швидкісні інтерфейси, які забезпечують передачу даних у реальному часі. Контролер руху обробляє команди, отримані від користувача, передає їх на виконавчі механізми маніпулятора і отримує зворотні дані від сенсорів.

Ключовим принципом роботи роботизованих маніпуляторів є використання зворотного зв'язку для моніторингу їхнього руху та позиціонування. Це досягається за допомогою датчиків, які відстежують положення кінцевих ефекторів і передають інформацію до контролера для корекції траєкторії руху. Крім того, системи управління можуть використовувати алгоритми машинного навчання для оптимізації роботи маніпулятора в складних умовах, що підвищує його ефективність і здатність до адаптації .

Області застосування кіберфізичних маніпуляторів.

Роботизовані маніпулятори знаходять широке застосування в різних галузях промисловості та науки завдяки своїй високій гнучкості, точності та здатності виконувати складні завдання. Основні сфери їх використання включають: виконують завантаження, розвантаження та транспортування товарів між різними зонами складу, що значно оптимізує логістичні операції та знижує час на обробку вантажів, зварювання, фарбування, монтаж і перевірка якості, що значно підвищує ефективність виробництва та зменшує витрати.

У медицині роботизовані маніпулятори застосовуються в хірургії для виконання точних операцій. Вони допомагають лікарям при складних процедурах, забезпечуючи високу точність і мінімізуючи ризики для пацієнтів. Крім того, маніпулятори використовуються для транспортування медичних інструментів у лікарнях, що підвищує ефективність роботи медичного персоналу.

В логістиці та складуванні маніпулятори автоматизують процеси завантаження, розвантаження та переміщення товарів на складах. Завдяки

здатності працювати в обмежених просторах і виконувати завдання з високою швидкістю, вони сприяють збільшенню продуктивності логістичних операцій.

У наукових дослідженнях роботизовані маніпулятори використовуються для автоматизації експериментів, збору даних та обробки зразків. Вони виконують рутинні операції з високою точністю, що дозволяє вченим зосереджуватися на більш складних аспектах досліджень.

Завдяки своїй універсальності кіберфізичні маніпулятори продовжують знаходити нові сфери застосування, адаптуючись до вимог сучасного суспільства [1,2].

Моделі використання кіберфізичних маніпуляторів

Модель використання роботизованих маніпуляторів представляє собою набір методів і технологій, що реалізують конкретні функції або завдання в залежності від сфери застосування. Кожна модель визначає специфічні алгоритми та властивості, які дозволяють маніпуляторам ефективно виконувати свої функції в різних умовах.

Модель для складальних робіт дозволяє маніпуляторам автоматизувати складальні процеси на виробничих лініях. Вона включає точне позиціонування компонентів, з'єднання деталей та контроль якості складання, що забезпечує високу точність і ефективність у процесах виробництва.

Модель логістики та складування орієнтована на автоматизацію переміщення вантажів на складах. Маніпулятори в межах цієї моделі

Сучасні маніпулятори зазвичай обладнані різними приводами (електричними, гідравлічними або пневматичними) і сенсорними системами, які забезпечують зворотний зв'язок для точного керування. Це дозволяє маніпуляторам працювати в складних середовищах, виконувати завдання зі збору, монтажу або зварювання, а також застосовуватися в медичних операціях, де потрібна особлива точність. Вдосконалення технологій керування та сенсорики забезпечує сумісність маніпуляторів з різними системами управління, що дозволяє використовувати їх у широкому спектрі застосувань [6].

Модель досліджень та аналізу орієнтована на автоматизацію наукових експериментів. Маніпулятори в цій сфері виконують точні маніпуляції зі зразками, автоматично проводять аналізи, збирають дані та виконують повторювані операції, що підвищує точність і ефективність наукових досліджень.

У медичних процедурах маніпулятори використовуються для виконання хірургічних і медичних завдань. Вони забезпечують високу точність і контроль під час операцій, а також можуть бути задіяні для транспортування інструментів та матеріалів в ході медичних процедур, покращуючи результативність і знижуючи ризики для пацієнтів.

Модель сервісних послуг дозволяє застосовувати маніпулятори в обслуговуванні, таких як в готелях чи ресторанах, де вони виконують завдання доставки їжі чи предметів. Ця модель сприяє оптимізації обслуговування клієнтів і підвищує ефективність бізнес-процесів.

Ці моделі демонструють високу універсальність роботизованих маніпуляторів, здатних адаптуватися до різноманітних потреб у різних галузях, і слугують основою для подальшого розвитку та впровадження нових технологій у виробництво і обслуговування.

Гуманоїдні маніпулятори

Гуманоїдні маніпулятори мають форму, що імітує людську руку, включаючи кисть і пальці. Вони розроблені для виконання завдань, які потребують точності та маневреності, подібно до людини. Такі маніпулятори часто використовуються в дослідженнях, автоматизації виробництв та для взаємодії з людьми у різних сферах [4].

Паралельні маніпулятори

Ці маніпулятори мають конструкцію, де рухомі кінцівки з'єднані з основою через паралельні ланцюги. Вони забезпечують високу жорсткість, стабільність і точність, що робить їх ідеальними для застосувань у медицині та промисловості, де потрібні точні маніпуляції.

Серійні маніпулятори мають послідовно з'єднані ланки, що дозволяє їм здійснювати широкий діапазон рухів. Вони часто використовуються в автоматизованих виробничих лініях для виконання таких завдань, як збирання, пакування та зварювання.

Мобільні маніпулятори. Цей тип маніпуляторів оснащений колесами або гусеницями, що дозволяє їм переміщатися по площині. Мобільні маніпулятори використовуються в середовищах, де потрібно забезпечити гнучкість та автономність, наприклад, в складських приміщеннях або на будівельних майданчиках.

Роботизовані маніпулятори Ці маніпулятори можуть виконувати завдання автономно, завдяки вбудованим датчикам і програмному забезпеченню. Вони знаходять застосування в медичних процедурах, де потрібно точне виконання операцій, а також у виробництві, для автоматизації процесів [5,6].

Система управління (мікроконтролери). Для обробки команд і керування сервоприводами використовуються мікроконтролери або програмовані логічні інтегральні схеми (ПЛІС). Вони також координують роботу датчиків і забезпечують зворотний зв'язок для коригування рухів маніпулятора.

1.1.2 РОБОТ Niryo One

Робот Niryo One є високоточним 6-DOF (Degrees of Freedom) роботизованим маніпулятором, що оснащений численними механічними та електронними компонентами для ефективного виконання різноманітних завдань. Основні елементи конструкції Niryo One включають:

Сервоприводи Niryo One відповідають за рух і точність маніпулятора. Вони дозволяють точно контролювати положення та кут повороту кожної з ланок маніпулятора, забезпечуючи високу маневреність і точність. Для цього маніпулятора використовуються безщіткові сервоприводи, що характеризуються високою ефективністю та тривалим терміном служби.

Ланки маніпулятора.. Ланки маніпулятора Niryo One виконані з легких, але

міцних матеріалів, таких як алюміній, що забезпечують стабільність конструкції при високій маневреності. Кожна ланка з'єднана з попередньою через суглоби, що дозволяють маніпулятору рухатися в різних напрямках. Завдяки шести ланкам, Niryo One здатен виконувати складні маніпуляції, маючи високу точність і адаптивність.

Зовнішній вигляд маніпулятора представлений на рисунку 1

Рисунок 1 – Зовнішній вигляд маніпулятора [12].

Системи управління. Для управління маніпуляторором Niryo Oe використовуються мікроконтролери, відповідають за обробку команд та управління сервоприводами. Вони інтегрують датчики, які забезпечую ть зворотний зв'язок і дозволяють точно конролювати рухи маніпулятора в реальному час

Niryo One оснащений різноманітними датчиками, такими як датчики положення, сили та натягу, а також візуальними датчиками. Вони забезпечують маніпулятору зворотний зв'язок про виконання завдань і дозволяють адаптувати його дії відповідно до змін в навколишньому середовищі.

1. Основна платформа.

Основна платформа Niryo One підтримує стабільність і надійність всієї

системи. Вона може бути закріплена на стаціонарному місці або оснащена можливістю переміщення, що дає змогу маніпулятору адаптуватися до різних робочих середовищ.

Ці компоненти разом утворюють ефективну і гнучку систему, що дозволяє Niryo One виконувати широкий спектр завдань в автоматизації, навчанні, дослідженнях і виробництві, забезпечуючи високу точність, надійність і адаптивність до різних умов.

Загальна інформація про Niryo One маніпулятор

Niryo One — це 6-DOF (Degrees of Freedom) роботизований маніпулятор, який розроблений для використання в навчанні, дослідженнях та автоматизації. Цей робот поєднує в собі доступність, універсальність і високу точність, що робить його ідеальним для різних застосувань, від освітніх проектів до складних промислових завдань.

Маніпулятор Niryo One оснащений шістьма ступенями свободи, що дозволяє йому здійснювати широкий спектр рухів у тривимірному просторі, що важливо для виконання точних маніпуляцій з об'єктами.

Однією з основних особливостей Niryo One є його гнучкість у програмуванні та інтеграції з іншими системами. Робот підтримує різноманітні інтерфейси для підкчення та управління, включаючи USB, Ethernet, а також можливість програмування за допомогою Python або через простий інтерфейс Niryo Studio, що дозволяє користувачам створювати власні алгоритми і завдання. Він також підтримує інтеграцію з різноманітними датчиками та камерами для зворотного зв'язку і автоматизації процесів.

Завдяки своїй конструкції та використанню високоякісних сервоприводів, Niryo One може виконувати завдання, що вимагають високої маневреності та точності, такі як складання, транспортування, вимірювання та багато інших.

При цьому, можливості підключення до різних сенсорів, камер та інших пристроїв, Niryo One може бути використаний для виконання завдань у таких сферах, як дослідження, навчання, прототипування та навіть у промисловості для автоматизації виробничих процесів. Робот є доступним для широкого кола

користувачів, від студентів та дослідників до професіоналів, що дозволяє йому виконувати як прості, так і складні завдання з високою ефективністю.

Необхідні елементи для створення Niryo One маніпулятора

Для створення маніпулятора Niryo One необхідно зібрати різноманітні механічні, електронні та програмні компоненти, що забезпечують його функціональність, точність і ефективність у виконанні завдань. Ось основні елементи, які використовуються для створення цього роботизованого маніпулятора:

Сервоприводи (двигуни). Безщіткові сервоприводи є ключовими компонентами для забезпечення руху маніпулятора. Вони відповідають за точне позиціонування кожної ланки маніпулятора, дозволяючи досягти високої маневреності та точності у виконанні завдань [5,6].

Датчики. Датчики є важливими для отримання зворотного зв'язку і точного управління маніпулятором. Це можуть бути датчики положення, сили, натягу або візуальні датчики, що дозволяють контролювати рухи маніпулятора та коригувати їх відповідно до змін у навколишньому середовищі.

Основна платформа Платформа маніпулятора забезпечує його стабільність і підтримку, надаючи можливість закріплення маніпулятора на робочому місці або забезпечуючи мобільність, залежно від потреби. Вона може бути стаціонарною або мати мобільну конструкцію.

Інтерфейси для підключення та комунікації

Niryo One використовує інтерфейси для з'єднання з іншими пристроями та системами, такі як USB, Ethernet, Wi-Fi або Bluetooth, для керування та передачі даних. Це дозволяє здійснювати віддалене управління та інтеграцію з іншими роботизованими системами чи пристроями.

Програмне забезпечення. Для програмування маніпулятора Niryo One використовуються такі платформи, як Python або спеціалізоване програмне забезпечення Niryo Studio. Вони дозволяють користувачам розробляти алгоритми для управління маніпулятором, налаштовувати його роботу та виконувати складні маніпуляції.

Електроживлення та акумулятори Для забезпечення безперебійної роботи маніпулятора необхідно мати джерело живлення, яке може включати акумулятори або мережеве живлення, в залежності від конфігурації робота.

Візуальні компоненти (камери) Маніпулятори можуть бути оснащені камерами або іншими візуальними датчиками для автоматичного розпізнавання об'єктів, позиціонування або надання зворотного зв'язку про середовище, що забезпечує точніші операції.

Ці елементи в комплексі забезпечують стабільну і точну роботу маніпулятора Niryo One, дозволяючи йому виконувати різноманітні завдання в автоматизації, навчанні та інших сферах. Механічне з'єднання ланок та приводів.

- Почати зі збирання механічної конструкції маніпулятора, з'єднуючи ланки за допомогою приводів (електродвигунів, гідравлічних або пневматичних приводів) і шарнірних з'єднань.

- Забезпечити надійне кріплення ланок, щоб уникнути люфтів або нестабільності під час роботи маніпулятора. Використовувати відповідні кріпильні елементи, такі як гвинти, гайки та підшипники.

- Приводи повинні бути встановлені так, щоб вони могли вільно переміщати ланки у всіх необхідних ступенях свободи [9].

2. Підключення сенсорів.

- Встановити датчики кутових енкодерів на кожному приводі для визначення положення маніпулятора в кожному шарнірі.

- Під'єднати додаткові датчики, такі як датчики зусилля, моменту або кінцеві вимикачі, до відповідних точок на маніпуляторі.

- Перевірити надійність з'єднання кабелів сенсорів та їхню роботу, щоб уникнути помилок у вимірюваннях.

3. Інтеграція контролера з приводами та сенсорами.

- Підключити виходи контролера до керуючих входів приводів. Це дозволить керувати швидкістю, напрямком та положенням кожного приводу.

- Вхідні сигнали з сенсорів підключити до відповідних входів контролера для отримання зворотного зв'язку про положення, силу та інші параметри.

- Налаштувати протоколи зв'язку між контролером та приводами для забезпечення синхронного управління всіма ступенями свободи.

4. Підключення джерела живлення.

- Забезпечити необхідне живлення для приводів, сенсорів і контролера. Для електричних приводів це можуть бути акумулятори або блоки живлення, які мають достатню потужність для роботи всіх компонентів.

- Переконатися, що всі силові кабелі правильно підключені, а напруга відповідає технічним характеристикам компонентів.

- Встановити вимикачі або реле для безпечного увімкнення та вимкнення живлення.

5. Програмна конфігурація контролера.

- Завантажити програмне забезпечення в контролер, яке забезпечує управління маніпулятором.

- Налаштувати алгоритми управління, які враховують дані з сенсорів, координують рухи приводів і запобігають перевищенню допустимих меж переміщення.

- Перевірити коректність роботи всіх з'єднань та компонентів, виконуючи тестові рухи маніпулятора.

Дотримання цих етапів підключення всіх частин дозволяє забезпечити безперебійну роботу маніпулятора та його точне виконання заданих

Ланки маніпулятора. Ланки виготовляються з легких, але міцних матеріалів, таких як алюміній або композити, що забезпечує зручну вагу і стабільність конструкції. Кожна ланка з'єднана з попередньою за допомогою суглобів, що дозволяє маніпулятору рухатися в кількох напрямках.

Загальні відомості маніпулятора Niryo One.

Niryo One — це роботизований маніпулятор з 6 ступенями свободи (6-DOF), розроблений компанією Niryo для використання в освітніх цілях, дослідженнях та автоматизації. Цей маніпулятор поєднує в собі високу точність, універсальність і доступність, що робить його ідеальним інструментом для

навчання робототехніці, програмуванню, а також для виконання різноманітних завдань у промисловості та наукових дослідженнях.

Niryo One оснащений сучасними сервоприводами для точного управління рухами, що дозволяє маніпулятору виконувати складні маніпуляції, включаючи монтаж, транспортування, збірку, пакування та багато іншого. Маніпулятор має можливість програмування через простий інтерфейс Niryo Studio або Python, що дозволяє користувачам створювати алгоритми для виконання конкретних завдань.

Основні характеристики Niryo One включають:

1. 6 ступенів свободи (6-DOF), що дає змогу маніпулятору виконувати широкий спектр рухів у тривимірному просторі.
2. Легка конструкція з міцних матеріалів, що дозволяє знизити загальну вагу робота, одночасно зберігаючи його міцність.
3. Високоточні серво-приводи, які забезпечують точне та надійне управління кожним із 6 рухів маніпулятора.
4. Програмування через Python або Niryo Studio, що дає змогу налаштувати робота під різні завдання.
5. Гнучкість завдяки можливості підключення до різних датчиків і камер для забезпечення зворотного зв'язку в реальному часі.
6. Легкий у налаштуванні та інтеграції в різні середовища для автоматизації, прототипування та наукових досліджень.

Niryo One — це універсальний інструмент для освітніх установ, лабораторій, стартапів і навіть для малого та середнього бізнесу, який хоче інтегрувати робототехніку та автоматизацію у свої процеси. Завдяки своїй простоті використання і адаптивності цей маніпулятор дозволяє швидко освоювати основи робототехніки, програмування та машинного навчання.

Опис Niryo One маніпулятора

Niryo One— це роботизований маніпулятор, розроблений для навчання, досліджень та автоматизації. Він має 6 ступенів свободи (6-DOF), що дає йому можливість виконувати широкий спектр складних маніпуляцій, таких як

складання, транспортування, пакування, зварювання, а також різні типи вимірювань і досліджень. Маніпулятор Niryo One є зручним інструментом для тих, хто хоче вивчати робототехніку, програмування та інтеграцію технологій у різних сферах.

Основні характеристики Niryo One:

1. 6 ступенів свободи (6-DOF) A1-A6: Це дає можливість маніпулятору здійснювати точні й складні рухи в тривимірному просторі, забезпечуючи маневреність, яка необхідна для виконання різноманітних завдань

2. Механічна конструкція: Niryo One має легку, але міцну конструкцію, виготовлену з алюмінію та композитних матеріалів. Це забезпечує надійність і зменшує загальну вагу, що важливо для точності рухів.

3. Сервоприводи високої точності: Маніпулятор оснащений безщітковими сервоприводами для кожного з ступенів свободи, що дозволяє досягти високої точності та стабільності в рухах.

4. Програмування через Python або Niryo Studio: Користувачі можуть програмувати маніпулятор через популярну мову програмування Python або через спеціальне програмне забезпечення Niryo Studio. Це дає можливість створювати алгоритми для автоматизації завдань та інтегрувати маніпулятор з іншими технологіями.

5. Підключення додаткових датчиків та камер: Niryo One можна оснащувати різноманітними датчиками, включаючи камери для комп'ютерного зору та інші сенсори, що дозволяє виконувати завдання із зворотним зв'язком і забезпечувати більш складні маніпуляції.

6. Гнучкість та адаптивність: Маніпулятор можна налаштовувати для різних типів завдань, від навчальних проектів до прототипування та автоматизації виробництва (рис.2).

7. Інтерфейси зв'язку: Niryo One підтримує комунікацію через USB, Ethernet і Wi-Fi, що дозволяє зручно підключати маніпулятор до комп'ютерів та інших пристроїв.

Характеристики Niryo One маніпулятора зведені в таблицю 1.

Застосування:

Niryo One використовується для навчання робототехніці, наукових досліджень, а також у промисловості та автоматизації. Завдяки своїй універсальності, маніпулятор ідеально підходить для лабораторій, освітніх установ, стартапів та підприємств, що працюють у сфері автоматизації процесів, створення прототипів або тестування нових технологій.

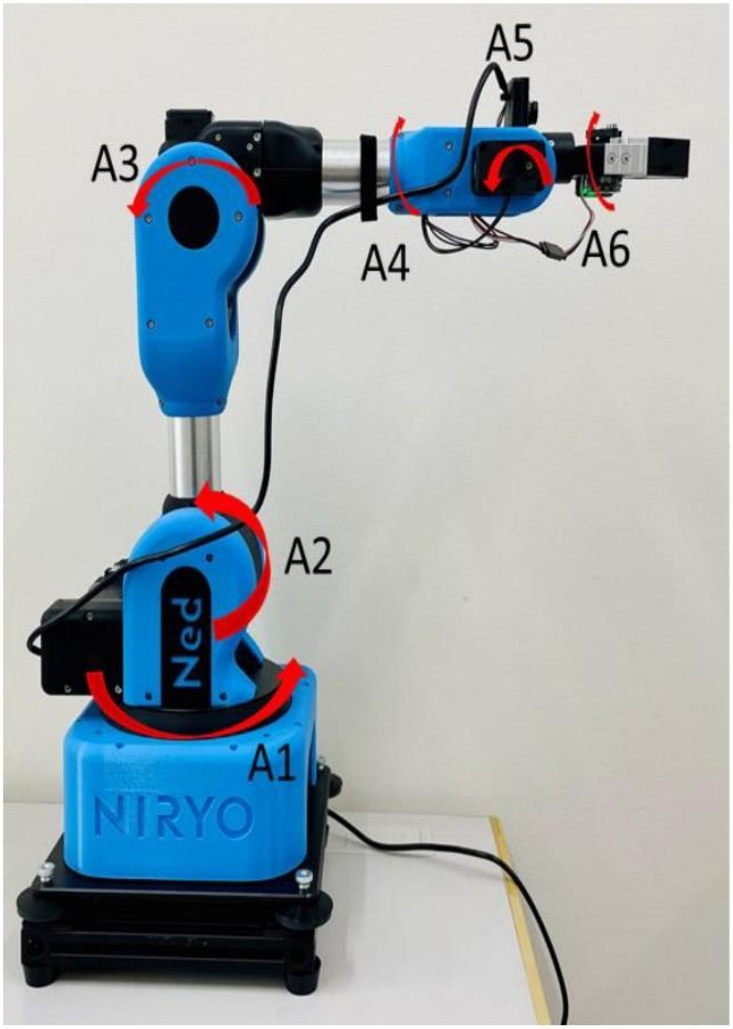


Рисунок 2 - Опис Niryo One маніпулятора [12].

Таблиця 1– Характеристики Niryo One маніпулятора

Параметр	Значення
Кількість ступенів свободи	6
Максимальна робоча область	0.6 м (радіус)
Точнійсть	±0.5 мм
Максимальне навантаження	4.2 кг
Тип приводу	Сервоприводи
Діапазон кутів повороту	±180°
Швидкість руху	До 0.4 м/с
Система керування	Linux, Windows, MacOS
Датчики	Підтримка камери, датчиків для комп'ютерного зору
Вживана енергія	12 В, 2 А

Живлення

Niryo One — це робот-маніпулятор, який зазвичай використовує стандартне живлення через адаптер, підключений до електричної мережі. Ось кілька деталей щодо живлення цього робота:

Напруга: Niryo One зазвичай працює від джерела живлення з напругою 12 В. Це стандартне живлення для багатьох роботизованих систем, яке забезпечує достатню потужність для роботи сервомоторів та інших компонентів робота.Адаптер живлення: У комплекті з роботом може бути адаптер для підключення до стандартної електричної розетки, який забезпечує стабільне живлення.

Споживана потужність: Споживана потужність Niryo One може змінюватися в залежності від завдання, яке виконує робот. Під час інтенсивних маніпуляцій з

важкими об'єктами чи швидких рухів може споживатися більше енергії.

Батареї (опція): Зазвичай робота живиться від адаптера, однак можна також використовувати спеціальні акумулятори для автономної роботи в деяких моделях або модифікаціях.

Загалом, Niryo One — це досить енергоефективний робот, що працює від стандартного джерела живлення 12 В для забезпечення стабільної роботи.

Входи та виходи. Робот Niryo One має різноманітні входи та виходи для взаємодії з навколишнім середовищем і підключення до інших пристроїв. Основні типи входів і виходів, які доступні у Niryo One: Виходи (Output) Входи (Input):

Сервомотори: Niryo One використовує серво-рухомі механізми для маніпуляцій із об'єктами. Кожен з цих сервомоторів можна контролювати через відповідні виходи на платі контролера. Виходи на платі відповідають за передачу сигналів для точного керування рухами руки робота.

Підключення до джерела живлення: Основний вихід — це підключення до адаптера живлення для подачі постійного струму (12 В). Це забезпечує енергетичну потужність для роботи всіх компонентів.

Комунікаційні порти.

USB-порти: Використовуються для підключення робота до комп'ютера для налаштування або програмування.

Ethernet-порт. Для підключення робота до локальної мережі для моніторингу та управління через віддалений доступ. Датчики положення (Інкрементальні датчики): Вхідні сигнали з датчиків позиції використовуються для точного визначення місця розташування кожного сервомотора робота.

Датчики сили: Це датчики, що вимірюють сили, застосовані до маніпулятора, особливо важливі для задач, що вимагають високої точності або м'якого взаємодії з об'єктами.

Камера або зображення: За допомогою підключених камер (наприклад, за допомогою RPi камери або інших сенсорів) робот може здійснювати візуальне сприйняття, отримувати зображення та використовувати їх для визначення

об'єктів чи позиціонування.

Роз'єми для зовнішніх датчиків або периферії:

Робот має кілька роз'ємів GPIO (General Purpose Input/Output) для підключення додаткових датчиків чи пристроїв. Ці порти можуть використовуватися для інтеграції з різноманітними зовнішніми сенсорами, такими як датчики температури, вологості чи інші інтерфейси для роботизованих систем.

Інтерфейс для програмування:

Використовуються стандартні входи через USB або Ethernet для підключення до комп'ютера, налаштування та керування роботом за допомогою програмного забезпечення, яке можна налаштувати через API або через графічний інтерфейс.

Загальні характеристики входів/виходів Niryo One:

Порти GPIO: Мають розширювальні можливості для додавання нових сенсорів або активації зовнішніх пристроїв.

Інтерфейс USB/Serial: Для підключення до ПК для керування або програмування.

Ethernet/Wi-Fi: Для з'єднання через локальну мережу або для віддаленого доступу.

Завдяки таким входам і виходам Niryo One має високу гнучкість для інтеграції з іншими пристроями та можливість виконання складних завдань, таких як маніпулювання об'єктами, обробка сенсорних даних і навіть робота в складних умовах [7].

Зв'язок, Зв'язок у роботі Niryo One організовано за допомогою кількох ключових технологій та інтерфейсів для забезпечення комунікації між роботом, комп'ютером або іншими пристроями. Ось основні способи зв'язку Niryo One:

1. Ethernet (Провідний зв'язок):

Ethernet порт на Niryo One дозволяє підключити робота до локальної мережі (LAN). Це дає можливість здійснювати моніторинг, програмування та керування роботом через мережу.

Таке підключення дозволяє організувати віддалений доступ до робота для налаштування і управління, а також інтегрувати Niryo One в більші автоматизовані системи.

2. Wi-Fi (Безпровідний зв'язок):

Wi-Fi є зручним варіантом для зв'язку з роботом без необхідності підключення через кабель. За допомогою Wi-Fi робота можна підключити до домашньої чи корпоративної мережі.

Це дозволяє використовувати програмне забезпечення для керування роботом через веб-інтерфейс або спеціальні API для віддаленого керування [8].

3. USB (Для програмування та налаштування): USB-порт на Niryo One дозволяє підключити робота безпосередньо до комп'ютера. Це зручно для налаштування та програмування робота за допомогою спеціального програмного забезпечення або через командний рядок.

За допомогою цього інтерфейсу можна безпосередньо передавати інструкції або отримувати дані від робота.

4. Інтерфейс для програмування (API):

API для програмування дає змогу користувачам розробляти свої власні програми для управління роботом. Через ці API можна писати скрипти для автоматизації завдань, підключати різні датчики чи інші пристрої.

API підтримує Python, що є популярною мовою для робототехніки, та дозволяє взаємодіяти з роботом через мережу, USB або інші інтерфейси.

5. Сенсори і камери (для взаємодії із середовищем):

Niryo One може бути оснащений сенсорами та камерами, що дозволяють роботу здійснювати візуальний зв'язок із зовнішнім світом. За допомогою цих сенсорів робот може «бачити» об'єкти, а також отримувати дані для визначення своїх рухів і маніпуляцій.

Камера або сенсори, такі як RGB камера або LiDAR, можуть бути підключені для розпізнавання об'єктів та взаємодії з ними через відповідні інтерфейси.

6. Додаткові пристрої через GPIO:

GPIO порти (General Purpose Input/Output) на Niryo One можуть використовуватися для підключення додаткових сенсорів, пристроїв чи активації зовнішніх елементів. Ці порти також можуть використовуватися для безпровідного або проводового зв'язку з іншими мікроконтролерами чи датчиками.

7. Мережеві протоколи та сервіси:

Niryo One підтримує різні мережеві протоколи, такі як TCP/IP, для зручного підключення до зовнішніх сервісів, що дозволяє організовувати зв'язок з іншими пристроями чи програмами для більш складних автоматизованих систем.

8. Інтерфейс для віддаленого керування:

Для користувачів доступні різні інтерфейси для віддаленого керування роботом, наприклад, через веб-браузер або спеціальні мобільні додатки, що підключаються до робота по Wi-Fi або Ethernet [9].

Це дозволяє виконувати завдання без необхідності фізичної присутності поруч з роботом, а також надавати доступ до його налаштувань через інтернет.

9. Зв'язок з іншими роботами:

За допомогою спеціальних протоколів та програмного забезпечення можна налаштувати зв'язок між кількома роботами Niryo One для виконання спільних завдань, таких як складання, транспортування чи інших координованих дій.

Технічні характеристики Niryo one маніпулятора

1. Висота: приблизно 40 см.

2. Довжина робочого простору: до 40 см, залежно від положення рук.

3. Вага: близько 3.5 кг.

4. Матеріали: конструкція виготовлена з алюмінію та пластику, що забезпечує легкість і міцність.

5. Розміри основи: база робота має квадратну форму, з розмірами приблизно 30 см на 30 см.

Розроблюється 3D модель Niryo one manipulator в застосунку CoppeliaSim (рис.3)

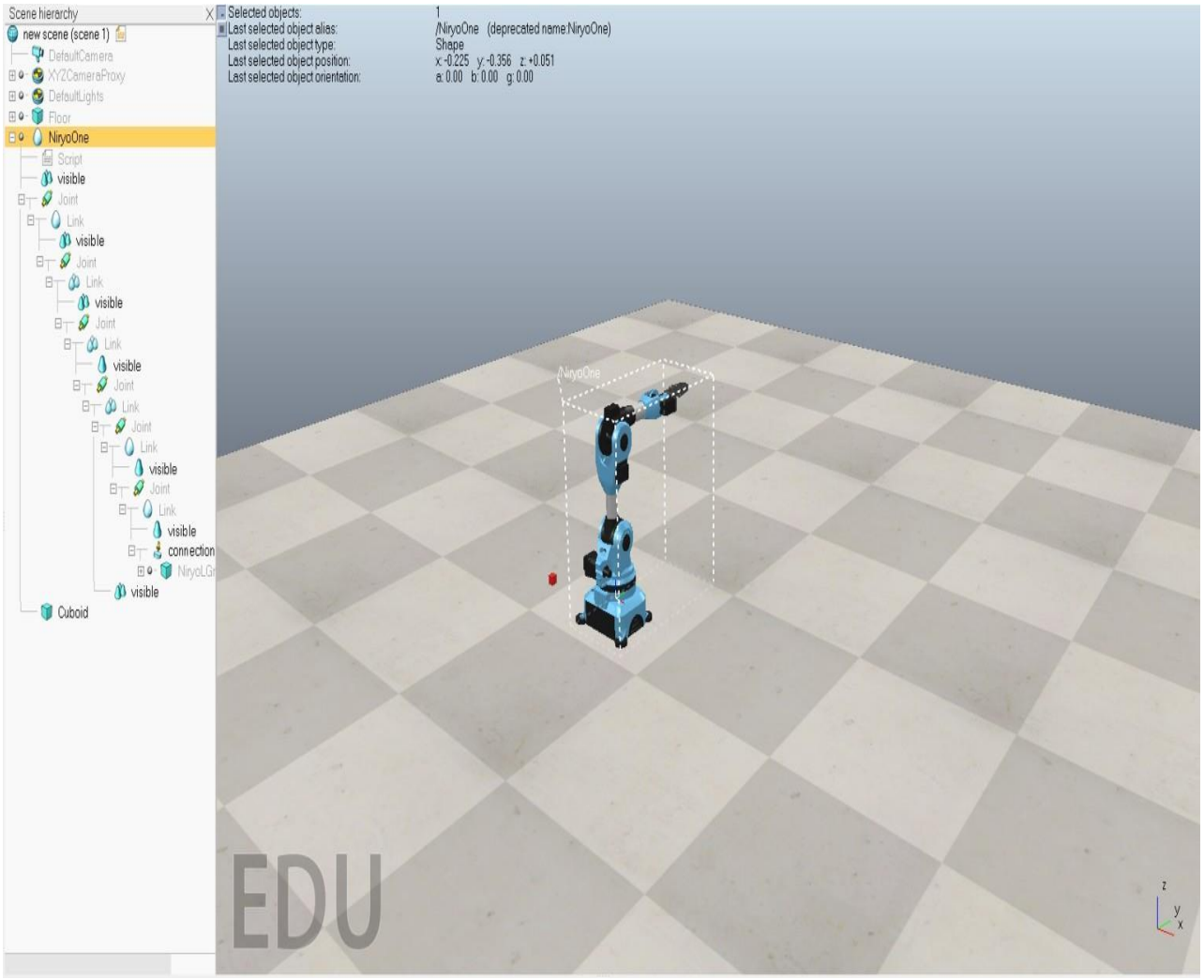


Рисунок 3 – 3D модель маніпулятора у програмі CoppeliaSim (різні режими роботи)
Джерело рисунка: скрін екрана (розробка автора)

Підключення компонентів Niryo one маніпулятора

Для правильного складання та інтеграції компонентів робота Niryo One необхідно виконати наступні кроки щодо підключення ланок, приводів, сенсорів, контролера та джерела живлення.

1. Механічне з'єднання ланок та приводів:

Почати зі збирання механічної конструкції робота, з'єднуючи маніпуляторні ланки за допомогою приводів (електродвигунів) і шарнірних з'єднань.

Забезпечити надійне кріплення ланок, щоб уникнути люфтів або нестабільності під час роботи робота. Для цього слід використовувати відповідні

кріпильні елементи, такі як гвинти, гайки та підшипники.

Приводи повинні бути правильно встановлені для забезпечення вільного руху ланок у всіх необхідних напрямках.

2. Підключення сенсорів: Встановити датчики положення (інкрементальні енкодери) на кожному приводі для визначення точного положення маніпулятора на кожному шарнірі.

Підключити додаткові сенсори, такі як датчики сили, кінцеві вимикачі або інші, для моніторингу зусиль і перевірки меж руху маніпулятора.

Перевірити надійність підключень кабелів сенсорів і їхню коректну роботу, щоб уникнути помилок в даних.

3. Інтеграція контролера з приводами та сенсорами:

Підключити виходи контролера до керуючих входів приводів для регулювання швидкості, напрямку і положення кожного приводу.

Під'єднати вхідні сигнали від сенсорів до відповідних входів контролера для отримання зворотного зв'язку про положення маніпулятора, зусилля або інші важливі параметри.

Налаштувати протоколи зв'язку між контролером і приводами, щоб забезпечити синхронізацію рухів та точне управління всіма ступенями свободи робота.

4. Підключення джерела живлення:

Забезпечити необхідне живлення для приводів, сенсорів і контролера, підключивши адаптер живлення або акумулятор, що відповідає вимогам по напрузі та потужності.

Переконатися, що всі силові кабелі підключені правильно, а напруга відповідає технічним характеристикам компонентів робота.

Встановити вимикачі або реле для безпечного включення і вимкнення живлення робота.

5. Програмна конфігурація контролера:

Завантажити програмне забезпечення на контролер робота, яке забезпечує керування маніпулятором.

Налаштувати алгоритми управління, що обробляють дані з сенсорів і координують рухи приводів, а також попереджають про можливі перевищення допустимих меж [10,11].

Перевірити роботу всіх компонентів, виконуючи тестові рухи робота для перевірки точності виконання завдань.

Дотримання цих етапів підключення та налаштування дозволить забезпечити надійну та точну роботу робота Niryo One, а також гарантувати правильне виконання заданих дій [10].

Знаючи які функції повинна виконувати дана розробка була написана програма. Фрагмент коду:

```
function sysCall_init() sim = require('sim')
proximity = sim.getObject('../Proximity_sensor');
 conveyorPath = sim.getObject('../conveyor');
 proximity1 = sim.getObject('../Proximity_sensor1');
conveyorPath1 = sim.getObject('../conveyor1');
end

function sysCall_actuation() end
function sysCall_sensing()

r, d, p, h, n=sim.readProximitySensor(proximity)
--sim.writeCustomTableData(conveyorPath1, '_____ctrl__',
{vel = 0})
 if(r > 0) then
sim.writeCustomTableData(conveyorPath, '_____ctrl__',
{vel = 0}) sim.writeCustomTableData(conveyorPath1, '___ctrl_____',
{vel = 0.1})
end

r1, d1, p1, h1, n1=sim.readProximitySensor(proximity1) if(r1 > 0)
then
sim.writeCustomTableData(conveyorPath1, '_____ctrl__',
```

```
{vel = 0}) sim.writeCustomTableData(conveyorPath, '____ctrl___',
{vel = 0.1})
end


end

function sysCall_cleanup() end
sim=require'sim'

-- This is a threaded script, and is just an example!

function moveToConfig(handles,maxVel,maxAccel,maxJerk,
targetConf) local params = {
joints = handles, targetPos = targetConf, maxVel = maxVel,
maxAccel = maxAccel, maxJerk = maxJerk,
}
sim.moveToConfig(params)
 end

function sysCall_thread() sim.wait(40)
local jointHandles={} for i=1,6,1 do
jointHandles[i]=sim.getObject('../Joint',{index=i-1}) end
local connection=sim.getObject('../connection') local
gripper=sim.getObjectChild(connection,0) local gripperName="NiryoNoGripper"
if gripper~=-1 then gripperName=sim.getObjectAlias(gripper,4)
end
-- Set-up some of the RML vectors:
local vel=20 local accel=40 local jerk=80 local
maxVel={vel*math.pi/180,vel*math.pi/180,vel*math.pi/180,vel*math.pi/
180,vel*math.pi/180,vel*math.pi/180}
local
maxAccel={accel*math.pi/180,accel*math.pi/180,accel*math.pi/180,accel*math.
pi/180,accel*math.pi/180,accel*math.pi/180}
local maxJerk={jerk*math.pi/180,jerk*math.pi/180,jerk*math.pi/180,jerk*math.pi/
180,jerk*math.pi/180,jerk*math.pi/180}

local targetPos1={-90*math.pi/180,-15*math.pi/180,0*math.pi/180,0*math.pi/
180,-38*math.pi/180,-90*math.pi/180}
moveToConfig(jointHandles,maxVel,maxAccel,maxJerk,targetPos1)
sim.setInt32Signal(gripperName..'_close',1)
```

```
    sim.wait(4)
    local targetPos3={0,0,0,0,0,0}
moveToConfig(jointHandles,maxVel,maxAccel,maxJerk,targetPos3)
    local targetPos2={90*math.pi/180,-15*math.pi/180,0*math.pi/180,0*math.pi/
180,-36*math.pi/180,-90*math.pi/180}
    moveToConfig(jointHandles,maxVel,maxAccel,maxJerk,targetPos2)
sim.clearInt32Signal(gripperName..'_close')
    sim.wait(4) moveToConfig(jointHandles,maxVel,maxAccel,maxJerk,targetPos3)
     end
```

3D модель і код маніпулятора в програмі CoppeliaSim представлено на рис. 4.

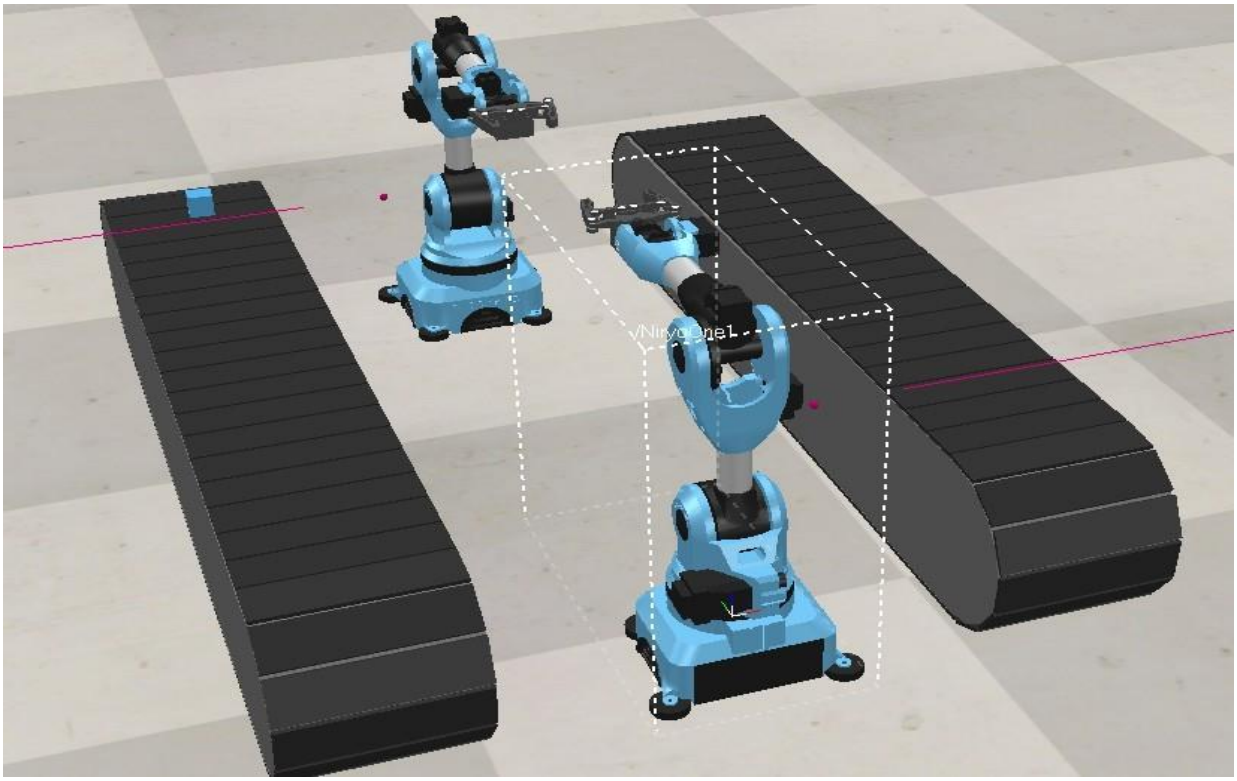


Рисунок 4 - 3D модель і код маніпулятора в програмі CoppeliaSim

Джерело рисунка: скрін екрана (розробка автора)

Скетч. У процесі розробки була розроблена 3D модель для демонстрації роботи системи рисунок 5

Simulation script "/Script"

```
function sysCall_init()
    sim = require('sim')
    proximity = sim.getObject('../Proximity_sensor');
    conveyorPath = sim.getObject('../conveyor');
end

function sysCall_actuation()

end

function sysCall_sensing()

    r, d, p, h, n=sim.readProximitySensor(proximity)
    if(r > 0) then
        print(conveyorPath)
        sim.writeCustomTableData(conveyorPath, '__ctrl__', {vel = 0})
        --sim.writeCustomTableData(conveyorPath,'__ctrl__',{pos=0.9})
    end
end

function sysCall_cleanup()

end
```

Рисунок 5 – Фрагмент коду програми - демонстрації роботи системи.

Джерело рисунка: скрін екрана (розробка автора)

1.1.3 7-DOF МАНІПУЛЯТОР

7-DOF маніпулятори, або роботизовані руки, мають семи ступенів свободи, що надає їм можливість виконувати складні маніпуляції в тривимірному просторі. Ці маніпулятори часто використовуються в різних галузях, таких як промисловість, медицина, дослідження та автоматизація.

Конструкція 7-DOF маніпулятора

1. Сервоприводи

Кожен ступінь свободи маніпулятора зазвичай реалізується за допомогою сервоприводів, які забезпечують точне управління рухами. Сервоприводи можуть бути електричними або пневматичними, залежно від вимог до маніпулятора (рисунок 6).

2. Кінематичний ланцюг

7-DOF маніпулятори складаються з декількох ланок, з'єднаних між собою за допомогою суглобів. Кінематичний ланцюг визначає, як кожен рух впливає на позицію та орієнтацію маніпулятора в просторі (рисунок 7).

3. Робоча область

Залежно від конструкції, 7-DOF маніпулятори мають різні робочі області, які визначають максимальні відстані і кути рухів. Це важливо для виконання завдань у певних умовах.

4. Датчики

Для забезпечення точності і контролю рухів маніпулятори часто оснащуються різноманітними датчиками, такими як датчики положення, сили та моменту. Вони дозволяють отримувати інформацію про стан маніпулятора та коригувати його рухи.

5. Контрольна система

Управління 7-DOF маніпулятором здійснюється за допомогою мікроконтролерів або ПК, що дозволяє реалізувати різні алгоритми для планування рухів, корекції траєкторій та взаємодії з навколишнім середовищем. Характеристики 7-DOF маніпулятора зведено в таблицю 2

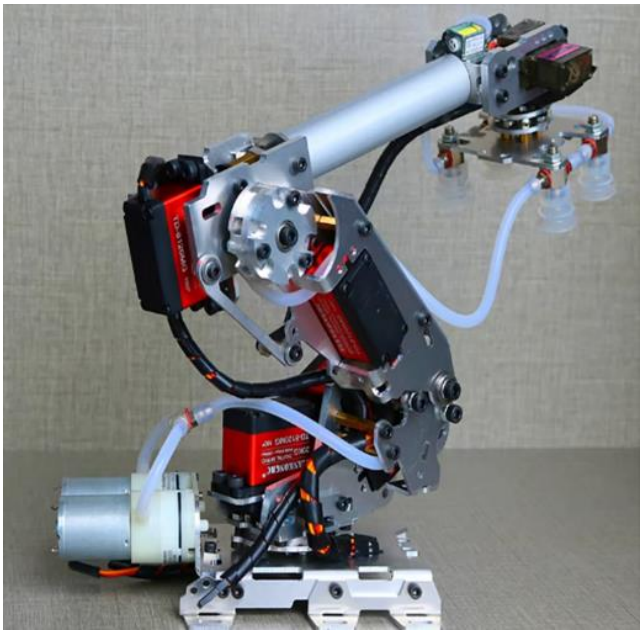


Рисунок 6 - 7 dof manipulator [12].

Застосування 7-DOF маніпулятора

7-DOF маніпулятори знайшли широке застосування в таких сферах:

- Промисловість: Застосовуються для збирання, зварювання, пакування та контролю якості.

- Медицина: Використовуються в хірургії, де необхідна висока точність.

Дослідження: Застосовуються в експериментах з робототехніки, автоматизації та штучного інтелекту.

-Арт: В деяких випадках 7-DOF маніпулятори використовуються для створення мистецьких проектів.

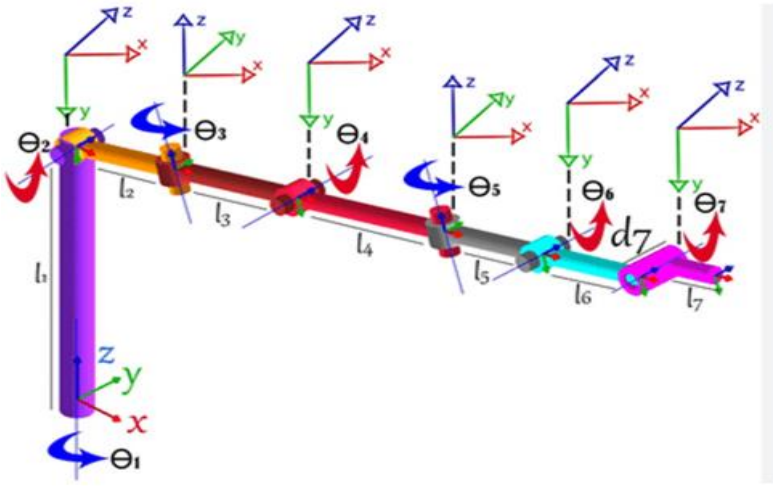


Рисунок 7 - Опис 7-DOF маніпулятора [12].

Таблиця 2 – Характеристики 7-DOF маніпулятора

Параметр	Значення
Кількість ступенів свободи	7
Максимальна робоча область	1.5 м (висота), 1.2 м (ширина)
Максимальне навантаження	5 кг
Тип приводу	Сервоприводи

Продовження таблиці 2

Діапазон кутів повороту	±180°
Швидкість руху	до 1 м/с
Точність позиціонування	±0.1 мм
Система керування	Мікроконтролер (наприклад, Arduino)
Датчики	Датчики позиції, сили, моменту
Вживана енергія	24 В, 2 А

Живлення. Живлення описує електричні параметри та вимоги для роботи 7-DOF маніпулятора. Основні аспекти живлення маніпулятора включають:

- Джерело живлення: Маніпулятор зазвичай живиться від зовнішнього блоку живлення з вихідною напругою 24 В. Це забезпечує достатню потужність для роботи всіх сервоприводів, датчиків та електроніки.

- Споживаний струм: При номінальному навантаженні маніпулятор може споживати до 2 А. У пікових режимах, наприклад, при різких рухах або утриманні важкого вантажу, споживання струму може збільшуватися.

- Регулювання напруги: Для стабільної роботи мікроконтролера та інших електронних компонентів використовуються регулятори напруги, які знижують 24 В до рівня, необхідного для живлення окремих компонентів (наприклад, 5 В або 3,3 В).

- Акумуляторне живлення: Якщо маніпулятор планується використовувати без підключення до стаціонарного джерела живлення, можна застосувати літій-іонні акумулятори з вихідною напругою 24 В і ємністю не менше 5000 мА·год для забезпечення тривалої автономної роботи.

- Захист від перенапруги та короткого замикання: У системі живлення повинні бути встановлені відповідні захисні елементи, такі як плавкі запобіжники та стабілізатори напруги, щоб уникнути пошкодження обладнання.

Ці аспекти забезпечують надійну та безпечну роботу маніпулятора в різних режимах.

Входи та виходи. Маніпулятор оснащений декількома входами та виходами, які забезпечують комунікацію з контролером і підключення сенсорів та виконавчих механізмів. Вхідні канали використовуються для зчитування даних із сенсорів, а вихідні – для керування двигунами або іншими виконавчими пристроями.

Кожен з цифрових входів/виходів може бути налаштований як вхідний або вихідний за допомогою відповідних команд. Максимальна напруга на цих лініях становить 5В, а допустимий струм для одного виходу не перевищує 40 мА. Окрім базових функцій, деякі входи/виходи мають спеціалізовані можливості:

- ШІМ (PWM): Декілька каналів маніпулятора підтримують широтно-імпульсну модуляцію, яка дозволяє плавно регулювати швидкість обертання двигунів або яскравість світлодіодів.

- Послідовний інтерфейс (Serial): Канали для обміну даними між маніпулятором та контролером, що забезпечують зв'язок з іншими модулями або центральним контролером.

- Аналогові входи: Деякі канали можуть вимірювати аналогові сигнали, наприклад, від датчиків кута нахилу або струму.

Для більшої гнучкості та інтеграції до платформи маніпулятору додатково передбачені спеціальні функції, такі як обробка зовнішніх переривань для миттєвого реагування на зміну зовнішніх умов.

Зв'язок. Маніпулятор може здійснювати зв’язок із зовнішніми пристроями через різні протоколи інтерфейсів, що забезпечують передачу даних і команд для управління. Найпоширеніші варіанти включають використання послідовного інтерфейсу (UART), інтерфейсу I2C, і SPI.

- Послідовний інтерфейс (UART): Застосовується для прямого обміну даними між маніпулятором і контролером або іншими периферійними пристроями. Він дозволяє передавати дані за допомогою приймально-передавальних каналів, що підтримують послідовну передачу інформації.

- Інтерфейс I2C (TWI): Дозволяє підключати декілька пристроїв у мережу, де кожен компонент може виступати як ведучий або ведений. Це дає змогу інтегрувати маніпулятор у системи, які вимагають координації кількох датчиків та акторів.

- SPI (Serial Peripheral Interface): Підходить для високошвидкісної передачі даних між маніпулятором і контролером. Використовується в ситуаціях, де потрібно забезпечити швидкий обмін великими обсягами інформації, такими як контроль стану або синхронізація з іншими модулями.

Ці інтерфейси дозволяють організовувати інтеграцію маніпулятора в комплексні кібер-фізичні системи, забезпечуючи обмін сигналами в реальному часі та керування пристроєм на основі зворотного зв'язку [8].

Розроблюється 3D модель 7 DoF manipulator в застосунку CoppeliaSim (рис.8)

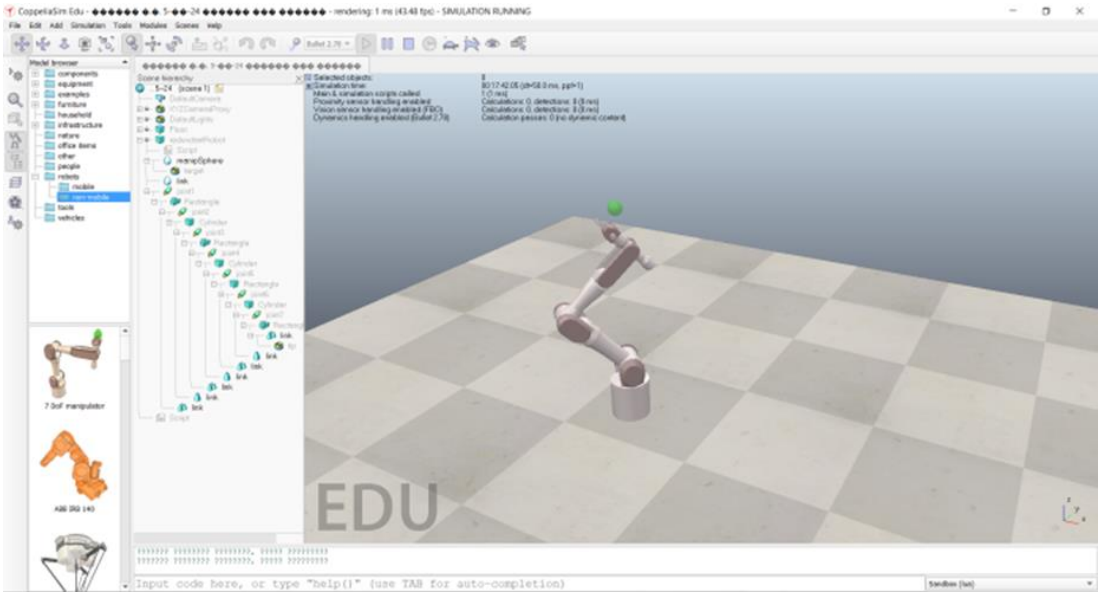


Рисунок 8 – 3D модель маніпулятора у програмі CoppeliaSim (різні режими роботи)

Джерело рисунка: скрін екрана (розробка автора)

Технічні характеристики 7 DoF маніпулятора

1. Кількість ступенів свободи: 7 (усі обертальні з'єднання).

2. Діапазон руху для кожного суглоба

- Зазвичай кожен суглоб може обертатися в межах від -180° до +180° або більше, залежно від конструкції.

- Деякі суглоби можуть мати обмежений діапазон руху (наприклад, ±90°) через механічні особливості.

3. Довжина ланок

- Залежить від конкретного маніпулятора. Наприклад, загальна довжина може варіюватися від 0.5 до 1.5 метрів для настільних моделей або від 1 до 3 метрів для промислових.

4. Вантажопідйомність

- Від 0.5 кг для легких маніпуляторів до 20 кг і більше для промислових моделей.

- Зменшується при збільшенні відстані до кінцевого ефектора.

5. Точність позиціонування*

- Може бути від 0.1 мм до 0.5 мм або менше, залежно від використання датчиків і приводів.

6. Повторюваність

- Зазвичай становить від ±0.02 мм до ±0.1 мм.

7. Швидкість руху

- Максимальна лінійна швидкість може бути в межах 1-2 м/с.

- Кутова швидкість для суглобів може досягати 120-300°/с.

8. Тип приводів

- Часто використовуються серводвигуни, але можуть бути і гідравлічні або пневматичні приводи для спеціальних застосувань.

9. Робоча зона

- Сферична або циліндрична, з радіусом, що відповідає довжині повністю витягнутої руки.

10. Матеріал конструкції:

- Зазвичай використовують алюміній або композитні матеріали для легкості, або сталь для промислових важких маніпуляторів.

Підключення компонентів 7-DOF маніпулятора

Для правильного складання та інтеграції компонентів 7-DOF маніпулятора необхідно виконати наступні кроки з підключення ланок, приводів, сенсорів, контролера та джерела живлення.

1. Механічне з'єднання ланок та приводів.

- Почати зі збирання механічної конструкції маніпулятора, з'єднуючи ланки за допомогою приводів (електродвигунів, гідравлічних або пневматичних приводів) і шарнірних з'єднань.

- Забезпечити надійне кріплення ланок, щоб уникнути люфтів або нестабільності під час роботи маніпулятора. Використовувати відповідні кріпильні елементи, такі як гвинти, гайки та підшипники.

- Приводи повинні бути встановлені так, щоб вони могли вільно переміщати ланки у всіх необхідних ступенях свободи [9].

2. Підключення сенсорів.

- Встановити датчики кутових енкодерів на кожному приводі для визначення положення маніпулятора в кожному шарнірі.

- Під'єднати додаткові датчики, такі як датчики зусилля, моменту або кінцеві вимикачі, до відповідних точок на маніпуляторі.

- Перевірити надійність з'єднання кабелів сенсорів та їхню роботу, щоб уникнути помилок у вимірюваннях.

3. Інтеграція контролера з приводами та сенсорами.

- Підключити виходи контролера до керуючих входів приводів. Це дозволить керувати швидкістю, напрямком та положенням кожного приводу.

- Вхідні сигнали з сенсорів підключити до відповідних входів контролера для отримання зворотного зв'язку про положення, силу та інші параметри.

- Налаштувати протоколи зв'язку між контролером та приводами для забезпечення синхронного управління всіма ступенями свободи.

4. Підключення джерела живлення.

- Забезпечити необхідне живлення для приводів, сенсорів і контролера. Для електричних приводів це можуть бути акумулятори або блоки живлення, які мають достатню потужність для роботи всіх компонентів.

- Переконатися, що всі силові кабелі правильно підключені, а напруга відповідає технічним характеристикам компонентів.

- Встановити вимикачі або реле для безпечного увімкнення та вимкнення живлення [10].

5. Програмна конфігурація контролера.

- Завантажити програмне забезпечення в контролер, яке забезпечує управління маніпулятором.

- Налаштувати алгоритми управління, які враховують дані з сенсорів, координують рухи приводів і запобігають перевищенню допустимих меж переміщення.

- Перевірити коректність роботи всіх з'єднань та компонентів, виконуючи тестові рухи маніпулятора.

Дотримання цих етапів підключення всіх частин дозволяє забезпечити безперебійну роботу маніпулятора та його точне виконання заданих дій.

Фрагмент коду

```
function sysCall_init()
    -- ???????? ????? ???????? ?? ?????
    joint1 = sim.getObjectHandle('./redundantRobot/joint1')
    joint2 = sim.getObjectHandle('./redundantRobot/joint2')
    joint7 = sim.getObjectHandle('./redundantRobot/joint7')  -- ????????? ????? ?????????? ???????
    sphere = sim.getObjectHandle('./redundantRobot/manipSphere')
    print("??????? ?? ????? ?????????????")
end

-- ??????? ???????, ??? ??????????? ?? ??????? ?????
function sysCall_actuation()
    -- ???????? ??????? ??????? ????????
    local currentPosition1 = sim.getJointPosition(joint1)
    local currentPosition2 = sim.getJointPosition(joint2)
```

```
-- ????????? ??????????? ?? ???????? ???????
local stepSize = 0.001

-- ??? joint1
if math.abs(currentPosition1 - targetPosition1) > stepSize then
    if currentPosition1 < targetPosition1 then
        sim.setJointPosition(joint1, currentPosition1 + stepSize)
    else
        sim.setJointPosition(joint1, currentPosition1 - stepSize)
    end
else
    sim.setJointPosition(joint1, targetPosition1)
end

-- ??? joint2
if math.abs(currentPosition2 - targetPosition2) > stepSize then
    if currentPosition2 < targetPosition2 then
        sim.setJointPosition(joint2, currentPosition2 + stepSize)
    else
        sim.setJointPosition(joint2, currentPosition2 - stepSize)
    end
else
    sim.setJointPosition(joint2, targetPosition2)
end
```

ВИСНОВКИ.

Здійснено проектування роботів – маніпуляторів Niryo One, та 7-DOF з визначенням їх технічних характеристик і функціональних можливостей. Також розроблено теоретичні моделі програмного забезпечення та алгоритмів для управління роботами, зокрема інтеграція сенсорів, систем навігації та комунікаційних протоколів. Оцінка ефективності розроблених рішень у теоретичному контексті дозволила виявити як потенціал, так і обмеження їх роботи. Представлені теоретичні аспекти робототехнічних технологій, їхні переваги та недоліки. Описано конструкції роботів – маніпуляторів Niryo One, та 7-DOF, систем управління і технічних компонентів на основі теоретичних моделей. Робота включає аналіз можливих схем підключення елементів системи та програмного забезпечення, а також розробку робота. Розробка таких роботів

– маніпуляторів як Niryo One, та 7-DOF є прикладом того, як можна інтегрувати різні технології в єдину теоретичну модель, що відкриває нові можливості для використання робота в різних сферах. Наголошено на важливості теоретичного підходу в розробці інноваційних рішень, які можуть бути реалізовані в практичному контексті в майбутньому.

SECTION 2. COMPUTER SCIENCE

DOI: 10.46299/ISG.2026.MONO.TECH.2.2.1

2.1 Оцінка підходів машинного навчання для класифікації стадій хвороби альцгеймера

Хвороба Альцгеймера є поширеним нейродегенеративним захворюванням, що супроводжується поступовим погіршенням когнітивних функцій і зниженням якості життя пацієнтів. У зв’язку зі зростанням кількості випадків деменції актуальним є раннє виявлення патологічних змін, точне розмежування нормального старіння, легких когнітивних порушень і різних стадій хвороби Альцгеймера, а також розроблення інтелектуальних засобів підтримки клінічного прийняття рішень.

Діагностика хвороби Альцгеймера ґрунтується на аналізі різнорідних медичних даних, зокрема нейровізуалізаційних зображень, клінічної інформації, результатів когнітивного оцінювання, генетичних і біомаркерних показників. Їхня мультимодальність, неоднорідність і можливі пропуски ускладнюють комплексний аналіз, однак водночас створюють передумови для застосування методів машинного та глибинного навчання.

Сучасні алгоритми машинного навчання, зокрема метод опорних векторів, дерева рішень, випадковий ліс, логістична регресія та ансамблеві моделі, дають змогу автоматизувати класифікацію станів пацієнтів на основі інформативних ознак. Методи глибинного навчання, зокрема згорткові нейронні мережі, автоенкодери та трансформерні архітектури, є перспективними для аналізу медичних зображень, семантичної сегментації та виявлення прихованих закономірностей, пов’язаних із прогресуванням захворювання.

Розроблення ефективних класифікаційних моделей передбачає попередню обробку даних, нормалізацію, сегментацію, відбір ознак, балансування класів, навчання, валідацію та оцінювання якості за такими метриками, як точність, чутливість, специфічність, прецизійність, F1-міра та площа під ROC-кривою.

Попри значний прогрес, залишаються проблеми обмеженості наборів даних, неоднорідності медичної інформації, ризику перенавчання, недостатньої інтерпретованості моделей і складності їх клінічної інтеграції. Тому актуальним є дослідження сучасних підходів машинного навчання для класифікації стадій хвороби Альцгеймера та визначення їхнього потенціалу для комп'ютеризованої діагностики й персоналізованої медицини.

Постановка проблеми. Хвороба Альцгеймера є одним із найпоширеніших нейродегенеративних захворювань, що поступово призводить до погіршення когнітивних функцій, втрати здатності до самостійного виконання повсякденних дій та суттєвого зниження якості життя пацієнтів. У сучасних умовах ця проблема набуває особливої актуальності у зв'язку зі старінням населення, збільшенням тривалості життя та зростанням кількості осіб, які потребують тривалого медичного, соціального й психологічного супроводу. За даними Всесвітньої організації охорони здоров'я, кількість людей, які живуть із деменцією, до 2030 року може зрости на 40% і досягти 78 мільйонів осіб. Така динаміка свідчить про необхідність розроблення нових підходів до ранньої діагностики, моніторингу перебігу захворювання та підтримки прийняття клінічних рішень [13].

Деменція, зокрема хвороба Альцгеймера як її найпоширеніша форма, впливає на пам'ять, мислення, мовлення, орієнтацію в просторі й часі, здатність до навчання, прийняття рішень та виконання щоденних завдань. На ранніх етапах прояви захворювання можуть бути малопомітними або подібними до природних вікових змін, що ускладнює своєчасне встановлення діагнозу. Водночас саме раннє виявлення патологічних змін є критично важливим, оскільки дає змогу своєчасно призначити відповідне лікування, сповільнити прогресування симптомів, забезпечити належний догляд і підготувати пацієнта та його родину до подальших етапів перебігу захворювання.

Одним із ключових завдань сучасної медичної діагностики є точне визначення стадії хвороби Альцгеймера. Це завдання має важливе клінічне значення, оскільки різні стадії захворювання потребують відмінних підходів до

лікування, реабілітації, когнітивної підтримки, медикаментозного супроводу та організації догляду. Наприклад, на ранніх стадіях особлива увага приділяється виявленню легких когнітивних порушень, профілактиці подальшого погіршення стану та підтримці самостійності пацієнта. На пізніших стадіях зростає потреба у постійному нагляді, адаптації середовища, підтримці родини та залученні мультидисциплінарної команди фахівців. Отже, коректна класифікація стадії захворювання є необхідною умовою для вибору оптимальної стратегії медичної допомоги.

Традиційні методи діагностики хвороби Альцгеймера ґрунтуються на клінічному огляді, когнітивному тестуванні, аналізі анамнезу, лабораторних показниках і результатах нейровізуалізаційних обстежень. Проте такі підходи часто залежать від досвіду фахівця, якості зібраних даних, доступності медичного обладнання та повноти інформації про пацієнта. Крім того, медичні дані, що використовуються для діагностики, є різнорідними за своєю природою: вони можуть включати зображення мозку, числові клінічні показники, результати тестів, генетичні маркери, біомаркери та текстові медичні записи. Така мультимодальність створює додаткові труднощі для їх комплексного аналізу, але водночас відкриває можливості для застосування методів штучного інтелекту та машинного навчання.

У цьому контексті особливого значення набувають методи машинного навчання, які дають змогу автоматизувати аналіз великих обсягів медичних даних, виявляти приховані закономірності, класифікувати стани пацієнтів і підтримувати процес прийняття клінічних рішень. Використання таких методів може сприяти підвищенню точності діагностики, зменшенню ймовірності суб’єктивних помилок, покращенню відтворюваності результатів і створенню інтелектуальних систем комп’ютеризованої діагностики. Особливо перспективним є поєднання методів машинного навчання з нейровізуалізацією, семантичною сегментацією медичних зображень, аналізом біомаркерів і персоналізованих медичних даних.

Упродовж останнього десятиліття спостерігається стале зростання наукового інтересу до проблеми класифікації стадій хвороби Альцгеймера. Щороку збільшується кількість публікацій, присвячених розробленню нових алгоритмів, порівнянню моделей, аналізу медичних зображень, використанню мультимодальних даних і впровадженню систем підтримки прийняття рішень у медичну практику. Для підтвердження цієї тенденції було проаналізовано кількість публікацій, пов’язаних із цією тематикою, у наукометричній базі Scopus за період 2013–2024 років. Для пошуку було сформовано запит із використанням релевантних ключових слів і виконано фільтрацію результатів за роками публікації. У результаті пошуку було виявлено 728 документів, що свідчить про зростання академічної уваги до проблеми прогресування, діагностики та класифікації хвороби Альцгеймера [14].

Попри значний прогрес у цій сфері, низка проблем залишається відкритою. Зокрема, існують труднощі, пов’язані з неоднорідністю медичних даних, обмеженістю вибірок, дисбалансом класів, різними протоколами збору даних, недостатньою інтерпретованістю моделей машинного навчання та складністю їх інтеграції у реальні клінічні процеси. Крім того, більшість моделей демонструє високу ефективність на окремих наборах даних, однак потребує додаткової перевірки щодо узагальнення, стабільності та практичної придатності в умовах різних медичних установ.

З огляду на проведений аналіз можна стверджувати, що станом на сьогодні актуальною науково-прикладною задачею є комплексний аналіз і систематизація сучасних підходів машинного навчання, які використовуються для класифікації стадій хвороби Альцгеймера. Важливим є визначення переваг і обмежень традиційних алгоритмів машинного навчання, глибинних нейронних мереж, гібридних і мультимодальних підходів, а також оцінювання їхнього потенціалу для застосування у системах комп’ютеризованої діагностики. Особливу увагу доцільно приділити питанням точності, надійності, пояснюваності та клінічної цінності таких моделей.

Постановка задачі полягає у дослідженні сучасного стану застосування методів машинного навчання для класифікації стадій хвороби Альцгеймера, узагальненні наявних підходів, визначенні основних джерел даних, методів попередньої обробки, алгоритмів класифікації та метрик оцінювання якості. Результати такого аналізу можуть стати основою для подальшого розроблення інтелектуальних, надійних і пояснюваних систем підтримки медичного прийняття рішень, орієнтованих на ранню діагностику, персоналізований супровід і покращення якості життя пацієнтів із хворобою Альцгеймера.

Аналіз останніх досліджень і публікацій. Сучасні дослідження демонструють значний прогрес у розробленні автоматизованих підходів до класифікації стадій хвороби Альцгеймера із застосуванням методів машинного та глибинного навчання. Зокрема, авторами роботи [15] запропоновано два класифікатори - глибоку згорткову нейронну мережу (DCNN) та модель на основі архітектури VGG16 (VCNN), які були навчені на 1000 МРТ-зображеннях із бази даних ADNI, що охоплювали чотири стадії деменції. Найвищої точності класифікації 96,19% було досягнуто з використанням моделі VCNN для розмежування легкої та дуже тяжкої деменції. Водночас у дослідженні зазначено потенціал подальшого підвищення ефективності моделей шляхом оптимізації архітектури та розширення набору даних.

У роботі [16] розроблено згорткову нейронну мережу для класифікації трьох станів: відсутність деменції, помірна деменція та дуже тяжка деменція. Модель було навчено на 3720 МРТ-зображеннях із бази даних ADNI, а досягнута точність становила 97,8%. Проте в цьому дослідженні недостатньо деталізовано етапи попередньої обробки даних, а також не проведено оцінювання ефективності моделі на зовнішньому незалежному наборі даних. У дослідженні [17] запропоновано багатоплощинну архітектуру CNN, здатну одночасно аналізувати сагітальні, корональні та аксіальні зрізи МРТ-зображень головного мозку. На основі 1500 зображень із бази ADNI ця модель досягла загальної точності 93%, однак її оцінювання було обмежене одним порівняно невеликим набором даних.

У праці [18] застосовано методи керованого машинного навчання, зокрема SVM, Random Forest, Naive Bayes, Gradient Boosting та інші алгоритми, до поздовжніх МРТ-даних із бази OASIS. Найкращі результати продемонстрував класифікатор Gradient Boosting, який забезпечив точність 97,58% та значення AUC 0,98. Водночас набір даних містив лише 150 зразків, а порівняння зазначених алгоритмів із моделями глибинного навчання не проводилося, що обмежує узагальнюваність отриманих результатів.

У дослідженні [19] розглянуто як задачу класифікації стадій, так і задачу виявлення хвороби Альцгеймера із застосуванням кількох архітектур CNN. Авторами запропоновано гібридну модель на основі ResNet50 та здійснено її порівняння з такими архітектурами, як VGG16, DenseNet201, AlexNet і немодифікована ResNet50. МРТ-дані з платформи Kaggle було розподілено на чотири стадії деменції. Запропонована гібридна модель досягла найвищої точності - 90%, однак якість класифікації для помірної деменції становила лише 70%, що автори пов’язують із дисбалансом класів і недостатньою кількістю зразків для цієї стадії.

Узагальнення результатів сучасних досліджень підтверджує високий потенціал архітектур глибинного навчання для класифікації стадій хвороби Альцгеймера з високою точністю. Водночас більшість робіт має низку обмежень, серед яких невеликі обсяги наборів даних, недостатня збалансованість класів, відсутність зовнішньої валідації, а також переважна орієнтація на бінарну або трикласову класифікацію замість повноцінної багатостадійної класифікації захворювання.

Задача класифікації стадій хвороби Альцгеймера здебільшого розв’язується із застосуванням підходів машинного навчання, які ґрунтуються на навчанні за вибіркою та алгоритмічному прийнятті рішень щодо віднесення вхідних даних до певного класу. У сфері аналізу медичних зображень класифікація спрямована на автоматичне виділення релевантних ознак із діагностичних зображень та їх подальше віднесення до відповідних категорій на основі виявлених характеристик.

Одним із найпоширеніших алгоритмів у цій галузі є метод опорних векторів (SVM), принцип роботи якого полягає у побудові однієї або кількох гіперплощин, що оптимально розділяють об'єкти різних класів у просторі ознак. Іншим часто застосовуваним методом є алгоритм дерева рішень - непараметричний метод керованого навчання, який формує модель на основі простих правил прийняття рішень, отриманих із характеристик вхідних даних. Класифікатор Random Forest розширює цей підхід завдяки ансамблевому навчанню, об'єднуючи результати кількох дерев рішень, що сприяє підвищенню стійкості та точності класифікації. Подібно до цього, багатошаровий перцептрон (MLP), який є різновидом штучної нейронної мережі прямого поширення, застосовується для керованої класифікації зображень. Навчання MLP здійснюється на основі розміченого набору даних із використанням алгоритму зворотного поширення помилки [20].

Мета роботи стосується комплексного аналізу сучасних підходів машинного навчання та глибинного навчання, що застосовуються для класифікації стадій хвороби Альцгеймера на основі медичних даних, зокрема нейровізуалізаційних зображень, клінічних показників і біомаркерів, а також визначити їхні переваги, обмеження та перспективи використання в системах комп'ютеризованої діагностики й підтримки клінічного прийняття рішень.

Виклад основного матеріалу. Для перевірки ефективності розробленого ансамблю моделей було проведено класифікацію зображень на тестовій вибірці, яка не використовувалася ні на етапі навчання, ні під час валідації класифікатора. Такий підхід дає змогу оцінити здатність моделі до узагальнення, тобто її спроможність коректно класифікувати нові дані, які раніше не були представлені системі. Це є важливим етапом експериментальної перевірки, оскільки висока точність лише на навчальних або валідаційних даних не завжди свідчить про реальну ефективність моделі в умовах практичного застосування.

За результатами класифікації тестових зображень було отримано значення точності 0,9417. Цей показник є дещо нижчим порівняно з точністю, досягнутою на валідаційній вибірці, однак він залишається високим, зважаючи на те, що

тестові дані були повністю невідомими для класифікатора. Отриманий результат свідчить про достатню стійкість ансамблевої моделі та її здатність зберігати високу якість класифікації поза межами даних, використаних під час налаштування параметрів.

Зниження точності на тестовій вибірці порівняно з валідаційною може бути зумовлене природною варіативністю медичних зображень, відмінностями у візуальних ознаках між окремими класами, а також складністю розмежування суміжних стадій захворювання. У задачах класифікації стадій хвороби Альцгеймера такі відмінності можуть бути особливо незначними, оскільки перехід між когнітивною нормою, ранніми когнітивними порушеннями та вираженими проявами деменції має поступовий характер. Саме тому результат на рівні 94,17% можна вважати достатньо високим і таким, що підтверджує доцільність використання ансамблевого підходу.

Для детальнішого аналізу результатів класифікації було побудовано матрицю помилок із використанням функції ConfusionMatrixDisplay бібліотеки sklearn.metrics. Матриця помилок дає змогу не лише оцінити загальну кількість правильних і помилкових передбачень, але й визначити, між якими класами модель найчастіше допускає помилки. Це особливо важливо для медичних задач, оскільки різні типи помилкової класифікації можуть мати різну клінічну значущість.

Отримана матриця помилок, наведена на рис. 1, дає можливість візуально проаналізувати розподіл правильних і хибних класифікацій для кожної стадії захворювання. Такий аналіз є необхідним для подальшого вдосконалення моделі, зокрема оптимізації її архітектури, балансування класів, розширення навчальної вибірки та покращення етапів попередньої обробки медичних зображень. Отже, результати тестування підтверджують ефективність розробленого ансамблю моделей для класифікації зображень і демонструють його перспективність для застосування в задачах комп’ютеризованої діагностики хвороби Альцгеймера.

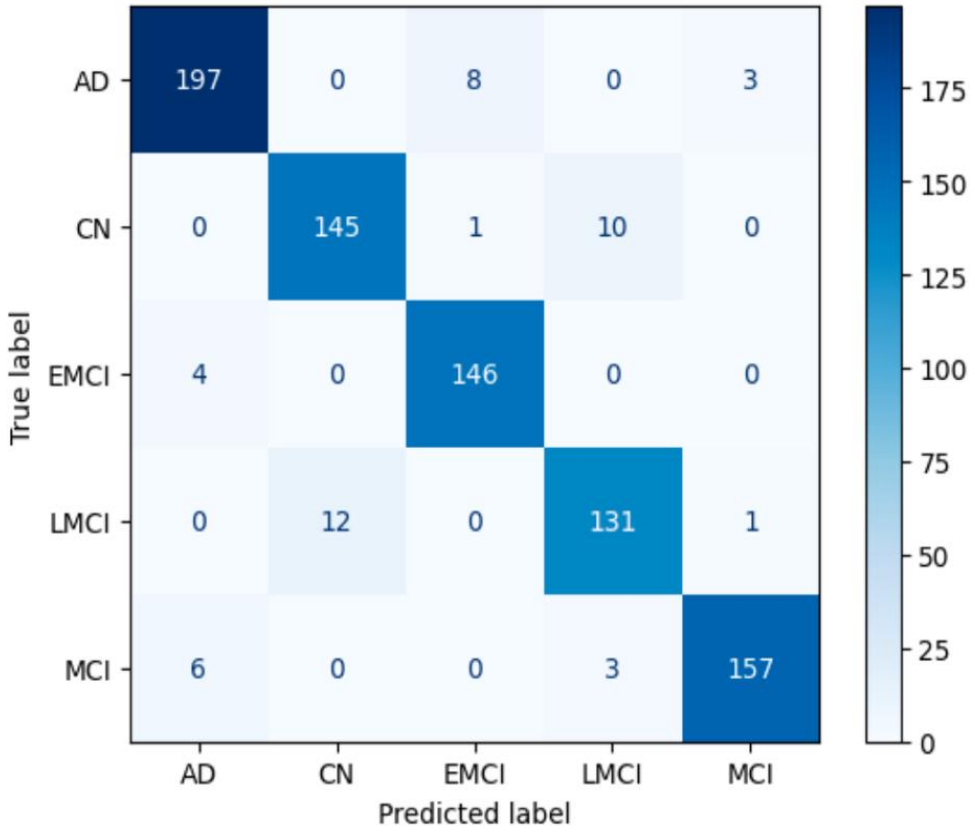


Рисунок 1. Матриця помилок ансамблю з м'яким голосуванням на тестовій вибірці.

Джерело рисунка: авторська розробка.

З урахуванням отриманої матриці помилок можна зробити висновок, що розроблений класифікатор загалом демонструє високий рівень коректності розпізнавання стадій хвороби Альцгеймера на тестовій вибірці. Водночас детальний аналіз розподілу помилкових класифікацій свідчить про те, що найбільша кількість неточностей спостерігається між класами CN та LMCI. Це може бути пояснено складністю розмежування когнітивної норми та пізнього легкого когнітивного порушення, оскільки на цих етапах зміни у структурі мозку можуть бути незначними, поступовими та частково подібними за візуальними ознаками. У медичному контексті така помилка є очікуваною, оскільки перехід від нормального когнітивного стану до легких когнітивних порушень часто має нечіткі межі та потребує врахування додаткових клінічних, когнітивних і біомаркерних показників.

Найменшу кількість помилок було зафіксовано під час класифікації стадії MCI, що свідчить про наявність більш виражених або стабільних ознак, які модель здатна ефективно ідентифікувати. Це може вказувати на те, що для цього класу сформовано достатньо інформативний набір характеристик, які дають змогу класифікатору надійно відокремлювати його від інших стадій захворювання. Загалом результати, отримані на основі матриці помилок, можна оцінити як достатньо високі, оскільки більшість зразків було правильно віднесено до відповідних класів.

Кількісна оцінка якості роботи моделі вимагає аналізу додаткових метрик класифікації, зокрема Precision, Recall та F1-score. Метрика Precision характеризує частку правильно класифікованих зразків серед усіх об’єктів, які модель віднесла до певного класу. Метрика Recall показує, яку частку зразків певного класу модель змогла правильно виявити серед усіх реальних представників цього класу. Своєю чергою, F1-score є узагальненою метрикою, що поєднує Precision і Recall та дає змогу оцінити збалансованість класифікації.

Для отримання зазначених показників було використано функцію classification_report з бібліотеки sklearn.metrics. Ця функція формує текстовий звіт з основними показниками ефективності класифікаційної моделі для кожного класу окремо, а також подає усереднені значення метрик. Такий звіт дає змогу не лише оцінити загальну якість моделі, але й виявити класи, для яких класифікація є найбільш або найменш точною. Результати такого аналізу наведено на рис. 2.

	precision	recall	f1-score	support
AD	0.95	0.95	0.95	208
CN	0.92	0.93	0.93	156
EMCI	0.94	0.97	0.96	150
LMCI	0.91	0.91	0.91	144
MCI	0.98	0.95	0.96	166
accuracy			0.94	824
macro avg	0.94	0.94	0.94	824
weighted avg	0.94	0.94	0.94	824

Рисунок 2. Звіт основних класифікаційних показників ансамблю з м'яким голосуванням.

Джерело рисунка: авторська розробка.

Результати показують, що модель демонструє належну ефективність із загальною точністю 0,94. Вона забезпечує достатньо високі значення метрик recall, precision та F1-score для більшості класів на нових, раніше невідомих даних, що свідчить про її ефективність у задачі класифікації зразків.

Перед побудовою ансамблевої моделі кожен класифікатор було окремо оцінено в контексті задачі п’ятикласової класифікації. Для аналізу їхньої ефективності застосовано стандартні метрики оцінювання, зокрема Accuracy, Precision, Recall, F1-score, а також Confusion Matrix. Кожну модель було навчено на визначеній навчальній підвибірці та перевірено на окремих валідаційних даних.

Налаштування гіперпараметрів здійснювалося із застосуванням методу Grid Search, який передбачає систематичний перебір комбінацій попередньо заданих параметрів із використанням крос-валідації. Такий підхід дає змогу визначити конфігурацію моделі, що забезпечує найвище середнє значення точності. Оптимальні параметри, отримані в результаті пошуку, було використано для навчання фінальної версії кожного класифікатора на повному наборі даних.

З метою порівняльного аналізу результати всіх чотирьох моделей було узагальнено в єдиній підсумковій таблиці 1. Проведений аналіз показав, що

найвищу ефективність продемонстрували класифікатори Random Forest, SVM та MLP, тоді як метод Decision Tree забезпечив найнижче значення точності серед досліджуваних моделей.

Таблиця 1.

Порівняльні результати тестування досліджуваних моделей

Модель	Метрика	AD	CN	EMCI	LMCI	MCI
Дерево рішень (DT)	Precision	0.87	0.85	0.90	0.79	0.95
	Recall	0.91	0.84	0.87	0.84	0.89
	F1-score	0.89	0.84	0.88	0.81	0.92
	Accuracy	0.87				
Випадковий ліс (RF)	Precision	0.96	0.93	0.94	0.88	0.98
	Recall	0.94	0.90	0.97	0.93	0.95
	F1-score	0.95	0.92	0.96	0.91	0.96
	Accuracy	0.94				
Метод опорних векторів (SVM)	Precision	0.95	0.92	0.93	0.92	0.96
	Recall	0.93	0.94	0.97	0.90	0.95
	F1-score	0.94	0.93	0.95	0.91	0.95
	Accuracy	0.94				
Багатошаровий перцептрон (MLP)	Precision	0.95	0.94	0.95	0.91	0.97
	Recall	0.96	0.93	0.97	0.92	0.95
	F1-score	0.95	0.94	0.96	0.92	0.96
	Accuracy	0.95				

Отже, з метою підвищення загальної стабільності та точності класифікації було побудовано ансамблеву модель на основі трьох найефективніших класифікаторів - SVM, RF та MLP. Для реалізації ансамблю обрано метод Voting, оскільки він характеризується обчислювальною простотою, швидким виконанням і гнучкістю застосування.

На відміну від складніших ансамблевих стратегій, таких як stacking або blending, метод голосування не потребує додаткового навчання метамоделі. Це дає змогу інтегрувати різнорідні базові класифікатори, які мають різні архітектури, параметри та принципи прийняття рішень. Завдяки цьому підхід

Voting є зручним для експериментального комбінування моделей і оптимізації їх взаємодії відповідно до специфіки конкретної класифікаційної задачі.

Використання ансамблевої моделі дає можливість зменшити вплив помилок окремих класифікаторів, підвищити надійність прогнозування та забезпечити більш стійкі результати під час класифікації стадій хвороби Альцгеймера. Архітектуру ансамблевого класифікатора, реалізованого із застосуванням VotingClassifier, наведено на рис. 3.

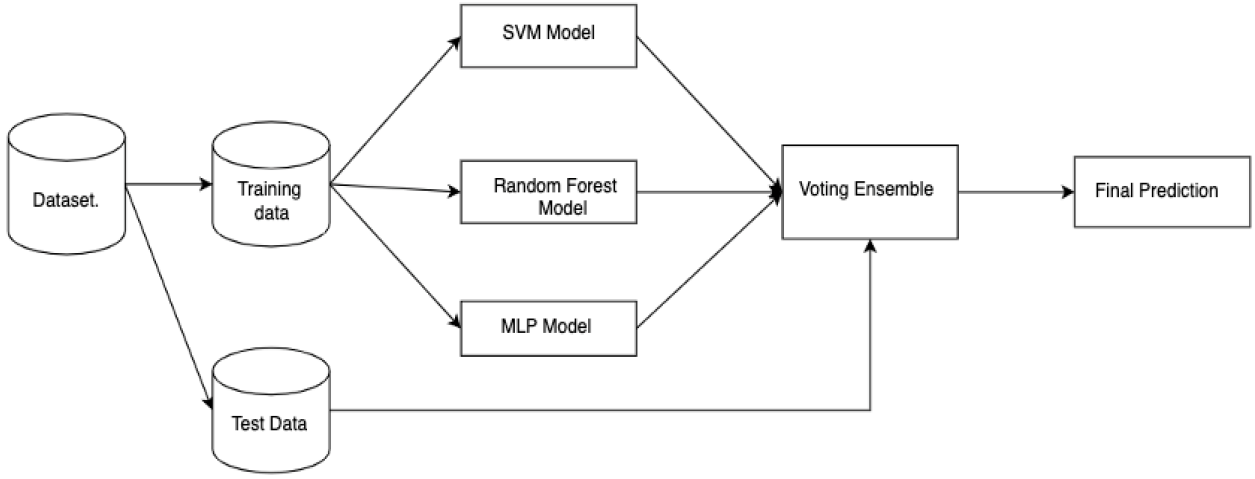


Рисунок 3. Схема створення ансамблю класифікаційних моделей.

Джерело рисунка: авторська розробка.

Проведено порівняльне оцінювання різних класифікаційних моделей, за результатами якого встановлено, що найвищу ефективність серед досліджуваних підходів продемонстрував багатошаровий перцептрон. На основі отриманих результатів для подальшого формування ансамблевої моделі було обрано три найефективніші класифікатори: RF, SVM та MLP. Їх інтеграція в ансамбль була спрямована на підвищення загальної точності класифікації та забезпечення більшої стабільності прогнозування.

Було проведено аналіз методів ансамблевого навчання, за результатами якого як найбільш доцільний підхід обрано VotingClassifier. Цей метод дає змогу поєднувати різнорідні класифікатори з відмінними конфігураціями параметрів, архітектурами та принципами прийняття рішень. Завдяки цьому підвищується

надійність прогнозування, оскільки кінцеве рішення формується не на основі одного алгоритму, а шляхом узгодження результатів кількох моделей.

У межах дослідження було проаналізовано дві стратегії ансамблевого голосування - hard voting та soft voting. Метод soft voting продемонстрував кращі результати, досягнувши загальної точності 0,96. Фінальна ансамблева модель показала високі значення метрик Precision та Recall для більшості класів, що підтверджує її ефективність у точному визначенні різних стадій хвороби Альцгеймера в межах використаного набору даних.

Висновки. У роботі досліджено задачу автоматизованої класифікації стадій хвороби Альцгеймера із застосуванням методів машинного навчання та ансамблевого моделювання. Проведене порівняння моделей Decision Tree, Random Forest, SVM та MLP показало, що найкращі результати забезпечили MLP, Random Forest і SVM, тоді як Decision Tree продемонстрував нижчу точність.

На основі трьох найефективніших класифікаторів було побудовано ансамблеву модель із використанням VotingClassifier. Порівняння стратегій hard voting і soft voting засвідчило перевагу методу soft voting, який забезпечив вищу стабільність і точність класифікації.

Експериментальні результати підтвердили ефективність запропонованого підходу: загальна точність ансамблевої моделі на тестових даних становила 0,9417, а максимальна точність досягала 0,96. Аналіз матриці помилок показав, що найбільше неточностей виникає між класами CN та LMCI, що пояснюється подібністю ознак когнітивної норми та легких когнітивних порушень.

Ансамблевий підхід є перспективним для багатокласової класифікації стадій хвороби Альцгеймера та може бути використаний як основа для систем комп’ютеризованої діагностики. Подальші дослідження доцільно спрямувати на розширення набору даних, балансування класів, зовнішню валідацію та підвищення інтерпретованості моделей.

SECTION 3. CYBERSECURITY AND INFORMATION PROTECTION

DOI: 10.46299/ISG.2026.MONO.TECH.2.3.1

3.1 Features of token-based authentication and authorization in a web-oriented microservice architecture

Introduction

In the context of the ongoing digital transformation of the economy, web-oriented information systems have become a foundational infrastructure for e-commerce, financial services, government e-services, and corporate information resources. The widespread adoption of distributed architectures, particularly microservices, along with the use of cloud and containerized platforms, increases the need for reliable authentication of both users and services while maintaining acceptable performance. Under these conditions, authentication mechanisms are no longer merely a local security component; they substantially affect scalability, fault tolerance, and the efficiency of computing resource utilization.

One of the de facto standards for token-based authentication is the JSON Web Token (JWT), which is widely used across the OAuth 2.0 and OpenID Connect (OIDC) ecosystems, as well as in proprietary solutions for securing RESTful APIs and microservice interactions. Standards and normative documents (e.g., RFC 7519, RFC 7518, RFC 8032) define the token format and the permissible signing algorithms. At the same time, in practical web-application design, the choice of a specific cryptographic algorithm and authentication scheme is often made empirically, without relying on formalized models or quantitative criteria to assess the impact of this choice on latency, throughput, and system resource consumption. As a result, designers may adopt either overly “heavy” algorithms that significantly increase latency under high load, or overly “lightweight” mechanisms that do not provide an adequate cryptographic level of protection—an issue that is particularly critical for microservice architectures, where tokens may be verified repeatedly across a chain of services.

An analysis of scientific publications and standards reveals that much of the research focuses on the security of authorization and authentication protocols, the

stability of access schemes, and the construction of architectural templates for using JSON Web Tokens (JWT) in web services. However, the issue of comprehensively quantifying the impact of cryptographic algorithm choices for JWT token signing and specific authentication schemes on the performance of web-oriented applications within various architectural approaches (e.g., monolith, SOA, and microservices) has not been sufficiently researched. This issue is usually considered only for specific technology stacks or load scenarios. Thus, there is a need to develop models, methods, and software tools that enable a comparative analysis of these mechanisms using uniform criteria and performance metrics.

1. Authentication processes in web applications

Authentication is a fundamental process of identifying a system entity (user, device, or service) in compliance with security requirements. In the context of information systems, implementing authentication mechanisms is a prerequisite for preventing unauthorized access to protected resources and services.

Functionally, authentication and authorization solve different security lifecycle tasks:

authentication answers the question “Who are you?” by verifying the claimed identity of the entity.

authorization determines the answer to the question “What are you allowed to do?” by establishing the level of access rights for the authenticated subject.

implementing these mechanisms requires creating a secure infrastructure to verify and authenticate the relevant entities.

In web application architecture, the need for authentication arises when the client-side accesses the API provided by the server-side of the system. Below are the main authentication mechanisms that are widely used in modern web solutions.

1.1. Basic HTTP authentication

The Internet Engineering Task Force (IETF) standardizes a number of authentication and authorization mechanisms suitable for use by web clients. The most fundamental form of authentication is the transmission of user credentials (username and password) via the HTTP Authorization header. This mechanism follows a

“challenge–response” model in which the server initiates the authentication procedure when a protected endpoint is accessed, and the client provides the required credentials.

The standard defines two primary schemes for implementing this approach. Basic Authentication involves the client sending the username and password in Base64-encoded form, having first concatenated them into a string of the format "username:password" [21].

Digest Authentication is a more secure scheme in which the client constructs a response to the server’s challenge as a hash value derived from the username, password, and a unique nonce provided by the server [22]. Historically, the MD5 algorithm was used for hashing; however, modern versions of the specification provide for the use of a cryptographically secure algorithm such as SHA-256. Despite improved security characteristics compared to Basic Authentication, the Digest scheme has not gained widespread adoption in web development practice.

It is important to note that this mechanism, by itself, does not ensure the confidentiality of the transmitted data. Base64 encoding of credentials is sometimes mistakenly perceived as equivalent to encryption; in practice, however, such data can be easily decoded back into plaintext. Therefore, the security of these schemes depends entirely on the security of the transport channel. The de facto standard for securing data transmission is the use of HTTPS, which encrypts traffic using the SSL/TLS cryptographic protocols.

1.2. API keys

API keys are cryptographically random sequences that function as a shared secret between a client and an API provider. The authenticating subjects may be either individual users or entire projects/applications integrated with the API. When API keys are used for user authentication, the API server generates a unique secret key for each account. In this context, the API server (i.e., the back-end server) is responsible for key generation, secure storage (typically in a protected database), and validation of the key on every incoming request.

To prevent predictability, API keys must be generated using cryptographically secure algorithms. A practical approach is to use Universally Unique Identifiers

(UUIDs), which are random strings generated for each registered user. For additional protection against forgery, a digital signature mechanism may be applied using a shared secret key.

The key distribution process typically involves delivering the key to the user via an HTTP redirect after successful initial authentication or registration. Depending on the implementation, keys may be provided to the client in the form of:

cookies stored in the browser for the corresponding domain;

URL redirect parameters.

When the cookie mechanism is used, each subsequent request to the API server automatically includes the cookie, enabling session identification.

Despite the simplicity of implementing client authentication and session support, API keys have significant security limitations. Because they constitute a shared secret, compromising a key enables an attacker to impersonate a legitimate user. As an alternative approach, API keys are commonly recommended for identifying projects/applications rather than individual users.

In architectures where a front-end service acts as a proxy between clients and the API server, a higher level of security can be achieved. Users do not have direct access to the front-end server, which allows the API key to be stored securely. This service appends the API key to each proxied request, forwards the response to the client, and implements its own mechanisms for authenticating individual users.

API key provisioning typically requires developers to register their applications to obtain a unique key, often accompanied by a warning that it will be shown only once. A common anti-pattern is embedding API keys directly in front-end source code, which can lead to compromise when the code is published in public repositories.

The recommended methodology is to use environment variables to store confidential data, including API keys. Modern Platform-as-a-Service (PaaS) cloud platforms provide built-in tools for configuring environment variables without requiring code modifications. Embedding API keys in client-side JavaScript or otherwise exposing them in a web browser is an especially critical error [23, 24].

1.3. Token-based authentication

Token-based authentication is another approach that has gained popularity in recent years. The authentication mechanisms discussed earlier follow a stateful model, which leads to scalability constraints and architectural heterogeneity. Token-based authentication enables a stateless authentication flow in which the server is not required to maintain any client-related state.

In token-based authentication, a front-end application interacts with a dedicated Identity Provider (IdP) by redirecting the user to the IdP's interface. The IdP performs user authentication, typically using a username–password pair, with optional additional verification factors (e.g., one-time passwords). Upon successful authentication, the IdP issues and returns to the client:

an ID token, which confirms the user's identity;

an access token, which grants access to protected APIs.

The front-end application stores the access token for subsequent use when requesting protected resources.

OpenID Connect (OIDC) defines a standardized protocol for token-based authentication [25], built on top of the OAuth 2.0 specifications. A key element of OIDC is the scope parameter with the value OpenID, which must be included in the authentication request [26, 27]. The authorization server returns an ID token in JWT format that contains identification information about the authenticated user as claims. OIDC specifies a clear set of rules for validating the ID token before relying on its contents. It should also be noted that the claims from the ID token are available via the authorization server's /userinfo endpoint [25], access to which requires presentation of a valid access token.

ID tokens contain a standard set of parameters that characterize the user session state:

issuer identifier;

subject identifier (user);

audience (intended token consumers);

expiration time;

authentication time.

A critical parameter of the authentication request is response_type, which must include the value id_token in order to obtain an ID token [28].

OIDC defines different authentication flows depending on the application type. For architectures with predominantly server-side logic processing (traditional web applications or SPAs with server-side middleware), OIDC specifies the Authorization Code Flow. In this case, the OIDC client receives an authorization code from the IdP and then exchanges it for an ID token and an access token via the token endpoint [26, 27]. According to the OAuth 2.0 specification, the authorization server always returns an access token; OIDC extends this requirement by also mandating the return of an ID token [29].

The need for an access token depends on the application architecture. In scenarios where only confirmation of the user's identity is required, the claims contained in the ID token are sufficient. In such cases, the access token may be used solely to retrieve additional user information via the /userinfo endpoint.

A sequence diagram of the Authorization Code Flow is shown in Fig. 1.

An important stage in the token-based authentication procedure is client identification when calling the token endpoint. The OIDC specification defines a set of mandatory client authentication methods that the Identity Provider (IdP) must support. Each of these methods uses different mechanisms to verify the client's authenticity via the *client_secret* parameter.

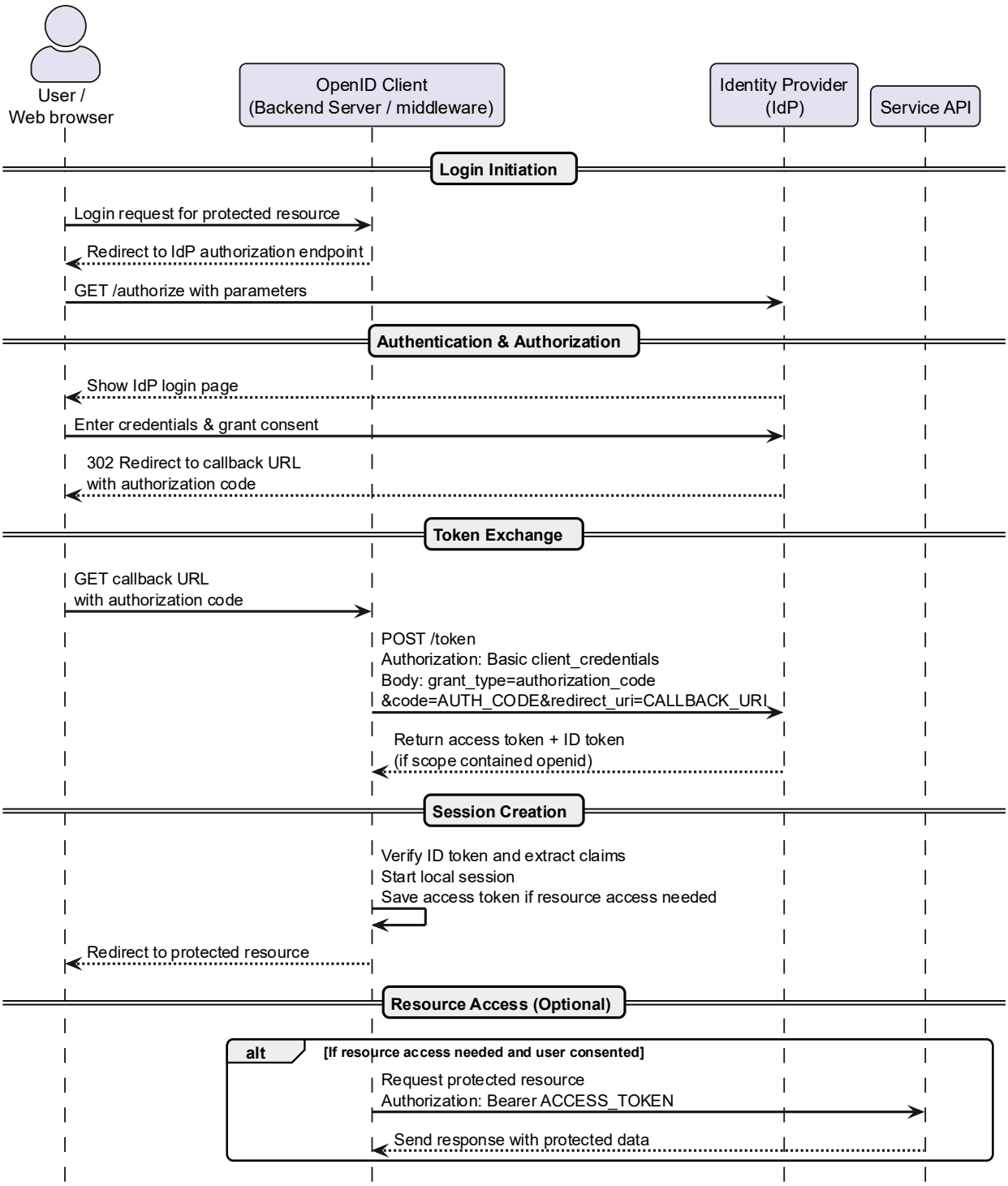


Figure 1. Token-based Authorization Code Flow

Source of the figure: original

1.4. Session Management

ID tokens and access tokens issued by the Identity Provider (IdP) typically have a limited validity period in order to enhance security. This necessitates periodic token

renewal by the client once the current tokens expire. An OIDC client is obliged to manage sessions for both token types, which may have different lifetimes.

Functionally, tokens are classified as follows:

identity token is intended for the OIDC client and contains information about the user's identity and authentication state, presented as claims in the payload. It is used to build the user profile.

access token is directed at the resource server and provides authorized access to protected resources on behalf of the user.

According to the OIDC specification, a user session is the period during which the OIDC client maintains a valid authenticated status with the IdP. The session management process involves monitoring this status. Session termination leads to invalidation of all active tokens, even if their formal expiration time has not yet been reached. Subsequent acquisition of new tokens requires user re-authentication.

This approach provides an additional layer of security because it enables immediate access revocation in the event of detected credential compromise or other security threats, regardless of the configured lifetimes of individual tokens. To maintain application continuity, it is recommended to implement automatic token renewal mechanisms using refresh tokens, which allow new access tokens to be obtained without user involvement (see Fig. 2).

To support continuity of authentication sessions in OIDC, two primary approaches are used, one of which is silent re-authentication [25].

This mechanism assumes that the web application initiates a standard authentication procedure using the same parameters as in the initial request to the IdP authorization endpoint. The key difference is that the client periodically contacts the OpenID Provider to check the end user's session status without the user's participation.

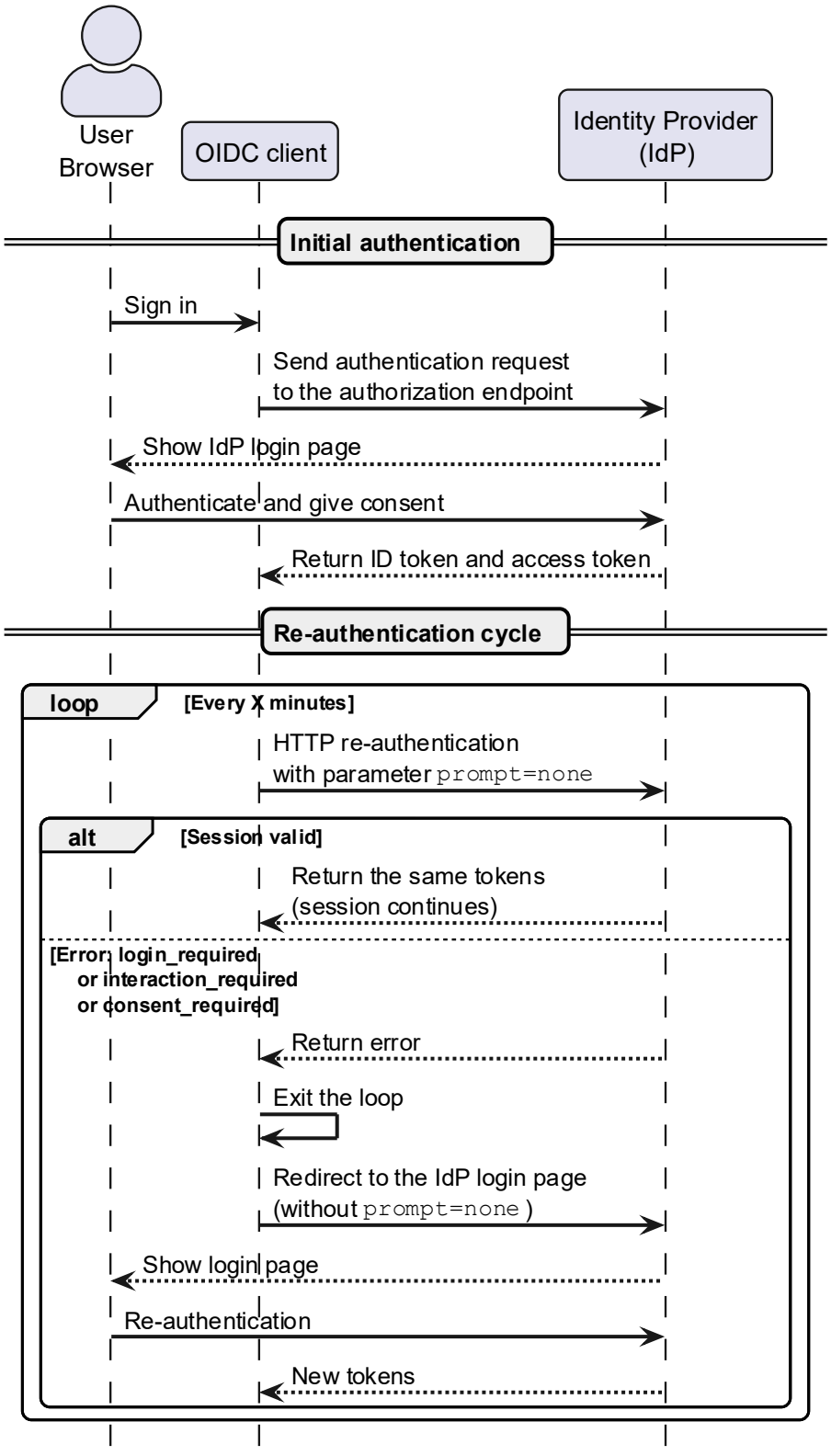


Figure 2. Session management using silent authentication

Source of the figure: original

A key technical parameter of this process is prompt=none, which instructs the Identity Provider to avoid interactive engagement with the user. Provided that an active session exists on the IdP side, the provider automatically returns the requested tokens.

The second method for maintaining sessions involves implementing a mechanism based on hidden iframes integrated with the Identity Provider (IdP). This approach is based on the principle of regularly polling the session status by the web client at predefined intervals [30].

1.5. Single Sign-On Integration

The Single Sign-On (SSO) system enables a user to utilize a single set of credentials to access multiple applications within an organizational environment. Implementing this architecture involves centralized storage and management of credentials at the organization's Identity Provider (IdP).

Principles of SSO Operation:

trusted infrastructure: each organizational application establishes a trust relationship with the IdP.

unified authentication point: the user is redirected to the IdP’s single sign-in page for all applications.

efficient status propagation: after successful authentication, the IdP conveys the confirmed identity status to the application.

In the context of OIDC, authentication status is propagated via an ID token, which contains verified information about the user [31]. This mechanism provides a standardized way for trusted parties to exchange authentication information, eliminating the need for separate authentication for each application.

The advantages of this approach include not only improved convenience for the end user but also enhanced security through centralized account management and the ability to enforce a unified security policy across the organization’s application ecosystem.

1.6. JWT Tokens

JWT is the de facto standard for implementing ID tokens in modern authentication systems [32]. The payload of an ID token contains a set of claims that

represent identification information about the user. Client applications use these data to personalize the user interface and adapt functionality.

A JWT is a JSON-based token format exchanged between parties. It is easy to transfer and compactly encodes information about a user or a session. A JWT (Fig. 3) typically consists of:

Header – the first part of the JWT is a JSON object encoded using Base64URL. It specifies metadata such as the signing algorithm (e.g., RS256).

Payload – contains the data conveyed by the token, including the user identifier, user information, session details, permissions, expiration time, and related attributes.

Signature – the Base64URL-encoded header and payload are concatenated and signed using a private key (for asymmetric algorithms) or a shared secret (for symmetric algorithms). The resulting signature is then Base64URL-encoded.

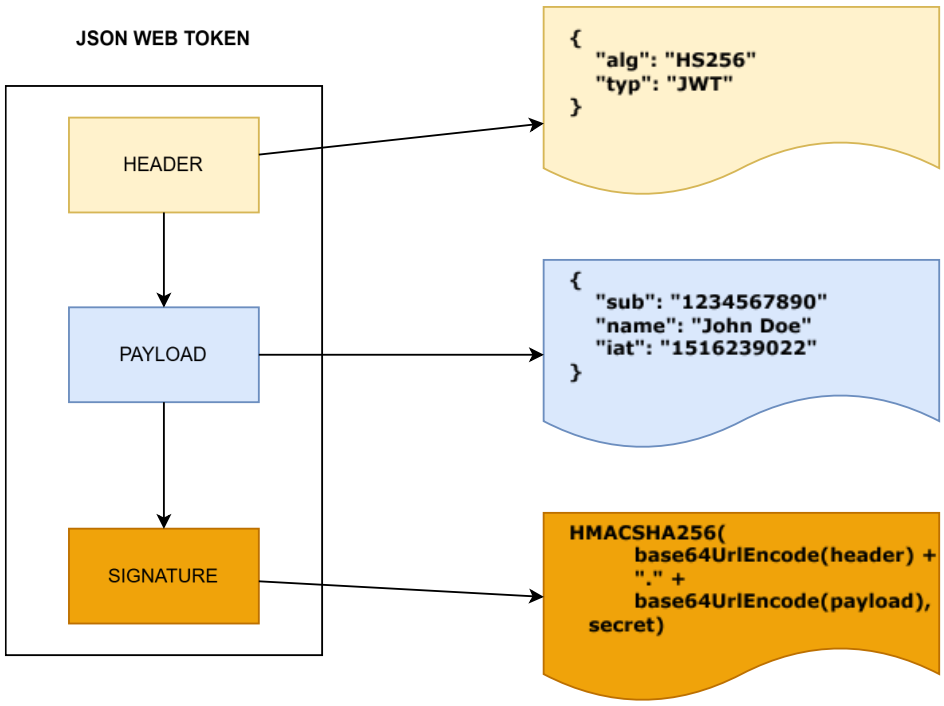


Figure 3. Structure of a JWT Token

Source of the figure: https://www.miniorange.com/blog/what-is-jwt-json-web-token-how-does-jwt-authentication-work/

JWT validation and processing:

token decoding – the front-end application first decodes the JWT from the Base64URL format defined by the standard [32];

digital signature verification – a mandatory step is verifying the JWT's cryptographic signature to confirm data integrity and authenticity;

payload analysis – after successful verification, the application gains access to the structured data in the form of a JSON object.

Base64URL encoding enables safe transmission of the token via HTTP headers by avoiding issues with special characters inherent in standard Base64. However, without prior signature verification, payload data must not be trusted, since it can be easily inspected after simple Base64URL decoding.

This mechanism allows applications to obtain up-to-date user information without additional server requests, which significantly improves performance and user experience.

Previous studies have evaluated token-based authentication using JWT, noting that JWT-based approaches may outperform alternative methods in terms of authentication mechanisms, access control mechanisms, compact and self-contained token structure, support for multi-factor authentication/SSO, and scalability of the access control system [31]. Other comparative evaluations analyzed JWT-based systems using both single and multiple tokens in both single-server and multi-server deployments, indicating that multi-token designs may exhibit lower performance than systems relying on a single token [33]. Common algorithms for JWT include HMAC-SHA-256, RSASSA-PKCS1-v1_5 with SHA-256, and ECDSA P-256 with SHA-256 [34]. In addition, RFC 8032 [35] defines newer algorithms, such as Ed25519 and Ed448; however, performance test results for these modern algorithms have not yet been widely published.

2. Schemes for securing microservices using JWT

JWT addresses two main challenges in microservice security design: protecting communication between services and propagating end-user context across microservices (Fig. 4). All microservices within a deployment trust a Security Token

Service (STS). The API Gateway exchanges JWTs received from client applications for new JWTs issued by the STS. In most cases, JWT is used together with mTLS

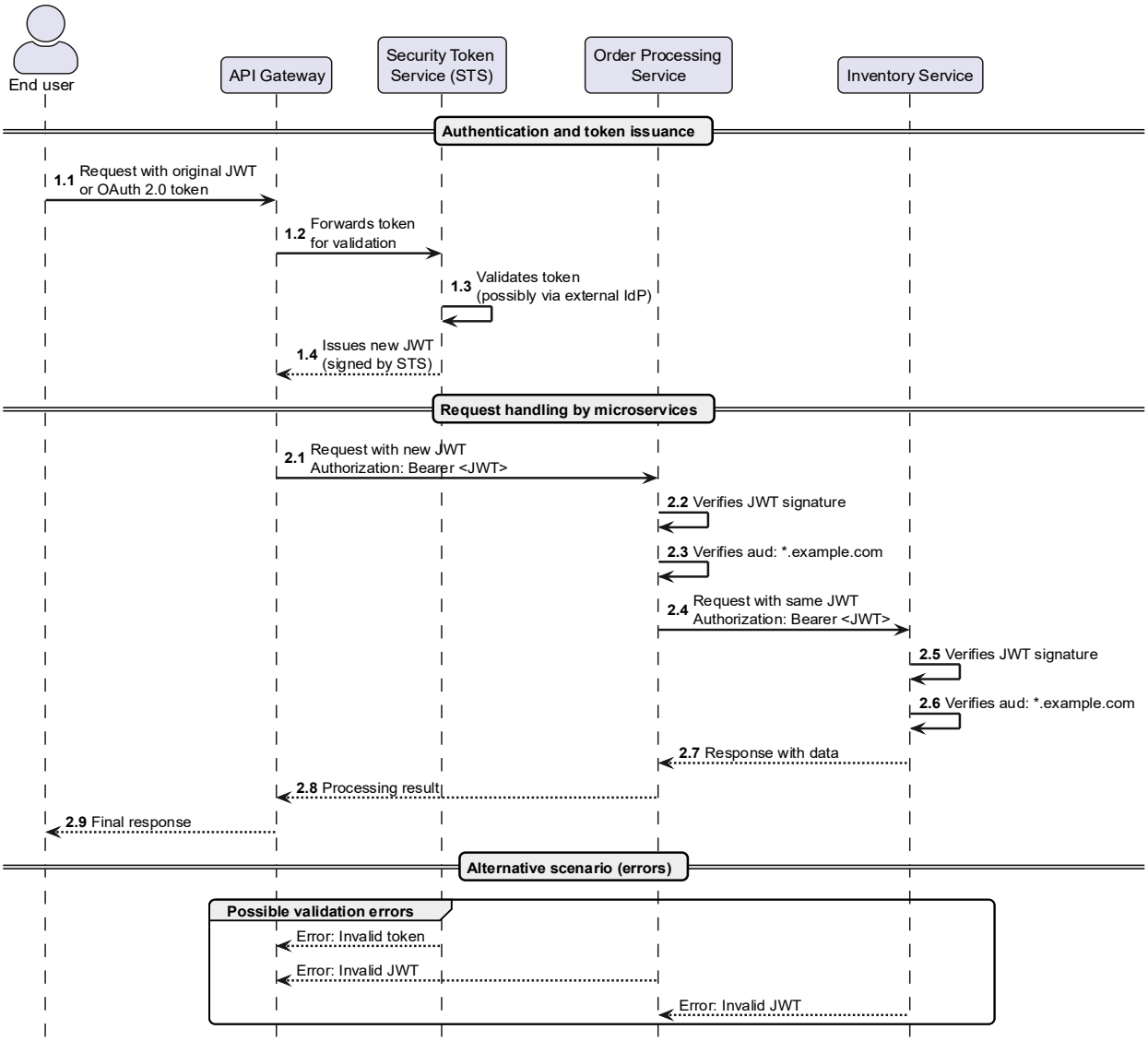


Figure 4. Propagation of end-user identity data via JWT among microservices

Source of the figure: original

2.1. Scheme for Sharing End-User Context Between Microservices Using a Shared JWT

When microservice-to-microservice identification is not critical but end-user identification (human or system) is required, using JWT should be considered. In this case, the services do not authenticate each other directly. Each request must carry the identity of the end user who initiates the message flow; otherwise, the receiving

microservice rejects the request. The flow for propagating end-user identity data in a JWT among microservices, as shown in Fig. 3, is as follows:

1.1. The end user initiates the request flow. This end user may be a human or a system.

1.2. The API Gateway authenticates the end user. The gateway intercepts the request, extracts the token (which may be an OAuth 2.0 reference token or a self-contained token), and interacts with the connected Security Token Service (STS) to validate it. The token provided by the end user may not have been issued by this STS; it can originate from any other identity provider trusted by the STS. Therefore, the STS must be able to validate the token presented at this step.

1.3. After validation, the STS issues a new JWT signed by the STS. This JWT contains end-user data copied from the original token (Step 2). When the API gateway forwards the new JWT to upstream microservices, those microservices only need to trust this STS to treat the token as valid. Typically, all microservices within a single trust domain rely on the same STS.

2.1. The API gateway forwards the STS-issued JWT to the Order Processing microservice over TLS, using the HTTP Authorization header with the Bearer scheme. The Order Processing microservice verifies the JWT signature to ensure that it was issued by a trusted STS. In addition to signature verification, it validates the `aud` (audience) claim. For the pattern described in this section to work, all microservices in the same trust domain (i.e., trusting the same STS) must accept JWTs with a wildcard `aud` value, such as **.shop.com*.

2.2. When the Order Processing microservice calls the Inventory microservice, it propagates the same JWT received from the API gateway. The Inventory microservice verifies the JWT signature to confirm that it was issued by the trusted STS and checks that the `aud` claim equals **.shop.com*.

This approach allows JWT to achieve two objectives. First, it enables propagation of end-user context across microservices in a way that is difficult to forge: because the JWT claim set is signed by the STS, no microservice can modify it without invalidating the signature. Second, it helps secure service-to-service communication: a

microservice can call another microservice only if it presents a valid JWT issued by a trusted STS. Any receiving microservice rejects requests that do not include a valid JWT.

2.2. User Context Sharing Scheme with a Session Service JWT for Each Inter-Service Interaction

The user-context sharing scheme based on issuing a session service JWT for each inter-service interaction is essentially a modification of the shared-JWT scheme. The objective remains limited to establishing the identity of the end user, whereas microservice identification is not critical. Instead of propagating the same JWT across all microservices with a single audience value accepted by each service, a new JWT is generated for each distinct service interaction. This approach provides a higher level of security compared to using a shared (distributed) JWT.

However, it should be emphasized that absolute security is unattainable: the appropriateness of a given approach depends on specific usage scenarios and the level of trust in the microservices deployment infrastructure.

The principle of operation of the circuit is shown in Fig. 5. The interaction flow is similar to that described in the previous section, except for Steps 2.1 and 2.2.

In Step 2.1, before the Order Processing microservice initiates interaction with the Inventory microservice, it contacts the STS and performs a token exchange operation. It submits the JWT received from the API gateway (issued with `aud`: op.shop.com) and requests a new JWT to access the Inventory microservice (see Fig. 4).

In Step 2.2, the STS issues a new JWT with `aud`: *iv.shop.com*. The processing flow then continues according to the scenario outlined in the previous subsection.

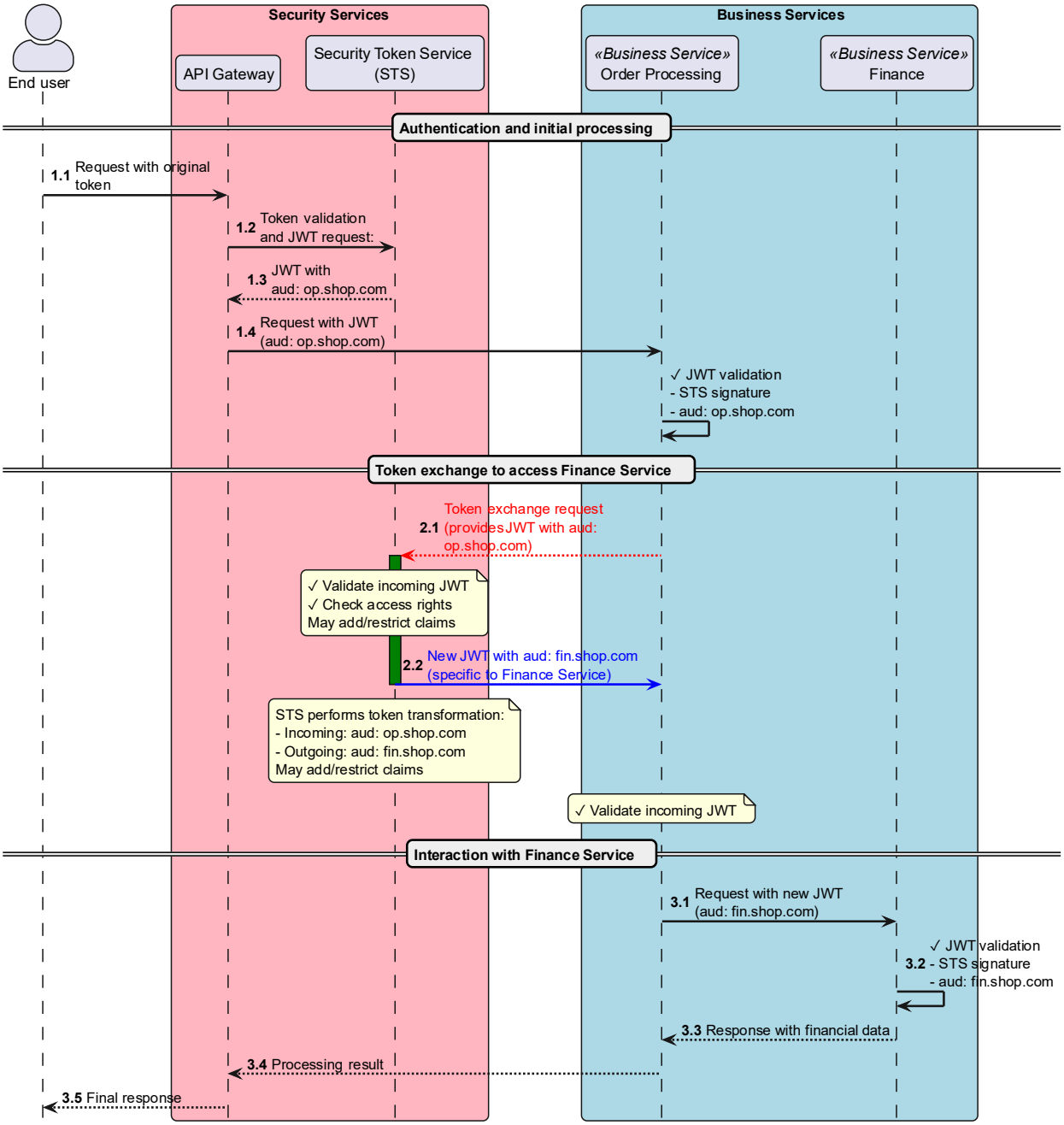


Figure 5. Propagation of end-user identity data in JWTs across microservices using token exchange

Source of the figure: original

The need to issue a new JWT with a new `aud` value during the interaction between the Order Processing microservice and the Inventory microservice is driven by the requirements of trust isolation between services and attack-surface minimization. This approach is more secure than allowing all microservices within a

deployment to share the same JWT received from the API gateway under a single audience value. At least two significant reasons can be highlighted:

*explicit mapping between a microservice and its audience (*aud*) value*. If each microservice in the deployment is assigned a unique audience value and the STS issues JWTs with that value, then for any token it is unambiguous which microservice is the intended recipient. For example, in Step 1.4 of Fig. 2.2, when a request arrives at the Order Processing microservice from the API gateway, the microservice can strictly verify that the token is intended for it rather than for another microservice. If the JWT is mistakenly or maliciously routed to a different microservice, it will be rejected due to an aud mismatch. This prevents incorrect or unauthorized use of the token outside its intended audience.

prevention of unauthorized token reuse across microservices. If the Order Processing microservice attempts to reuse the received JWT "as is" to access another service (e.g., a Finance microservice, which ideally it should not be permitted to access), the request will be rejected because the aud value in the original JWT does not match the audience expected by the Finance microservice. Consequently, the only way for the Order Processing microservice to interact with the Finance microservice is to submit its current JWT to the STS and perform a token exchange to obtain a new JWT with an audience value acceptable to the Finance microservice. In this model, access-control decisions are centralized at the STS: the STS determines whether the Order Processing microservice is allowed to obtain a token for accessing the Finance microservice. This enables finer-grained and more manageable control over inter-service interactions and implements the principle of least privilege.

2.3. Scheme for Sharing User Context Between Microservices in Different Trust Domains

The use case in this section extends the token-exchange use case discussed in Section 2.1.2. As shown in Fig. 6, most steps are straightforward. There are no changes relative to the previous section up to Step 3.2. (Steps 3.1 and 3.2 in Fig. 6 are equivalent

to Steps 2.1 and 2.2 in Fig. 5.

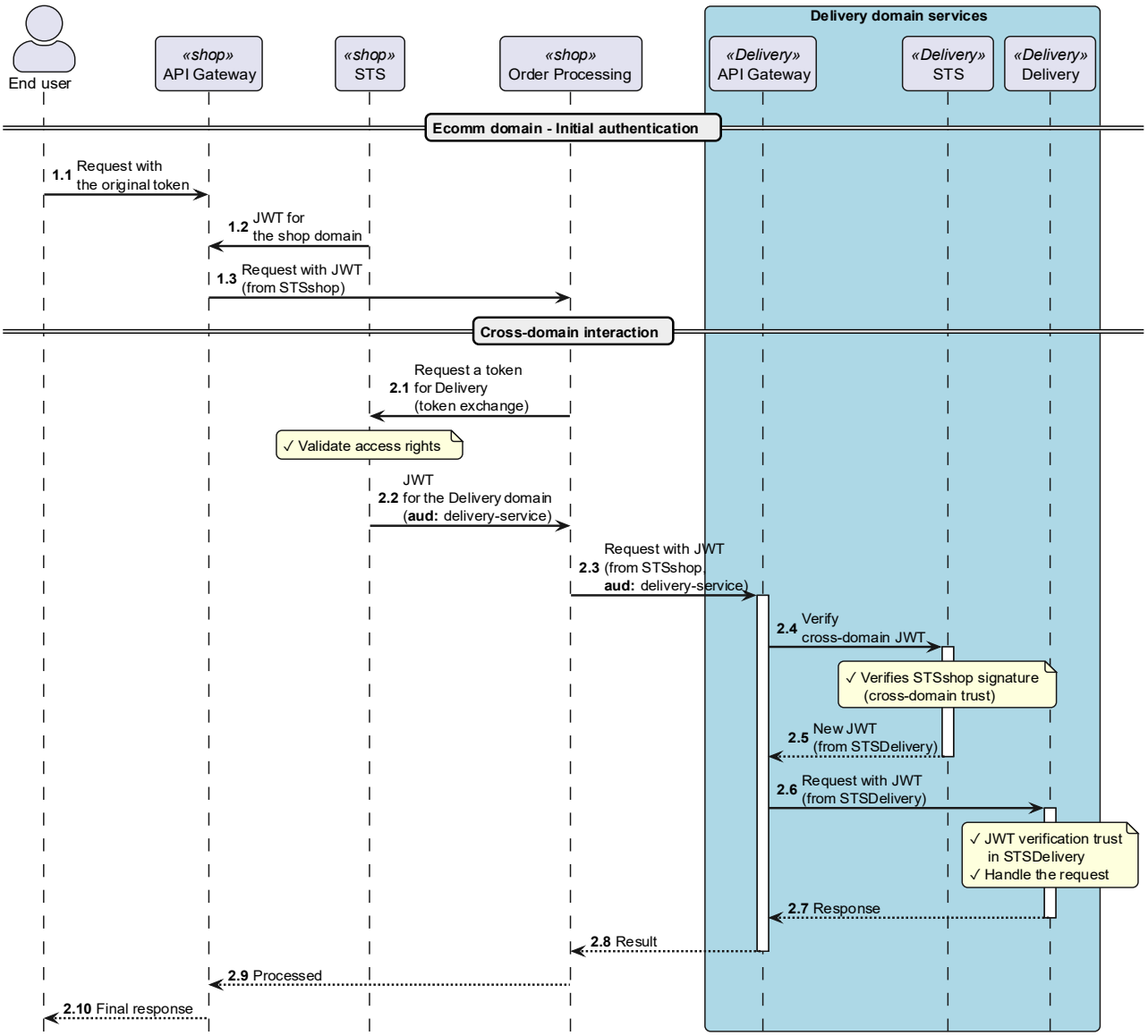


Figure 6. Cross-domain authentication and sharing of user context across multiple trust domains

Source of the figure: original

In Step 2.3, the Order Processing microservice in the shop domain attempts to access the Delivery microservice in the delivery domain via the delivery API gateway. The JWT included in this request (Step 2.3) is issued by the STS in the shop domain and has an audience value corresponding to the Delivery microservice.

In Step 2.4, the API gateway in the delivery domain interacts with its own STS to validate the JWT. Validation succeeds only if the delivery STS trusts the shop STS.

In other words, the public key required to verify signatures on JWTs issued by the shop STS must be known to the delivery STS.

If this condition is satisfied, then in Step 2.5 the delivery STS issues its own JWT and forwards it to the Delivery microservice via the API gateway. All microservices within the delivery domain trust only the STS of their own domain.

2.4. Self-Issued JWTs

In the previous use cases, the identity of microservices as separate entities was not critical. The system relied on a JWT issued by a trusted STS that contained the end-user identity. In contrast, in the self-issued JWT model (see Fig. 7), identifying microservices during service-to-service interaction becomes essential, similarly to the mTLS model.

As with mTLS, in this model each microservice must possess its own cryptographic public–private key pair. The initiating microservice constructs a JWT, signs it with its private key, and sends it along with the request to the receiving microservice via the HTTP Authorization header using the Bearer scheme over a secure TLS channel. Because the JWT in this case is a bearer token, TLS is effectively mandatory to protect against interception and subsequent unauthorized use. The receiving microservice verifies the JWT signature using the initiating microservice's public key, thereby unambiguously identifying it.

A self-issued JWT should include the standard JWT claims (see Table 1), as well as any additional claims (see Table 2) required by the application, including the key claims that are commonly used.

Table 1

1) Standard JWT Claims

Claim	Name	Description
1	2	3
`iss`	Issuer	Identifier of the service that issued the token. In the case of a self-signed (self-issued) JWT, this is the identifier of the microservice that created it (e.g., "order.shop.com").
`sub`	Subject	Identifier of the subject (often matches `iss`). For a microservice, this may be the identifier of the microservice itself or the user on whose behalf it acts. However, for self-issued JWTs, `sub` frequently contains the same identifier as `iss`.

Continuation of Table 1.

1	2	3
`aud`	Audience	Target service(s) for which the token is intended. This may be one or multiple services. For example, "inventory.shop.com".
`iat`	Issued At	Token issuance time in Unix timestamp format.
`exp`	Expiration	Token expiration time.
`jti`	JWT ID	Unique token identifier used to prevent replay attacks.

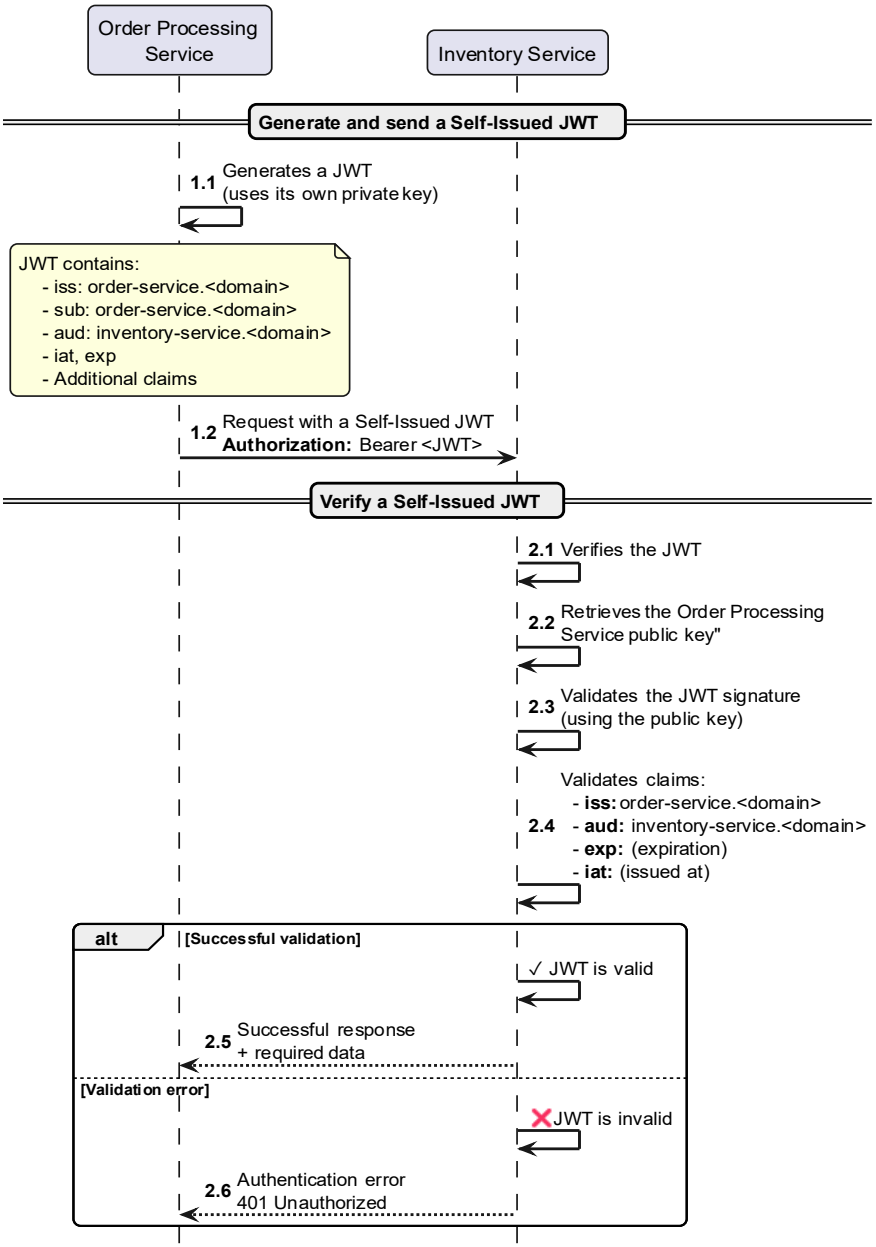


Figure 7. Self-issued JWT. The JWT is signed using the private key of the order-processing microservice <JWT>

Source of the figure: original

Table 2

2) Recommended JWT Claims

Claim	Name	Description
nbf	Not Before	The time from which the token becomes valid.
scope	Scope	A list of permissions granted by the token (e.g., "inventory:read inventory:write"). This is not a standard JWT claim, but it is commonly used in OAuth 2.0.
roles	Roles	The service's roles within the system.
service_id	Service ID	The identifier of the service used to discover/resolve it within the environment (Docker, Kubernetes). The ID should be a URI (domain name) or a set of aliases that clearly identify a specific resource or a group of resources.
service_version	Service Version	The service version used for tracing/observability.
environment	Environment	The execution environment (production/staging/development).

Custom claims may also be added when additional information must be conveyed (e.g., roles, access rights, etc.; see Fig. 8).

```
{
  // Main claims
  "iss": "order-service.example.com",
  "sub": "order-service.example.com",
  "aud": "inventory-service.example.com ",
  "iat": 1620000000,
  "exp": 1620003600,
  "jti": "550e8400-e29b-41d4-a716-446655440000",

  // Additional safety claims
  "nbf": 1620000000,

  // Authorization requests
  "scope": "inventory:read inventory:write",
  "roles": ["order_processor", "data_reader"],

  // Service context statements
  "service_id": "order-service-v1",
  "service_version": "1.2.3",
  "environment": "production",

  // Business context (optional)
  "tenant_id": "tenant-123",
  "permissions": ["read_inventory", "update_stock"]
}
```

Figure 8. Example of a JWT token payload

Source of the figure: original

2.5. Security Considerations for Bearer Tokens

A JWT used as a bearer token is inherently similar to cash: anyone who possesses the token can use it until it expires, without providing additional proof of ownership. If an attacker intercepts a JWT, they can issue requests on behalf of the legitimate token holder for as long as the token remains valid.

Therefore, when JWT is used for authentication between microservices (i.e., for mutual service-to-service authentication), it is necessary to:

secure the communication channel using TLS to minimize the risk of token interception;

keep the JWT lifetime as short as possible (short-lived tokens) to reduce the potential impact of a compromised token.

When these requirements are met, the risk of abuse of a stolen token is significantly reduced, and the self-issued JWT model becomes a powerful mechanism for implementing secure inter-service communication while preserving context and supporting non-repudiation properties.

2.6. Nested JWTs

The usage variant considered in this subsection is an extension of the scenario described in Section 2.1.3. A nested JWT is a token in which one JWT encapsulates another (see Fig. 9). A typical use case is to protect communication between two microservices using a self-issued JWT that contains an inner JWT issued by a trusted Security Token Service (STS) and carrying the end-user context. The resulting nested JWT therefore combines two layers of trust: trust in the STS and trust in the calling microservice.

On the recipient side, the microservice must:

1) verify the signature of the outer (enclosing) JWT using the public key of the calling microservice;

2) extract the inner JWT and verify its signature using the corresponding public key of the trusted STS.

Thus, the nested JWT simultaneously contains the end-user identity data and the calling microservice’s data, while the integrity and authenticity of these data are

cryptographically protected: forging such a token without compromising the corresponding private keys is practically impossible.

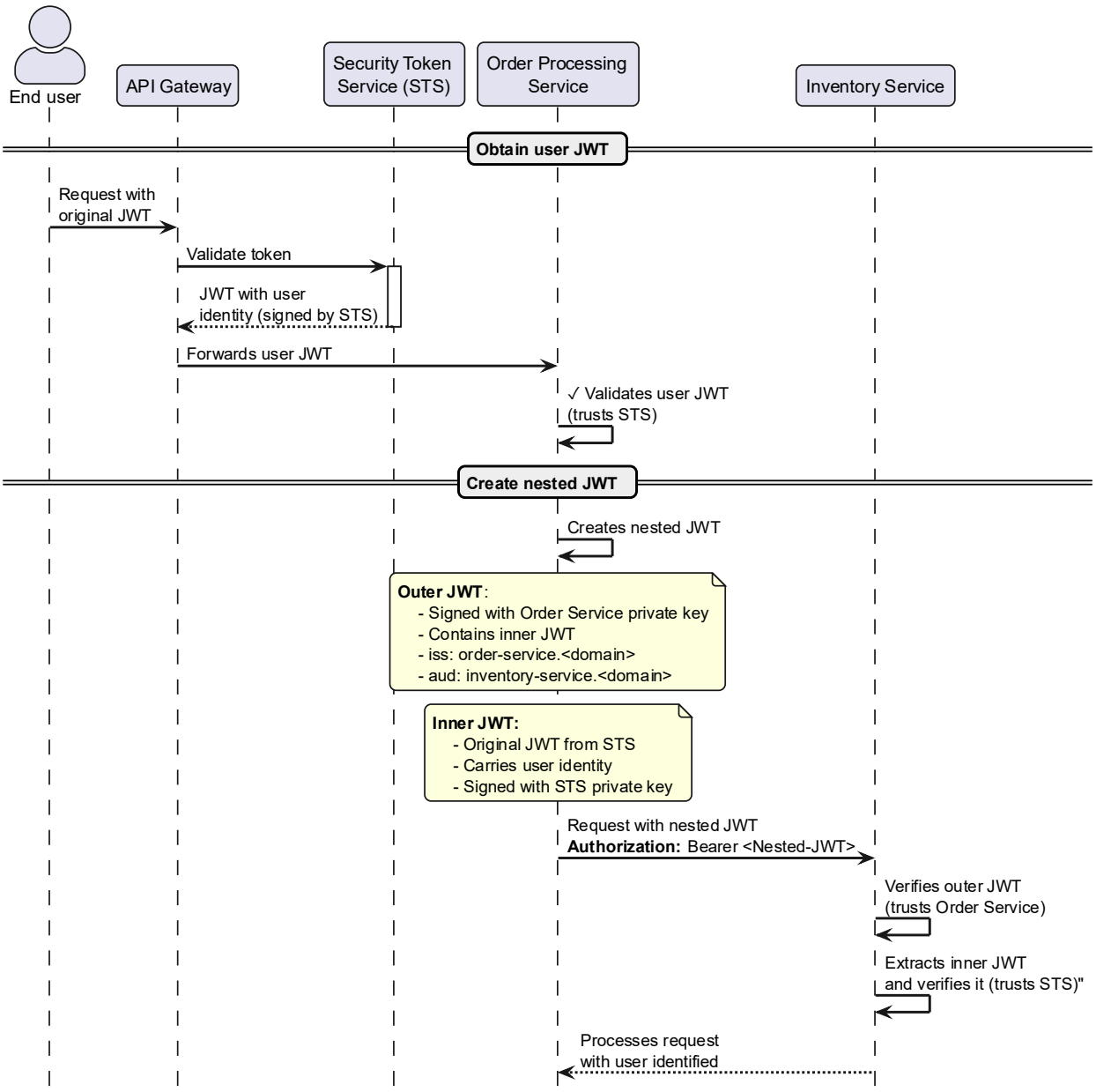


Figure 9. Nested JWT: The order-processing microservice creates its own JWT and embeds within it the JWT received from the next microservice (or the API Gateway)

Source of the figure: original

3. Model for Implementing JWT-Based Authentication in the Web-Oriented Microservice E-Commerce System “Shop”

A prototype model of a JWT-based authentication service is developed for the web-oriented e-commerce system “Shop,” built using a microservice architecture. The application domain is an online store where users register, browse the product catalog, manage a shopping cart, place orders, and handle their financial accounts. Such systems are characterized by a high volume of concurrent requests, the need for horizontal scaling, and heightened security requirements for operations involving money and personal data. Therefore, the objective is to build a prototype that demonstrates how JWT can be used as a basic mechanism for authentication and for propagating user identity across independent microservices.

The overall structure of the Shop model is shown in Fig. 10. At the top level, infrastructure services are distinguished from functional services of the e-commerce domain. The infrastructure layer includes the User Authentication service (User AuthN) and the Security Token Service (STS). The STS is responsible for issuing cryptographically protected JWTs, while User AuthN interacts with the user, validates credentials, and generates token-issuance requests. Another key infrastructure element is the API Gateway, which serves as the single external entry point to the microservice ecosystem of the Shop.com model. It receives all HTTP requests from clients, performs preliminary checks and routing, and is responsible for transmitting JWTs between the user and internal services.

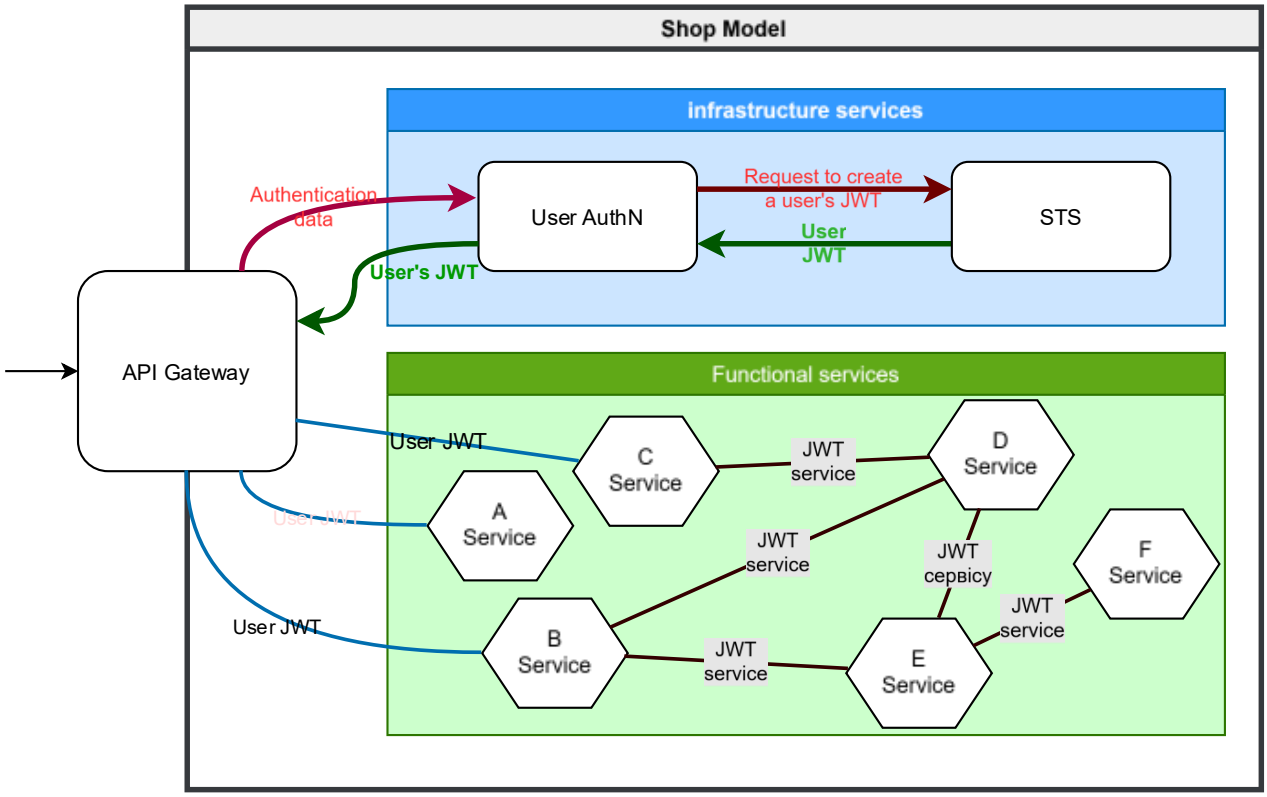


Figure 10. Shop model. Primary propagation via a JWT token

Source of the figure: original

The functional layer consists of a set of business services that implement the core capabilities of the online store. These include the User Service, Account Service, Product Service, and Basket Service, which are conventionally labeled in the diagram as Services A, B, C, D, E, F. Each of these services is autonomous: it has its own database, implements a limited set of well-defined operations, and scales independently. All inter-service calls are performed via HTTP interfaces and are accompanied by JWT transmission, ensuring end-to-end propagation of user identity and enforcement of access policies at the level of each individual microservice.

The STS is the core of the security model. Its primary function is to issue JWTs for user sessions and, when required, service tokens for inter-service communication. Upon successful authentication, the STS generates a token that includes the user identifier (UserID), a role or set of roles, additional attributes (e.g., a list of permitted operations or identifiers of associated accounts), as well as the issuance time and

expiration time. The token is signed with the STS private key using one of the selected algorithms from the JWT family (e.g., HS256, RS256, EdDSA/Ed25519) and is then returned through User AuthN to the API Gateway. From that point onward, this token becomes the primary carrier of information about the user's authenticated identity for the entire system.

The API Gateway acts as a facade for clients and as a central router for internal services. All initial requests arrive at the gateway directly from a browser or a mobile application. If the user is not yet authenticated, the request is routed to User AuthN together with credentials (username and password, and other authentication attributes). After successful verification, User AuthN submits a request to the STS to issue a user JWT. The token received from the STS is returned to the API Gateway and then to the client, typically in a response header or as an HTTP cookie. Subsequently, the client attaches this JWT to every request (e.g., in the Authorization: Bearer header), and the gateway verifies its signature, validity, and lifetime. Only after successful validation is the request forwarded to the target functional service together with the identity token.

The User Service is responsible for the lifecycle of user accounts. It provides operations for creating and deleting users, changing passwords, updating profiles, and initiating login. Upon registration, a unique UserID is created and subsequently used as the primary identifier across all other services. During login, the User Service does not issue a token itself, but delegates this function to the STS via User AuthN. This centralizes security policies and allows signing algorithms or token formats to be changed without modifying business logic. The obtained JWT is stored on the client side.

The Account Service models the financial subsystem of the online store. Each user has one or more accounts linked to their UserID. The service supports opening and closing accounts, retrieving a list of accounts for a specific user, and changing account balances (debiting funds for purchases, deposits, refunds, etc.). All requests to the Account Service must contain a valid JWT. The service extracts the user identifier from the token and checks whether the user is authorized to perform the requested operation on the specific account. Thus, even if a request reaches the service directly

without passing through the gateway (e.g., as a result of an internal call from another service), authorization checks are still performed based on the same token.

The Product Service implements the product catalog. It stores product descriptions, prices, stock levels, and other attributes in its own database. The service provides operations for adding new products, updating characteristics and prices, retrieving product lists or detailed information for a specific productID, and modifying inventory when orders are placed. For most read operations, a valid user token is sufficient, whereas administrative actions (creating and editing products) require a specific role in the JWT (e.g., role = admin). This keeps authorization logic straightforward: the Product Service relies on the already validated token and does not consult a centralized session store.

The Basket Service is responsible for forming and managing orders. Each user has a separate basket identified by the UserID contained in the JWT. The service supports adding items to the basket, removing items, changing quantities, clearing the basket, and initiating checkout. When the user proceeds to payment, the Basket Service interacts with the Product Service (to verify availability) and the Account Service (to verify and reserve the required balance). All such internal interactions are accompanied by token propagation: in the simplest case, the same user JWT is used; however, the model also allows issuing service JWTs ("service JWTs") with restricted permissions that duplicate or refine the user context for inter-service calls.

A key characteristic of the model is that both external user access and internal calls between services follow the same token-based authentication principle. In Fig. 2.8 this is represented by two types of markers: the "User JWT" passed from the API Gateway to services A, B, C, etc., and the "Service JWT" used during inter-service interactions (e.g., when Service B calls Service E). This approach simplifies end-to-end security policy enforcement, enables new services to be added without changes to central authentication logic, and supports more advanced schemes, including nested tokens and delegated authorization.

For correct operation of the STS in the shop.com prototype, three key assumptions are formulated. First, the time synchronization problem is considered

solved: the clock of the node issuing tokens and the clocks of all nodes validating them are synchronized. This ensures correct interpretation of issuance time (iat), not-before time (nbf), and expiration time (exp) embedded in JWTs, and minimizes the risk of premature rejection or acceptance of expired tokens. Second, all channels used to exchange tokens are assumed to be protected by TLS or an equivalent mechanism. If the channel is not secure, a token may be intercepted and reused (replay attack) within its validity period. To mitigate this risk, token lifetimes in the prototype are limited to relatively short intervals. Third, the STS private key is stored in a secure environment and is inaccessible to attackers; otherwise, attackers could sign their own JWTs and present them as legitimate.

The token-based authentication mechanism selected for the Shop model is typical for modern distributed systems. The token acts as a compact cryptographic object that encodes all information required to confirm user identity and verify access rights. Token issuance is performed on the server side, while the client is responsible only for correctly transmitting tokens in subsequent requests. Due to token self-containment, microservices do not require a shared session store and do not maintain user state in memory; for each request, the context is reconstructed from the JWT. This substantially improves system scalability and simplifies deployment of new service instances.

The model accounts for the fact that token-based authentication can be implemented using different authorization approaches: role-based, attribute-based, and fine-grained access control. The STS prototype supports embedding both roles (e.g., USER, ADMIN, SUPPORT) and specific permissions into JWTs, which are interpreted directly by business services. For example, the Account Service may rely only on the user role, the Product Service may check an attribute permitting product creation, and the Basket Service may consider a maximum order amount limit encoded in the token. Thus, the prototype does not constrain authorization policy, but provides a flexible mechanism for implementing it.

The request-processing flow in the prototype can be summarized as follows:

1. An incoming HTTP request from the client arrives at the API Gateway. If the request already contains a JWT, the gateway immediately validates it (signature validity, non-expiration, absence of revoked identifiers, and other rules).

2. If the token is absent or validation fails, the user is redirected to User AuthN to perform login.

3. Based on data from the User Service, User AuthN submits a request to the STS to issue a new token.

4. The STS issues an access token, which is returned to the gateway and, depending on the scenario, to the client.

5. The gateway forwards the request to the appropriate functional service, attaching the JWT in the request headers.

6. The functional service re-validates the token (if required by policy) and executes the business operation.

7. If the operation requires calls to other services, the current service either forwards the user JWT or requests a service token from the STS and uses it for inter-service interactions.

The prototype implementation emphasizes that security cannot be evaluated solely from a performance standpoint, yet its impact on microservice performance is critical. Each JWT validation operation incurs additional computations: cryptographic signature verification, processing of header and payload fields, and analysis of authorization attributes. In a microservice architecture, where a single user request often triggers a cascade of calls across multiple services, the token may be validated repeatedly. For this reason, the Shop.com prototype model is subsequently used as a test environment to study the trade-off between security level and cryptographic overhead under different algorithms and token propagation strategies.

Therefore, the proposed JWT-based authentication service prototype for the Shop model demonstrates a comprehensive approach to security in a microservice architecture. It combines centralized token issuance in the STS, a single entry point via the API Gateway, autonomous business services (User, Account, Product, Basket, and

others), and end-to-end propagation of user identity using cryptographic tokens. The model clearly separates authentication and authorization concerns from business logic, ensures scalability and flexibility, and provides a foundation for further experiments measuring how different JWT implementation variants affect the performance of e-commerce microservices.

3.1. Model for Testing JWT Token Performance

The JSON Web Algorithms (JWA) specification is a key element of the JOSE standards family [34, 35], because it formally defines the permissible cryptographic algorithms for producing digital signatures and Message Authentication Codes (MACs) used in JWT and JSON Web Signature (JWS). In the JWT context, the alg header parameter indicates the selected signing or MAC algorithm, while JWA defines the allowed values for this parameter, usage rules, and implementation requirements. This design ensures interoperability between independent implementations and enables clients and servers to unambiguously interpret the cryptographic protection applied to a particular token.

In JWA, algorithms are grouped by cryptographic primitives. Symmetric algorithms include HMAC based on SHA-2 hash functions (HS256, HS384, HS512), which use a shared secret key and are typically applied in configurations where issuance and verification occur within a single trust domain. Asymmetric algorithms include RSA-based signature schemes with PKCS#1 v1.5 padding (RS256, RS384, RS512) and RSASSA-PSS schemes (PS256, PS384, PS512), which use a public/private key pair. A separate group comprises ECDSA algorithms (ES256, ES384, ES512), based on the P-256, P-384, and P-521 elliptic curves, combining compact parameters with a high level of cryptographic strength. The special value none corresponds to the absence of a digital signature or MAC and is intended only for strictly controlled scenarios.

Further evolution of the standards extended the JOSE algorithm set by introducing Edwards-curve signatures. Based on RFC 8037 [36] and RFC 8032 [35], the use of EdDSA with Ed25519 and Ed448 is defined. For EdDSA, the alg header value is EdDSA, while the specific curve (Ed25519 or Ed448) is specified in the JSON

Web Key (JWK) representation (via the key parameters). Table 3 summarizes the supported algorithms and their implementation requirements. In this work, this set of algorithms constitutes the experimental object for performance evaluation during JWT generation and validation.

Table 3

3) Signature and MAC Algorithms from RFC 7518 and RFC 8032

"alg" value	Digital signature / MAC algorithm	Implementation requirement
HS256	HMAC using SHA-256	Required
HS384	HMAC using SHA-384	Optional
HS512	HMAC using SHA-512	Optional
RS256	RSASSA-PKCS1-v1_5 using SHA-256	Recommended
RS384	RSASSA-PKCS1-v1_5 using SHA-384	Optional
RS512	RSASSA-PKCS1-v1_5 using SHA-512	Optional
ES256	ECDSA using P-256 and SHA-256	Recommended
ES384	ECDSA using P-384 and SHA-384	Optional
ES512	ECDSA using P-521 and SHA-512	Optional
PS256	RSASSA-PSS using SHA-256 and MGF1 with SHA-256	Optional
PS384	RSASSA-PSS using SHA-384 and MGF1 with SHA-384	Optional
PS512	RSASSA-PSS using SHA-512 and MGF1 with SHA-512	Optional
none	No digital signature or MAC	Optional
EdDSA	EdDSA (Ed25519 or Ed448, as specified in JWK) [34]	Recommended

Based on JWA and the RFCs defining Ed25519/Ed448, a generalized model for benchmarking JWT generation and validation is formulated. The model is intentionally implementation-agnostic: it defines an abstract set of entities, parameters, and measurement procedures that can be implemented in any software environment supporting the algorithms in Table 3. It specifies which token and cryptographic

properties must be recorded, in which modes experiments are conducted, and which metrics are collected for subsequent analysis.

The model is grounded in a typical JWT usage scenario in the web-oriented e-commerce system “Shop.” As shown in Figs. 11 – 13, the token structure includes a header with alg and typ, a payload containing a set of claims (user identifier, issuer, audience, issuance time, expiration time, scope, etc.), and a cryptographic signature produced according to the selected algorithm. The model assumes the same logical JWT structure across all experiments; differences are limited to the choice of alg, the key type, and the payload size.

Figure 11. Three main elements of a JWT token

Source of the figure: https://medium.com/@ilham76c/jwt-explained-a-complete-guide-to-its-structure-flow-and-security-considerations-2c9b379308f3

$$\mathrm{WT} = [header].[payload].[signature],$$

where $header = \mathrm{base64url}_{\mathrm{encode}}(′\{"\mathrm{alg}": "\mathrm{HS256}", "\mathrm{typ}": "\mathrm{JWT}"\}′)$

$payload = \mathrm{base64url}_{\mathrm{encode}}(′\{\mathrm{sub}: 131175321, \mathrm{name}: \mathrm{John}\}′)$

$$signature = base64url_{encode}\left(hmac_{sha256}(header + "." + payload, key_{secret})\right)$$

Description of experiment input parameters. The model fixes:

– the set of algorithms under test (alg values from Table 3, including EdDSA with Ed25519/Ed448 keys);

– operation mode: token generation (encode), token validation (verify), or a combined mode (encode→verify);

– key material characteristics: key type (symmetric, RSA, ECDSA, EdDSA) and key length or curve parameters;

– payload parameters: a baseline payload of fixed size and, if required, a set of payloads of different sizes to model “lightweight” and “heavy” tokens;

– the number of iterations per measurement and the number of repeated runs per algorithm/mode.

Description of the test environment. For each test series, the model requires recording platform characteristics: CPU model and clock frequency, number of cores/threads, RAM size, operating system version, runtime/interpreter version (e.g., JVM or Python), and versions of cryptographic libraries implementing JWA and EdDSA. These parameters are not part of the experimental procedure itself, but they are mandatory metadata to ensure comparability and reproducibility.

```
Header: Algorithm & Token Type
{
  "alg": "ES256",
  "typ": "JWT"
}

Payload: Data
{
  "sub": "131175321",
.."iss": "order-service.shop.com",
  "name": "Fuul Name",
  "aud": "gw.shop.com"
  "iat": 1760352633,
  "exp": 1760356233,
.."scope": "gw:auth"
}

Sign JWT: Private Key
{
  "kid": "3d570486-d154-4b15-bffd-5a1560fe5f22",
  "kty": "EC",
  "alg": "ES256"
  "use": "sig",
  "crv": "P-256",
  "x": "2KRDaXgtNJ1AJotZ8qpo_ghUJxIdyU99HhBzBpRPhPo",
  "y": "2ZcTq-CfwuJi1P_Gc2hB2cSazatPG0U71emHwOqvKFA"
  "d": "Bpg9RGkiUHyo5F9HnG9dm2c1fuPZYAcKgX2WjZRWTow",
}
```

Figure 12. Example of a JWT token and ES256 private key (JWK)

eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9
.
eyJzdWIiOiIxMzExNzUzMjEiLCJuYW1lIjoiRGFuaWVsIEt1ZHJpYSIsImFkbWluIjp0cnVlLCJpYXQiOjE3NjAzNTI2MzN9
.
Q923hWL19zStZFYONL_h-EhY9oxxZSuenH3t1WR-6dzLRz3bfMpHcBRZuFCJiKZTxzx_PSM8FWe6xr-WhuOtoQ

Figure 13. Example of a signed and Base64URL-encoded JWT

Procedure for generating test tokens. For each algorithm in Table 3, key material is pre-generated according to the relevant standard: a random secret key for HS256/HS384/HS512; a public/private key pair for RSASSA-PKCS1-v1_5 and RSASSA-PSS; EC keys on P-256/P-384/P-521 for ECDSA; and Ed25519/Ed448 key pairs for EdDSA. For each “algorithm–key type” combination, a baseline JWT header is formed with the corresponding alg value and typ="JWT", together with one or more representative payload variants that reflect real scenarios (user authentication and authorization to access services such as order-service, payment-service, gw.shop.com, etc.).

Execution time measurement scheme. The model assumes high-resolution monotonic timers to record the start and end of an operation series. In encode mode, a new JWT is generated iteratively using the fixed header and payload; fields such as iat and exp may be updated to approximate realistic token issuance behavior. In verify mode, one or more valid tokens are pre-generated and stored; the benchmark loop performs only signature and integrity verification without generating new tokens.

Performance metrics. Each experiment must report:

total execution time for the iteration series for the selected algorithm and mode;

number of operations (iterations), enabling throughput calculation (operations per second);

average time per operation (e.g., in microseconds);

token configuration parameters (algorithm, key type, payload size).

If required, the model allows additional metrics (e.g., memory consumption or CPU utilization), but time-based metrics are treated as baseline.

Result storage structure. All measurements are stored in tabular form (e.g., CSV), where each row corresponds to a single experiment or a single iteration series. Mandatory fields include: algorithm identifier (alg), test mode (encode, verify, encode→verify), iteration count, total execution time, derived metrics (ops/s, average time/op), key characteristics, and a concise description of the test environment. This format aligns measurements with Table 3 and binds each record to a specific JWA/EdDSA algorithm.

Thus, the proposed performance testing model for JWT generation and validation relies on the formal requirements of JWA and related RFCs and defines a unified framework for experiments comparing the algorithms in Table 3. It specifies the algorithm set, operating modes, token parameters, environment metadata, timing procedures, and result storage format without being tied to any particular implementation.

3.2. Justification for Choosing the Flask Web Framework for Modeling a Microservice Architecture

To implement the software model of the microservice system within the scope of the master's thesis, the Flask microframework (Python) was selected. Flask is a lightweight WSGI (Web Server Gateway Interface) web framework that follows a philosophy of minimalism and extensibility. Its architectural core is based on the Werkzeug[1] library (routing and WSGI request handling) and the Jinja2 templating engine. Unlike full-stack frameworks, Flask does not impose a rigid project structure or prescribe specific tools for database access, which makes it particularly suitable for academic modeling where flexibility and component isolation are critical. Table 4 summarizes the advantages and disadvantages of Flask for use in microservice-based systems.

[1] https://werkzeug.palletsprojects.com/en/stable/

Table 4

Characteristics of the Flask Framework in the Context of Microservices

Advantages	Disadvantages
Architectural flexibility. Flask allows the developer to choose design patterns independently. This is critical for modeling microservices because each service may require a distinct set of libraries (e.g., one service uses SQL, another uses NoSQL, and a third is purely computational).	Limited native asynchrony. Although Flask 2.0+ introduced support for async view functions, its asynchronous capabilities are generally less efficient under high I/O workloads than those of frameworks designed around an async runtime (e.g., FastAPI).
Lightweight nature. Minimal startup overhead makes it possible to deploy many microservice models on limited hardware resources (e.g., on a developer's local machine when testing service interactions).	"Empty box" problem. Basic concerns (ORM, data validation, authorization) must be configured manually, which may increase initial setup time if reusable templates are not available.
Low entry barrier. The simplicity of syntax allows focusing on business logic and service interaction algorithms rather than on extensive framework configuration.	
Mature ecosystem. A wide range of extensions (Flask-RESTful, Flask-SQLAlchemy, Flask-Migrate) enables rapid integration of required functionality.	

A comparison with the most common alternatives in the Python ecosystem—Django and FastAPI—is provided in Table 5.

Table 5

4) Comparison of Web Frameworks in the Python Ecosystem

Characteristic	Flask	Django	FastAPI
1	2	3	4
Type	Microframework	Full-stack (monolithic)	Microframework (async)
Architecture	Modular, extensible	"Batteries included"	Asynchronous, based on Starlette

Continuation of Table 5

1	2	3	4
MVP development speed	High (for simple services)	High (for standard CMS/CRUD)	Very high (automatic documentation generation)
Performance	Medium	Lower (higher overhead)	High
Suitability for microservices	High (logic isolation)	Low (excessive coupling)	High

Django is primarily designed for monolithic applications. Using it for a microservice where only an API endpoint is required is often excessive, as it introduces a substantial set of additional components (e.g., an admin interface and a heavyweight ORM), which conflicts with the lightweight principle typical of microservices.

FastAPI is a strong competitor and a leader in performance. However, for modeling purposes, Flask offers advantages in stability and ease of debugging. FastAPI also encourages strict typing (via Pydantic) and assumes familiarity with asynchronous programming, which can complicate rapid prototyping when the researcher is focused on interaction logic rather than high-load optimization.

Given the requirements for flexibility, rapid prototype development, and the need to demonstrate component interaction principles, Flask is an appropriate choice. It provides a balanced combination of functionality and simplicity, allowing the research to focus on architectural modeling rather than on the complexity of the tooling.

3.3. Analysis of Database Technologies for Microservice Architecture and Justification for Selecting the Modeling Toolset

In modern software engineering, microservice architecture entails decentralized data management. According to the Database-per-Service pattern, each microservice must own its own data store, which is accessed exclusively through its API. This ensures loose coupling between components and enables selecting the database technology that best fits the nature of a particular service’s data.

In production environments, this approach is most commonly implemented using robust client–server database management systems (DBMSs), such as:

PostgreSQL and MySQL as a standard choice for transactional data requiring ACID compliance;

MongoDB or Cassandra for unstructured data and high availability in distributed clusters;

Redis for caching and fast access to key–value data.

However, when designing a research environment and modeling microservice interactions, the use of full-scale server-based DBMSs often introduces excessive complexity that is not justified by the research objectives. Deploying a separate PostgreSQL instance or a MongoDB cluster for each modeled service requires substantial computational resources (CPU, RAM), network configuration, containerization, and administration. This shifts the focus from studying algorithms and architectural patterns to addressing infrastructure (DevOps) concerns.

Given these constraints, SQLite was selected as the baseline data management system for this study. The rationale for choosing SQLite for modeling is as follows:

architectural compliance with the isolation principle: SQLite is an embedded, serverless library that stores the database as a regular file on disk. This enables physical enforcement of data isolation for each microservice: each service operates on its own .db file. This directly emulates the Database-per-Service pattern without the need to maintain dozens of network connections [37];

lightweight nature and performance: SQLite does not require a separate server process. Read and write operations are performed via direct library calls, minimizing overhead from inter-process communication (IPC) and eliminating network latency. This allows complex multi-service interaction scenarios to be executed and tested even on a researcher’s local machine with limited resources;

full-featured sql support: Despite its compactness, SQLite supports most of the SQL-92 standard, including ACID transactions, triggers, complex queries, and indexes. This enables development of business logic that remains syntactically and logically compatible with “larger” DBMSs (e.g., PostgreSQL). Code written against SQLite via an ORM (e.g., SQLAlchemy) can later be migrated to a production DBMS without architectural changes;

simplified experiment reproducibility: Because the entire database is contained in a single file, creating backups, resetting the system state, or transferring test datasets for reproducing experimental results reduces to copying the file. This is essential for research, where reproducibility is a key requirement;

support for modern capabilities: Recent SQLite versions, particularly with Write-Ahead Logging (WAL), provide improved read concurrency, which is sufficient for load simulation in a modeling setting. In addition, extensions (e.g., sqlite-vec) enable experimentation with vector search for AI-related components.

Thus, selecting SQLite represents a strategically justified compromise: it enables a "clean" microservice architecture with strict data isolation while avoiding infrastructure overhead. This allows the research to focus on service interaction logic, data consistency, and process orchestration, which are central to this master's thesis.

3.4. Selection of a JWT Token Support Library for the Python Platform

To implement authentication and authorization in the microservice system, the PyJWT library (jpadilla/pyjwt) was selected as a widely adopted JWT implementation for Python. This choice is driven by the need for rapid prototyping and microservice-architecture modeling within the master's thesis, where ease of integration, the availability of documentation and examples, and compliance with RFC 7519 [32] are critical factors.

PyJWT is an open-source library that supports encoding and decoding JWTs in accordance with RFC 7519. It provides practical support for the main JOSE/JWA signing families used with JWT in real systems: symmetric HMAC (HS256/HS384/HS512), RSA signatures (RS* and PS*), elliptic-curve signatures (ES*), and EdDSA (with Ed25519/Ed448 key types, depending on the cryptographic backend). This breadth allows selecting a security–performance trade-off appropriate to the experimental objectives.

From a maintenance and adoption perspective, PyJWT is actively released on PyPI (e.g., 2.10.1 published in late November 2024, and 2.11.0 published on January 30, 2026) and has a large public user base (deps.dev reports ~22k public dependents, which is a reasonable proxy for ecosystem adoption). (PyPI) On GitHub, the repository

is also among the most starred JWT libraries in the Python ecosystem (≈5.6k stars in GitHub) The minimalist API (jwt.encode(), jwt.decode()) and straightforward installation via pip further support rapid prototyping [38].

A comparative analysis with alternative Python libraries – python-jose, jwcrypto, and Authlib – is provided in Table 6. The star counts below reflect GitHub repository metadata as of February 2026.

Table 6

5) Characteristics of JWT Libraries on the Python Platform

Feature	jpadilla/pyjwt[2]	mpdavis/python-jose[3]	latchset/jwcrypto[4]	lepture/authlib[5]
GitHub stars	~5.6k	~1.7k	~473	~5.2k
Sign	+	+	+	+
Verify	+	+	+	+
`iss`	+	+	+	+
`sub` (built-in validation)	limited / app-level	+	+	+
`aud`	+	+	+	+
`exp`	+	+	+	+
`nbf`	+	+	+	+
`iat`	+	+	+	+
`jti` (replay semantics)	app-level	+	+	+
typ	implementation-dependent	implementation-dependent	+	implementation-dependent
HS256/384/512	+	+	+	+
RS256/384/512	+	+	+	+
ES256/256K/384/512	+	+	+	+
PS256/384/512	+	limited	+	+
EdDSA	+	unclear/limited	+	+

[2] https://github.com/jpadilla/pyjwt/
[3] https://github.com/mpdavis/python-jose/
[4] https://github.com/latchset/jwcrypto/
[5] https://github.com/lepture/authlib

Interpretation of the comparison. python-jose provides broad functionality and aims at a more complete JOSE scope (JWS/JWE/JWK/JWA), which can be excessive when the research goal is specifically JWT issuance and verification rather than full JOSE coverage. jwcrypto is a cryptography-focused JOSE implementation (including JWE), but its design emphasis on completeness can increase complexity for a JWT-centric prototype. Authlib is primarily an OAuth 2.0 / OIDC framework; while it includes JOSE/JWT features, its lifecycle considerations matter: Authlib documentation explicitly states that authlib.jose is being deprecated in favor of joserfc.

Overall, PyJWT offers the most favorable functionality-to-complexity ratio for the thesis objectives: it is simple to integrate, sufficiently feature-complete for JWT-centric microservice experiments, actively released, and widely adopted. (GitHub) Potential gaps in strict claim semantics (e.g., project-specific `sub` constraints or jti replay protection) are typically handled at the application layer and are not critical for modeling baseline JWT authentication scenarios.

3.5. Containerized Deployment of Microservices. Docker

A production environment must provide four key capabilities:

service management interface – enables developers to create, update, and configure services. Ideally, this interface is exposed as a REST API invoked via command-line tools and GUI deployment tools;

runtime service management – ensures continuous operation of the required number of service instances. If a service instance fails or cannot handle requests, the production environment must restart it. If a host machine fails, the environment must restart the affected service instances on another machine;

monitoring – provides developers with visibility into service behavior, including logs and metrics. If issues occur, the production environment must notify developers.

request routing – routes user requests to the appropriate services.

Based on [39], the choice of Docker as the baseline containerization technology for implementing a microservice architecture is justified primarily by the fact that microservices increase the number of independent components and, consequently, the number of environments, dependencies, and integration points that must be managed.

In team development, this complexity grows rapidly: differences in operating systems (Windows/macOS/Linux), language and library versions, and configurations across local machines, test servers, and production environments create the classic “works locally, but fails elsewhere” risk. Docker mitigates this risk by encapsulating each microservice in a container with fixed dependencies and configuration, ensuring runtime reproducibility and reducing conflicts between development and server environments. For microservices, the key benefits are OS-level isolation, standardized runtime conditions, advantages over traditional virtualization, and support for subsequent orchestration, which is a baseline requirement for operating distributed systems.

Docker Compose provides a flexible interface for managing multi-container applications through a declarative approach to describing and orchestrating distributed services. It offers a comprehensive set of commands for full lifecycle management of containers.

Main Docker Compose commands include:

docker-compose up – creates and starts all services defined in the configuration file;

docker-compose down – stops and removes containers, networks, and volumes;

docker-compose start / docker-compose stop – manages the state of services without deleting resources;

docker-compose ps – displays the status of all containers in the project;

docker-compose logs – provides access to aggregated logs of all services;

docker-compose exec – executes commands in running containers;

docker-compose build – rebuilds service images;

docker-compose pull – downloads the latest image versions from the registry;

docker-compose config – validates and checks the configuration file.

The commands support modular options for flexible configuration, such as:

-f – select an alternative configuration file;

--env-file – load environment variables;

--scale – dynamically scale specific services.

This interface integrates automation mechanisms for sequential deployment of services while accounting for inter-service dependencies, which significantly simplifies managing complex microservice architectures during development and testing.

4. Modeling of the theoretical research results achieved.

4.1. Implementation of a JWT Token-Based Authentication Model for the Web-Oriented Microservice E-Commerce System “Shop”

The Basket Service interacts with the Inventory Service and the Account Service. To place an order, it first retrieves detailed product information and then checks the available balance by contacting the Account Service. The service APIs are presented in Tables 6 – 11.

Table 6

6) Account Service REST API

HTTP Method	Endpoint	Input Data	Function
GET	/	–	Returns a list of all service endpoints.
GET	/accounts	userID (optional query parameter)	Retrieves all accounts or accounts for a specific user.
POST	/accounts	{"userID": int}	Creates a new account for a user with an initial balance.
GET	/accounts/<acc_num>	acc_num (path parameter)	Retrieves an account by number.
DELETE	/accounts/<acc_num>	acc_num (path parameter)	Closes (deletes) an account.
POST	/accounts/<acc_num>	{"amount": int}	Changes the account balance by the specified amount.
POST	/accounts/user/<user_id>/update_balance	{"amount": int}	Updates the user’s account balance by userID.
GET	/accounts/user/<user_id>	user_id (path parameter)	Retrieves the user’s account balance.

Table 7

7) User Service REST API

HTTP Method	Endpoint	Input Data	Function
GET	/	–	Returns a list of all service endpoints.
GET	/users	username (optional query parameter)	Retrieves all users or a specific user by username.
POST	/users	{"username": str, "pwd": str}	Registers a new user.
DELETE	/users/userID/<userID>	userID (path parameter)	Deletes a user by ID.
POST	/users/login	{"username": str, "pwd": str}	Authenticates a user (returns a token).

Table 8

8) Product Service REST API

HTTP Method	Endpoint	Input Data	Function
GET	/	–	Returns a list of all service endpoints.
GET	/products	–	Retrieves all products.
POST	/products	{"name": str, "price": float, "stock": int}	Adds a new product.
GET	/products/<product_id>	product_id (path parameter)	Retrieves product information.
PUT	/products/<product_id>/updatestock	{"stock": int}	Updates product stock quantity.

Table 9

9) Basket Service REST API

HTTP Method	Endpoint	Input Data	Function
GET	/	–	Returns a list of all service endpoints.
GET	/basket	–	Retrieves all baskets for all users.
POST	/basket/<user_id>/<product_id>	{"quantity": int}	Adds a product to the user's basket.
DELETE	/basket/<user_id>/<product_id>	–	Removes a product from the user's basket.
GET	/basket/<user_id>	–	Retrieves the user's basket.
POST	/basket/checkout/<user_id>	–	Places an order (debits balance, updates inventory).

Table 10

10) Gateway Service REST API

HTTP Method	Endpoint	Input Data	Function
GET	/	–	Returns a list of all service endpoints.
GET	/users	username (optional query parameter)	Retrieves user information.
POST	/users	{"username": str, "pwd": str}	Registers a user.
GET	/users/<userID>/accounts	–	Retrieves user accounts.
POST	/users/<userID>/accounts	–	Opens a new account for the user.

Continuation of table 10

GET	/users/<userID>/accounts/<accNum>/transactions	–	Retrieves account transactions.
POST	/accounts/<userID>	–	Retrieves the user's account balance.
POST	/cart/checkout/{user_id}	{"cart": object}	Places an order via the basket.
POST	/login	{"username": str, "pwd": str}	Authenticates a user.
POST	/logout	–	Logs out a user (not implemented).

Table 11

11) STS (Security Token Service) REST API

HTTP Method	Endpoint	Input Data	Function
GET	/	–	Returns a list of all service endpoints.
POST	/login	{"username": str}	Issues a JWT for a user (without password verification).

The system is implemented in Python 3.13. All HTTP methods are exposed via REST APIs implemented using Flask. The number of worker processes per service can be configured. The Account and User services maintain separate SQLite databases, which keeps services loosely coupled. To simplify the experimental setup, each microservice is deployed as a single instance and can run in isolation within a container.

The experiments were conducted on a Windows workstation using Docker Desktop (Linux-based containers). The host system configuration was: Windows 10; Intel Core i7-6700 CPU @ 3.40 GHz (4 cores, 8 logical processors); 16 GB RAM.

The C4 component diagram (see Fig. 14) illustrates the deployment and communication protocols among the Shop services.

The implementation supports multiple users operating either concurrently or sequentially. When the test client increases the request rate, the application uses multi-process execution. Each service uses its own database and resources. The Shop e-commerce system is implemented as a set of microservices and follows a composite architectural model.

The test client registers two users and opens accounts for them. Initial balances are then credited sequentially. After that, load testing is performed for 10, 100, 500, 1,000, 2,500, and 10,000 requests. These requests are sent to both users either concurrently (in parallel) or sequentially to execute checkout operations. For each user, a basket containing a predefined number of items is created, after which a series of checkout operations is performed.

For each load configuration, the experiment is conducted in two modes: (1) without security mechanisms enabled and (2) with JWT-based authorization. To evaluate system performance on the client side, the average execution time of checkout requests is calculated. In addition, total response time and throughput of checkout operations across all requests are recorded. The checkout operation involves interaction among four microservices (see Fig. 3.1). The duration of a single operation is affected by network latency and processing time inside the microservices, including database access time. In the specified configuration, successful execution is possible only if the user’s account has sufficient funds; otherwise, the operation is not performed.

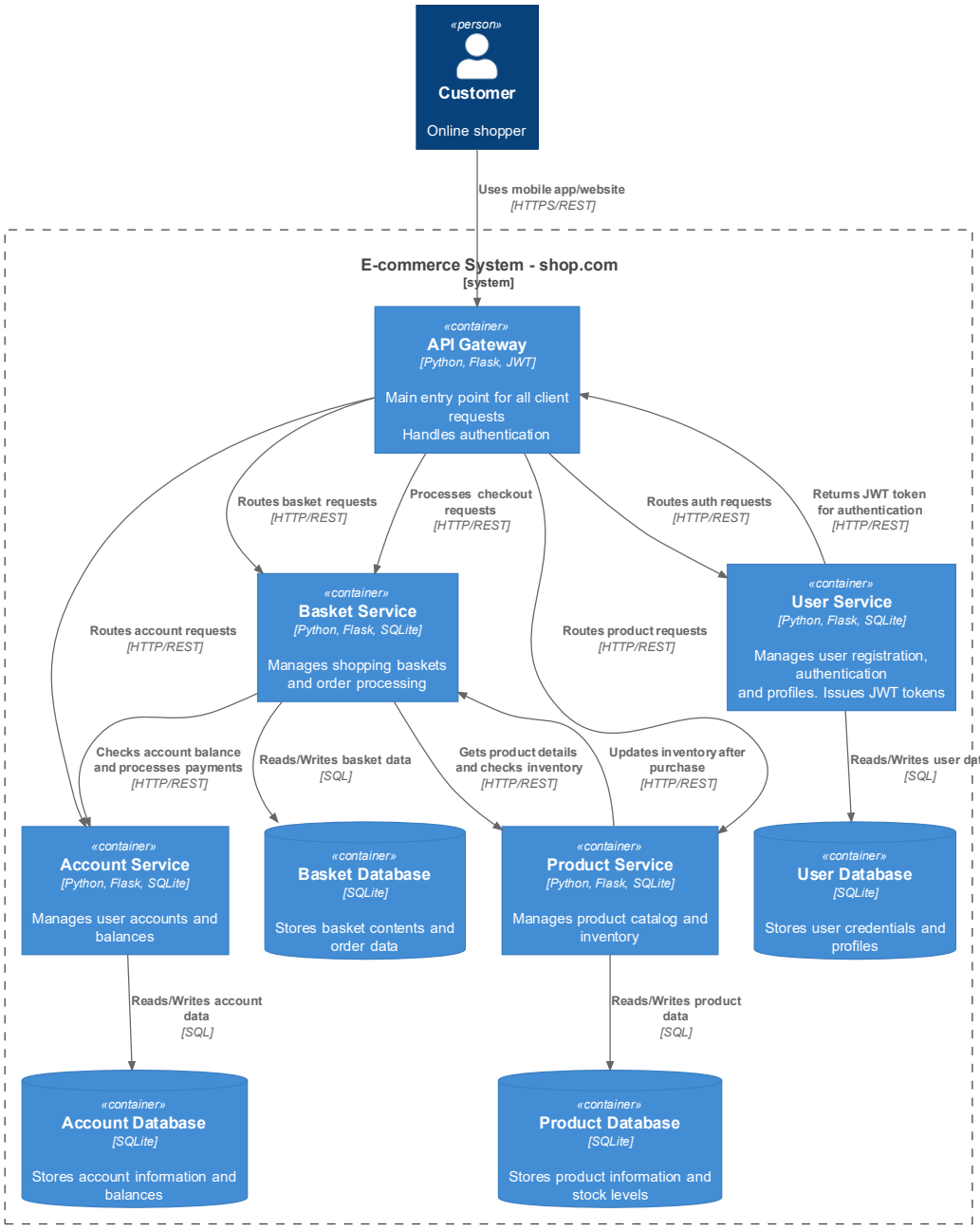


Figure 14. Component diagram of service deployment and communication in the Shop model

Source of the figure: original

4.2. Implementation of an Environment Model for Testing JWT Performance Across Cryptographic Algorithms

An environment model was implemented to evaluate the performance of JWT token generation (encode) and validation (verify) operations for a set of cryptographic algorithms from the HMAC, RSA, RSASSA-PSS, ECDSA, and EdDSA families. A

simulation–statistical approach was adopted: for each algorithm, a series of identical operations with a fixed number of iterations is executed, and aggregate indicators are obtained by averaging results across repeated runs.

The baseline assumptions are that testing is performed in an isolated containerized environment with fixed resource constraints, background system load is negligible, and the impact of network latency and input/output operations is minimized. A limitation of the model is that it reproduces a typical, but not exhaustive, range of industrial web-application deployment scenarios.

he experimental platform is based on Docker Desktop running on Windows 10 on a workstation equipped with an Intel® Core™ i7-6700 CPU @ 3.40 GHz (4 physical cores, 8 logical threads) and 16 GB of RAM. To enhance reproducibility, the Docker container running the load-testing module was strictly constrained to 1 GB RAM and 1 virtual CPU. This configuration simulates the deployment of an individual microservice under resource constraints.

Input data preparation includes generating key material (secrets for HMAC; public/private key pairs for RSA/RSASSA-PSS and ECDSA; and key pairs for EdDSA), and forming a unified JWT header and payload that emulate typical authentication and authorization claims. For each “algorithm–mode (encode/verify)” combination, the number of iterations, payload size, and key parameters are specified.

The model is implemented in Python using the PyJWT library with a cryptographic backend based on the cryptography library. The module employs a high-resolution monotonic timer to measure total execution time, calculates average per-operation latency and throughput (operations per second), and automatically stores results in CSV/JSON formats.

4.3. Description of modeling results.

4.4. Load Modeling on the Authentication Subsystem with JWT Tokens

The experimental study aimed to assess the impact of JWT tokens on the operation of the authentication service in a web-oriented system. For comparison, two resource-access modes were considered:

- sequential requests without security mechanisms enabled;

– sequential requests with JWT token validation enabled.

The load was generated by two users; however, the implementation supports scaling the number of clients. Before the experiments, baseline system parameters were recorded: CPU utilization was approximately 0.03%, and each microservice operated in a Docker container limited to 1024 MB of RAM and one CPU core. This made it possible to study system behavior under controlled resource conditions.

Tables 12 and 13 present the measured total response time for different numbers of sequential requests, as well as the calculated throughput in transactions per second (TPS). In the mode without security functions enabled, 10 requests resulted in a total response time of 0.04 s with a throughput of 250 TPS; as the number of requests increased to 10,000, the total response time rose to 27.23 s and throughput to 367 TPS. In the mode with JWT validation enabled, 10 requests produced 0.043 s and 233 TPS, while 10,000 requests yielded 30.27 s and 330 TPS, respectively. Thus, across all experimental sets, the security-enabled configuration demonstrates a slightly higher total processing time and a slightly lower throughput.

The measurement results for the mode without safety functions enabled are shown in Table 12.

Table 12

Sequential requests without security mechanisms enabled

Total number of requests	Total response time (s)	Throughput (TPS)
10	0.04	250
50	0.156	321
100	0.34	294
500	1.73	289
1,000	2.61	383
2,500	6.63	377
10,000	27.23	367

Similar results for the mode with the JWT security feature enabled are shown in Table 13.

Table 13.

12) Sequential requests with JWT validation enabled

Total number of requests	Total response time (s)	Throughput (TPS)
10	0.043	233
50	0.196	255
100	0.383	261
500	1.92	260
1,000	3.078	325
2,500	7.89	317
10,000	30.27	330

The relationship between the total response time and the number of consecutive requests in both modes is shown in Fig. 15.

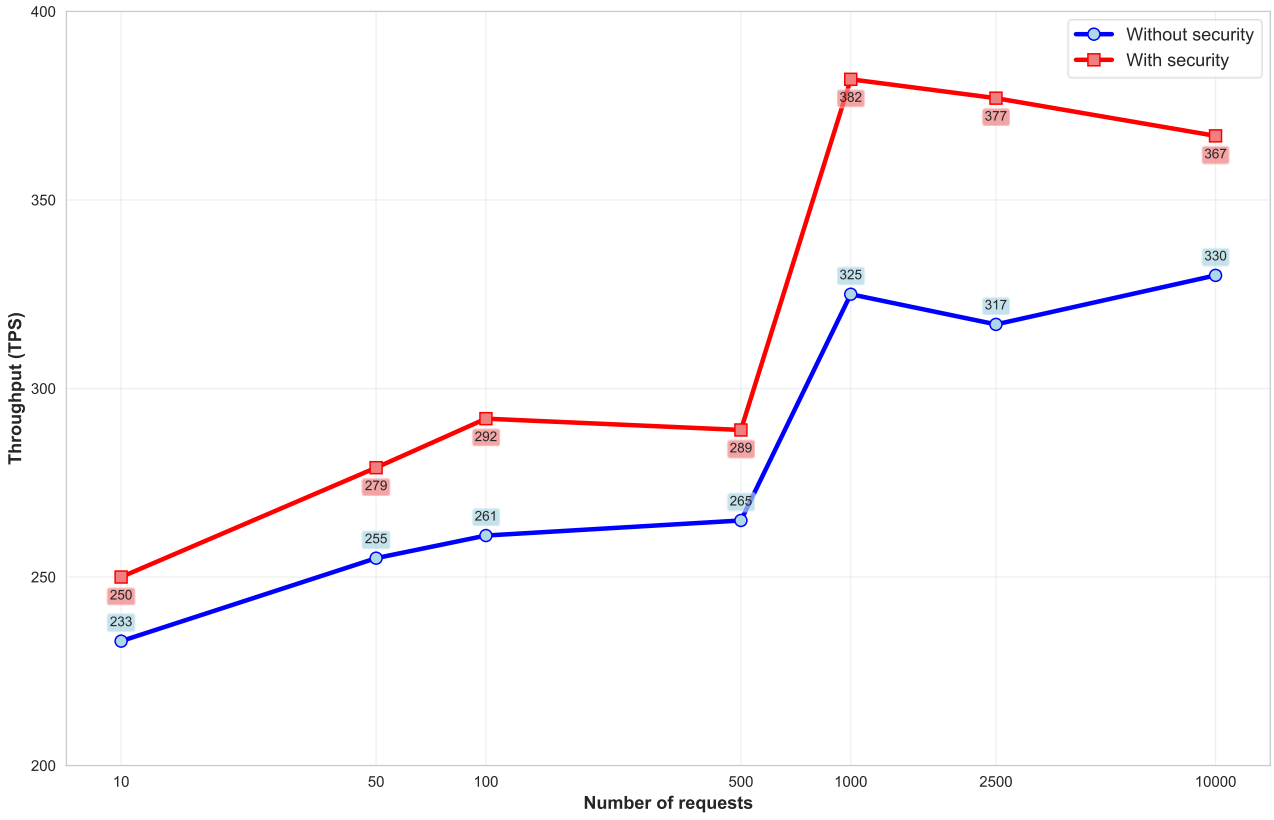


Figure 15. Total response time versus the number of sequential requests (with and without JWT validation)

Source of the figure: original

Both curves exhibit a pronounced nonlinear pattern. Under low load (10–100 requests), response time increases slowly, remaining below one second, and the difference between the secured and unsecured modes is negligible. Starting from 500 requests, growth becomes steeper: for 1,000 requests, the time with JWT validation exceeds three seconds, whereas without security it is approximately 2.6 s. At 10,000 requests, the delay is the largest, and the difference between modes reaches several seconds. The data indicate additional overhead from JWT validation, averaging approximately 10–20% of the total request-processing time. At the same time, the absolute magnitude of this difference remains moderate even under maximum load.

The comparative throughput (TPS) for identical experimental sets is shown in Fig. 16.

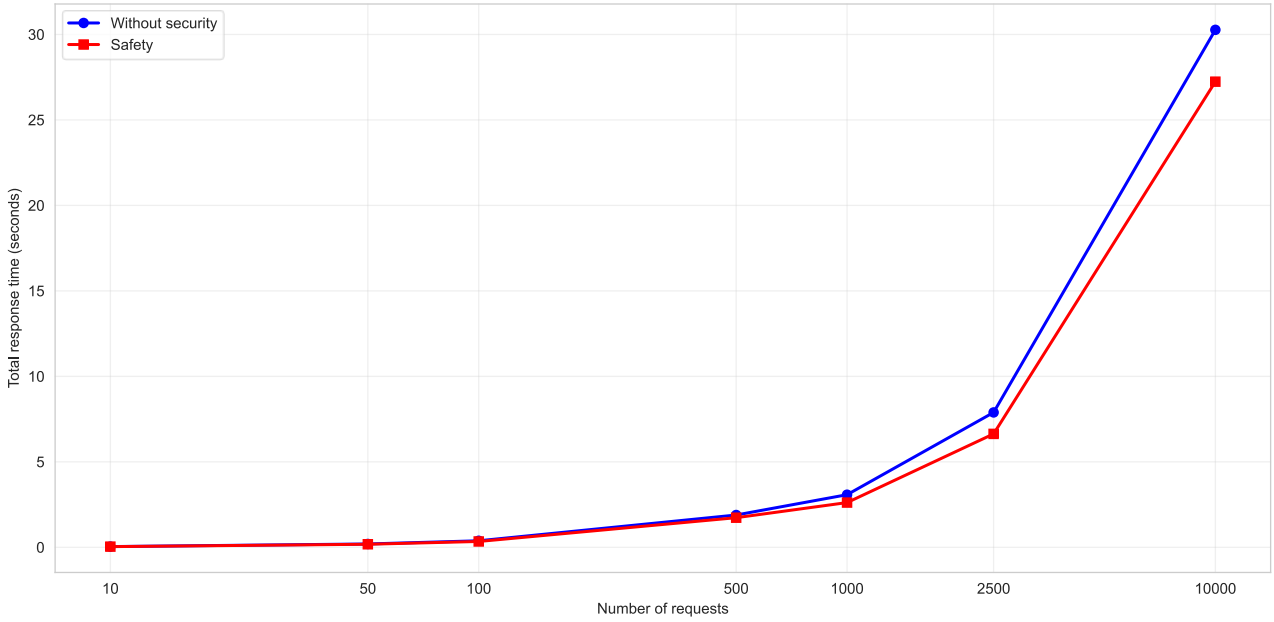


Figure 16. Throughput (TPS) comparison (with and without JWT validation)

Source of the figure: original

The curves for the secured and unsecured modes are close, confirming the limited impact of JWT validation on peak processing speed. In the initial range (10–100 requests), TPS increases moderately in both modes due to more efficient CPU utilization as the number of transactions grows. For 500–1,000 requests, throughput

reaches its maximum: above 380 TPS without security and above 320 TPS with JWT. Increasing the number of requests to 2,500 and 10,000 does not cause a substantial drop in TPS, indicating the absence of obvious bottlenecks in the implemented architecture. The relative throughput decrease in the JWT-enabled mode compared to unrestricted access does not exceed approximately 10–15%.

In parallel with response time and throughput, CPU load and memory usage were measured. As the number of requests increased, CPU utilization rose as expected, approaching upper bounds at 2,500–10,000 requests. However, values remained within a range that ensured stable service operation, and switching from the unsecured mode to JWT validation did not produce a multiplicative increase in CPU load. Memory consumption changed only slightly in both configurations, as request/response payload sizes were fixed and token lifetimes were relatively short; this enabled experiments to be conducted in containers with strict memory limits.

In summary, using JWT tokens to secure sequential requests leads to a moderate increase in response time and a slight reduction in throughput, but does not impose critical CPU or memory load even for tens of thousands of requests. The observed overhead is acceptable for most web applications where authentication and data integrity are prioritized. Therefore, JWT-based access control can be considered a balanced solution between system performance and security requirements.

4.5. Results of Modeling JWT Token Performance Across Cryptographic Algorithms

This section presents the results of performance modeling for JWT token generation (encode) and validation (verify) operations for the selected cryptographic algorithms HS256, ES256, EdDSA (Ed25519), RS256, and PS256. The experiment was conducted as a series of 50 trials (runs) for each algorithm; within each trial, 100 iterations of token creation and signature verification were performed, enabling statistically stable averaged indicators. In parallel with timing characteristics, CPU utilization and RAM consumption of the Docker container hosting the performance testing module were recorded; during the tests, the allocated virtual CPU core was fully utilized.

Testing was performed under a workload of 50 trials × 100 iterations per trial for the main algorithms ES256, EdDSA, HS256, PS256, and RS256. Under these conditions, container memory consumption and CPU usage approached their maximum values. The results are presented in Tables 14 – 17, and their visualizations are shown in Figs. 17 – 19.

Table 14

13) Average JWT encoding time for signing algorithms HS256, ES256, EdDSA, PS256, and RS256 (50 experiments, 100 tests each)

№	ES256	EdDSA	HS256	PS256	RS256	№	ES256	EdDSA	HS256	PS256	RS256
1	2	3	4	5	6	1	2	3	4	5	6
1	0.06025	0.10396	0.05337	0.64549	0.57444	26	0.06179	0.06947	0.03144	0.66356	0.60384
2	0.06926	0.08088	0.04444	0.62325	0.61790	27	0.05479	0.06068	0.03282	0.70024	0.60813
3	0.07084	0.09522	0.04027	0.62870	0.63756	28	0.06254	0.05646	0.03179	0.63741	0.63457
4	0.06798	0.10038	0.03895	0.62532	0.55204	29	0.06851	0.05643	0.03086	0.56239	0.61937
5	0.07661	0.08069	0.03640	0.56110	0.55926	30	0.07124	0.06792	0.03167	0.59093	0.56166
6	0.06464	0.07918	0.03676	0.59891	0.56766	31	0.07266	0.06835	0.03329	0.70063	0.55917
7	0.05910	0.09964	0.03400	0.61242	0.62927	32	0.06468	0.06986	0.03341	0.64476	0.86553
8	0.06448	0.08379	0.03313	0.56697	0.66046	33	0.05852	0.08392	0.03276	0.65107	1.19382
9	0.06591	0.08028	0.03378	0.61122	0.57786	34	0.07085	0.07161	0.03162	0.58054	0.83847
10	0.06248	0.07494	0.03162	0.59768	0.59678	35	0.06747	0.06834	0.03266	0.57925	0.93222
11	0.07958	0.07471	0.03257	0.62826	0.60186	36	0.07036	0.08095	0.03212	0.67512	1.21455
12	0.06423	0.07977	0.03322	0.68066	0.60077	37	0.06083	0.06684	0.03269	0.62844	0.92167
13	0.05776	0.07547	0.03087	0.56481	0.57232	38	0.06622	0.08865	0.03227	0.60600	0.70097
14	0.05829	0.07032	0.03276	0.65926	0.56853	39	0.06192	0.10305	0.03201	0.65298	0.64792
15	0.05864	0.07049	0.03052	0.57739	0.55452	40	0.06585	0.06544	0.03233	0.69768	0.63142
16	0.05747	0.07433	0.02862	0.58888	0.61042	41	0.06497	0.07059	0.03145	0.62680	0.60631
17	0.05928	0.06971	0.03628	0.57806	0.63676	42	0.06603	0.07100	0.03163	0.62369	0.58624
18	0.08611	0.06861	0.03312	0.58610	0.59658	43	0.06320	0.06162	0.02880	0.57408	0.66438
19	0.06263	0.06792	0.03456	0.60972	0.71413	44	0.06334	0.05956	0.03069	0.56715	0.57567
20	0.07182	0.06401	0.03025	0.57973	0.62363	45	0.06928	0.07008	0.03037	0.65034	0.79267
21	0.06666	0.06433	0.02934	0.61542	0.74352	46	0.07535	0.06761	0.03134	0.60824	0.57069
22	0.05320	0.06512	0.03070	0.64058	0.60483	47	0.07390	0.09140	0.03127	0.56374	0.56861
23	0.06391	0.06588	0.03179	0.62099	0.64152	48	0.07328	0.06387	0.02966	0.59938	0.60291
24	0.05202	0.06708	0.03229	0.66363	0.62917	49	0.06752	0.06184	0.03132	0.62383	0.55654
25	0.05731	0.06730	0.03225	0.64811	0.68206	50	0.06881	0.06370	0.03145	0.59300	0.54738

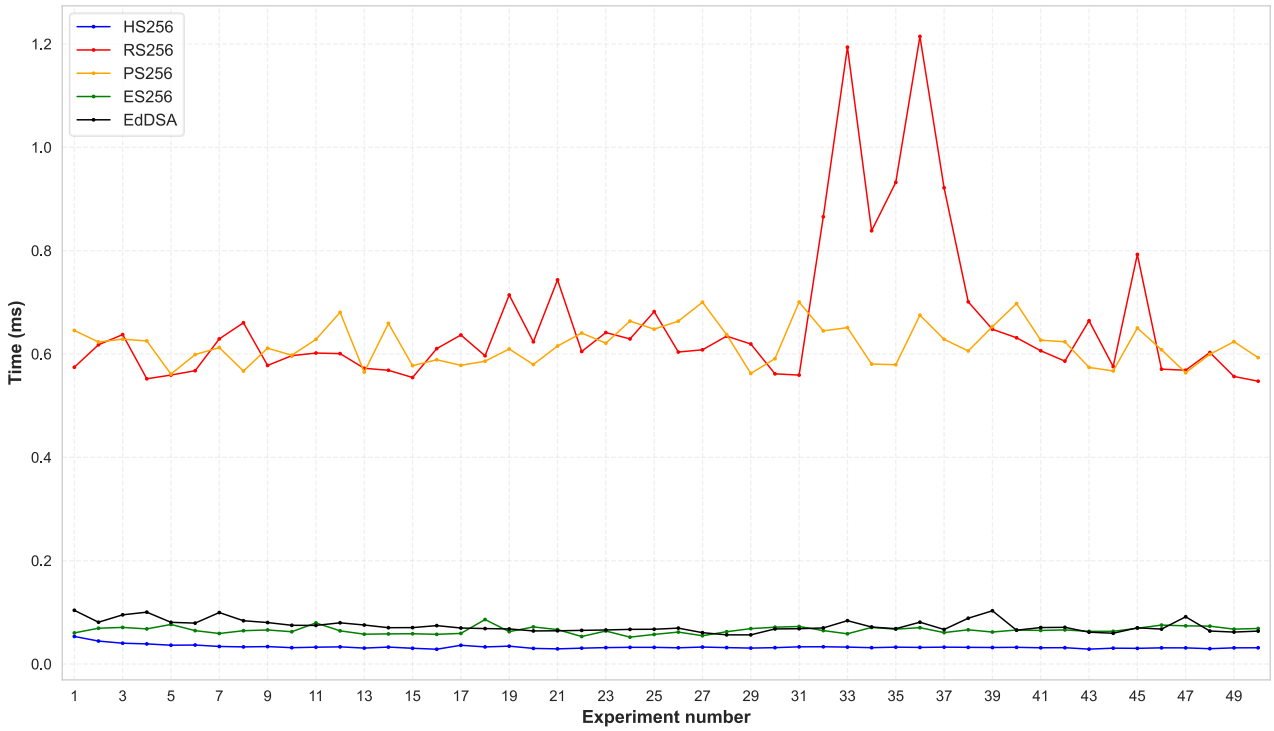


Figure 17. Comparison of JWT signing algorithms by encoding time (encode)

Source of the figure: original

Within the experiment, the execution time of the JWT encoding (signing) operation was measured for five algorithms: HS256, ES256, EdDSA (Ed25519), and the RSA-based algorithms RS256 and PS256. For each mode, a series of 50 runs was conducted with a fixed payload size and identical hardware parameters. The table reports the average time per operation (microseconds), and the corresponding plot is provided in milliseconds to enable a visual comparison of the selected algorithms.

The results demonstrate a clear performance hierarchy. The symmetric algorithm HS256 exhibits the shortest encoding time: average values are approximately 0.03 – 0.04 ms per operation, with only minor variation across runs. Elliptic-curve algorithms ES256 and EdDSA are moderately slower: typical values for ES256 are around 0.06 – 0.07 ms, and for EdDSA around 0.07 – 0.09 ms per operation. Thus, ES256 and EdDSA are roughly 2 – 3× slower than HS256, but remain in the sub-millisecond range, which is acceptable for high-load systems.

The lowest encoding performance is observed for RS256 and PS256. Their average encoding time typically falls within 0.55 – 0.70 ms, and in some runs exceeds 0.8 – 1.0 ms, i.e., approximately 10 – 20× slower than HS256 and several times slower than ES256 and EdDSA. PS256 is generally slightly slower than RS256 due to the higher computational complexity of the PSS padding scheme.

Overall, the plot reflects a stable efficiency ordering: HS256 is the fastest, ES256 and EdDSA form an intermediate group, while RS256 and PS256 lag substantially and form the highest-latency group. Therefore, when latency and throughput are critical, HS256 is preferable (provided a symmetric key model is acceptable) or ES256/EdDSA can be used as a compromise between performance and cryptographic strength. RS256/PS256 are more appropriate primarily for compatibility requirements rather than for maximum speed.

Table 15

14) Results of measuring JWT encoding throughput for algorithms HS256, ES256, EdDSA, RS256, and PS256 (50 experiments, 100 tests each)

№	ES256	EdDSA	HS256	PS256	RS256	№	ES256	EdDSA	HS256	PS256	RS256
1	16598.25	9618.84	18736.06	1549.21	1740.81	26	16184.15	14393.81	31805.97	1507.03	1656.06
2	14437.98	12364.55	22500.35	1604.49	1618.38	27	18250.37	16480.16	30473.84	1428.08	1644.39
3	14115.93	10502.48	24834.01	1590.59	1568.47	28	15989.08	17711.44	31456.40	1568.86	1575.88
4	14709.38	9962.27	25677.10	1599.17	1811.46	29	14596.46	17721.63	32407.98	1778.13	1614.54
5	13052.72	12392.78	27471.70	1782.20	1788.06	30	14036.26	14722.33	31578.95	1692.26	1780.42
6	15469.10	12630.20	27201.18	1669.69	1761.61	31	13762.65	14631.37	30035.91	1427.29	1788.35
7	16921.58	10035.94	29410.09	1632.87	1589.14	32	15461.15	14314.60	29932.65	1550.95	1155.37
8	15508.28	11934.49	30184.24	1763.77	1514.10	33	17088.93	11915.46	30523.45	1535.92	837.65
9	15172.00	12456.70	29601.61	1636.07	1730.53	34	14114.08	13963.76	31626.50	1722.53	1192.65
10	16004.37	13343.87	31624.40	1673.14	1675.67	35	14820.97	14633.53	30622.69	1726.38	1072.71
11	12566.74	13385.07	30699.64	1591.69	1661.53	36	14211.85	12353.19	31137.92	1481.22	823.35
12	15568.26	12536.55	30099.86	1469.16	1664.53	37	16440.54	14961.21	30586.63	1591.25	1084.98
13	17313.26	13251.03	32391.64	1770.50	1747.29	38	15101.34	11280.31	30992.08	1650.16	1426.59
14	17155.40	14221.31	30525.21	1516.86	1758.92	39	16148.60	9703.62	31242.42	1531.43	1543.40
15	17054.25	14186.59	32762.79	1731.93	1803.38	40	15185.88	15281.41	30935.75	1433.32	1583.74
16	17400.73	13453.16	34941.08	1698.15	1638.23	41	15392.35	14167.25	31796.05	1595.41	1649.32

Continuation of table 15

17	16869.70	14345.31	27560.77	1729.93	1570.46	42	15145.44	14084.62	31616.80	1603.35	1705.77
18	11613.52	14574.48	30191.19	1706.20	1676.23	43	15822.76	16227.85	34726.86	1741.91	1505.16
19	15966.22	14723.49	28932.70	1640.09	1400.30	44	15788.80	16788.79	32588.71	1763.19	1737.11
20	13924.08	15623.03	33062.53	1724.94	1603.53	45	14435.02	14270.41	32922.36	1537.66	1261.56
21	15001.18	15544.12	34087.02	1624.92	1344.95	46	13272.21	14791.79	31911.90	1644.10	1752.28
22	18796.15	15357.43	32570.37	1561.09	1653.37	47	13531.76	10941.44	31978.12	1773.87	1758.68
23	15646.32	15178.04	31453.35	1610.34	1558.81	48	13645.45	15656.51	33711.43	1668.38	1658.62
24	19224.75	14907.47	30969.85	1506.87	1589.40	49	14811.26	16170.77	31930.62	1602.99	1796.83
25	17448.32	14859.44	31009.31	1542.94	1466.16	50	14531.72	15697.87	31799.63	1686.35	1826.87

The experimentally measured throughput during JWT encoding (signature) for five algorithms: HS256, ES256, EdDSA (Ed25519), RS256, and PS256 is shown in Fig. 17. The x-axis represents the algorithm, and the y-axis represents the average number of encoding operations per second. Because the payload sizes and hardware configuration were fixed across runs, the figure primarily reflects differences in cryptographic overhead.

The highest throughput is achieved by HS256, providing approximately 30,000 encoding operations per second (roughly 19,000 – 35,000 ops/s). ES256 and EdDSA form an intermediate group: ES256 achieves about 15,000 ops/s, and EdDSA about 13,000 – 14,000 ops/s, i.e., 2–2.5× lower than HS256, but substantially higher than RSA-based schemes.

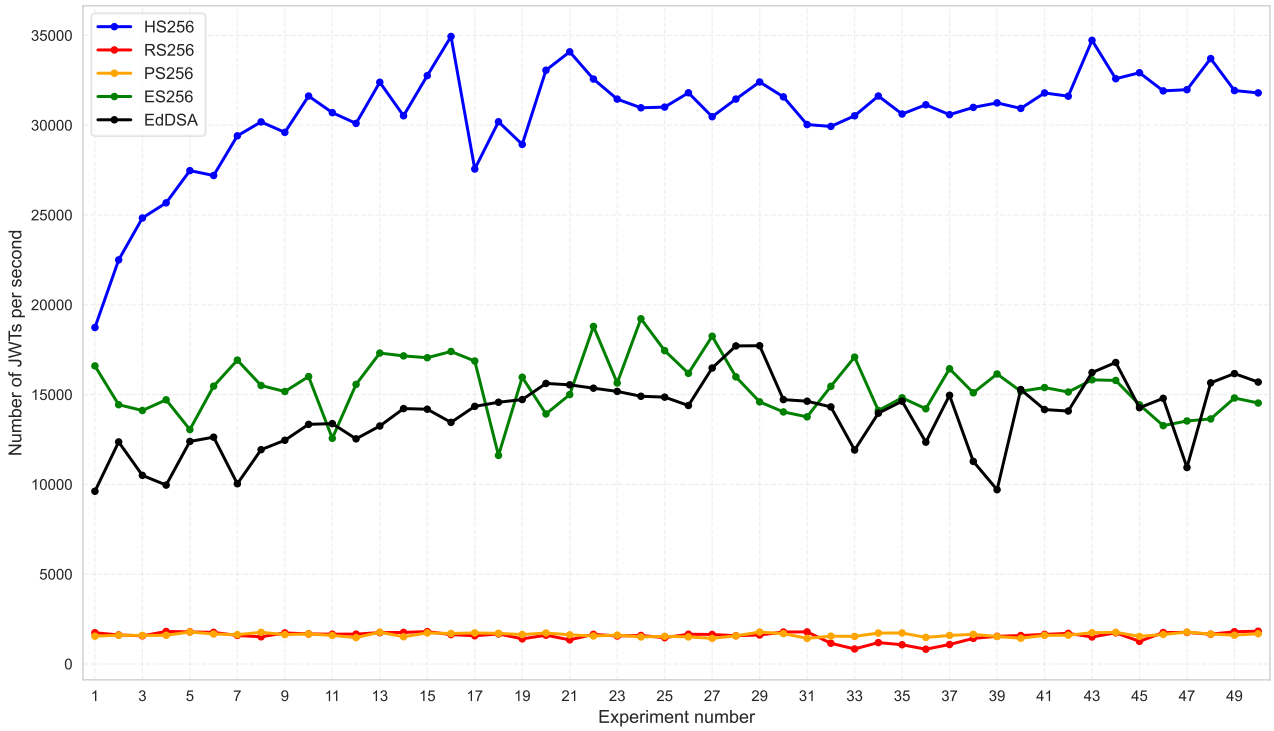


Figure 18. Comparison of the throughput of JWT signature algorithms by the number of encoding operations per second

Source of the figure: original

The lowest throughput is observed for RS256 and PS256 (RSA-2048): average values are approximately 1,500 – 1,600 ops/s, and in some runs RS256 drops to about 800 ops/s. Thus, switching from HS256 to RS256/PS256 reduces signing capacity by roughly 15–20×. Consequently, HS256 is the clear leader by the “operations per second” criterion; ES256 and EdDSA provide a practical compromise between performance and security; RS256 and PS256 are justified mainly when compatibility with existing infrastructure is prioritized over maximum throughput.

Table 16

15) Results of measuring JWT signature verification time for algorithms HS256, ES256, EdDSA, PS256, and RS256 (50 experiments, 100 tests each)

	ES256	EdDSA	HS256	PS256	RS256		ES256	EdDSA	HS256	PS256	RS256
1	0.10847	0.17793	0.03768	0.05268	0.05235	26	0.09436	0.11637	0.02863	0.05914	0.04622
2	0.1188	0.1903	0.04081	0.05715	0.04954	27	0.09913	0.1153	0.02804	0.05679	0.05407
3	0.11796	0.19853	0.03545	0.06113	0.04472	28	0.11395	0.12467	0.02879	0.04839	0.05011
4	0.12618	0.18734	0.03483	0.05552	0.04788	29	0.11939	0.13305	0.03173	0.05062	0.04176
5	0.10865	0.19698	0.03164	0.04649	0.05002	30	0.11716	0.13554	0.02961	0.05178	0.04413
6	0.10841	0.2313	0.03327	0.05897	0.04359	31	0.12224	0.14309	0.0302	0.0589	0.04768
7	0.10852	0.19308	0.03017	0.0546	0.05278	32	0.1102	0.14546	0.03168	0.05634	0.06928
8	0.11293	0.16764	0.02905	0.04879	0.04877	33	0.11917	0.16892	0.02911	0.05032	0.08817
9	0.1023	0.15777	0.02921	0.05937	0.05019	34	0.12382	0.13952	0.03004	0.05269	0.07161
10	0.10593	0.15435	0.02858	0.04828	0.05023	35	0.11371	0.13907	0.02819	0.04881	0.08834
11	0.12004	0.1629	0.03137	0.05893	0.0483	36	0.10752	0.18628	0.02942	0.06188	0.15708
12	0.10694	0.15978	0.02864	0.04954	0.05056	37	0.114	0.13898	0.03	0.05924	0.07771
13	0.09994	0.15037	0.02834	0.05413	0.04284	38	0.09955	0.1868	0.02807	0.05694	0.07574
14	0.10299	0.14928	0.02801	0.06174	0.04349	39	0.12272	0.1605	0.02902	0.06158	0.04898
15	0.09837	0.14728	0.0272	0.05039	0.04787	40	0.10792	0.12931	0.03073	0.0575	0.05024
16	0.10485	0.14272	0.03609	0.05148	0.05356	41	0.10708	0.14285	0.02956	0.05536	0.04906
17	0.10244	0.13722	0.02996	0.05308	0.05358	42	0.10977	0.14437	0.02589	0.05264	0.07708
18	0.11681	0.13845	0.03054	0.05031	0.04664	43	0.10989	0.11676	0.02573	0.0505	0.04673
19	0.11351	0.13164	0.02927	0.0478	0.04712	44	0.11175	0.12372	0.02771	0.06244	0.04261
20	0.11724	0.13482	0.02642	0.04791	0.0784	45	0.12686	0.14038	0.02726	0.06816	0.04598
21	0.10805	0.13583	0.02681	0.0718	0.04977	46	0.1212	0.13772	0.02802	0.05135	0.04544
22	0.10008	0.13784	0.02795	0.05143	0.04784	47	0.13334	0.15176	0.02662	0.05175	0.04883
23	0.11338	0.1358	0.02858	0.05717	0.04774	48	0.11722	0.12917	0.0276	0.05038	0.0456
24	0.09395	0.13997	0.02976	0.05361	0.05187	49	0.11303	0.13333	0.02714	0.04981	0.04439
25	0.10011	0.1371	0.02835	0.05886	0.0615	50	0.11681	0.12731	0.02751	0.05177	0.04385

The data show that HS256 consistently has the shortest verification time: the average is approximately 0.030 ms, with variation roughly within 0.026–0.041 ms.

RSA-based algorithms (PS256 and RS256) form an intermediate group, with an average verification time of about 0.055 ms, i.e., roughly 1.8–2× slower than HS256. The largest verification costs are observed for ES256 and EdDSA, with average values of approximately 0.11 ms and 0.15 ms, respectively, corresponding to a 3.5–5× slowdown compared to HS256. Thus, HS256 is the most efficient for verification, RSA-based schemes occupy the middle position, and ES256/EdDSA are the slowest modes in verify operations.

The graph (Fig. 3.6) in the form of a bar chart shows the average values of the JWT signature verification time, calculated based on the results of 50 runs for each algorithm. The X-axis shows the signature algorithms, and the Y-axis shows the average verification time in milliseconds.

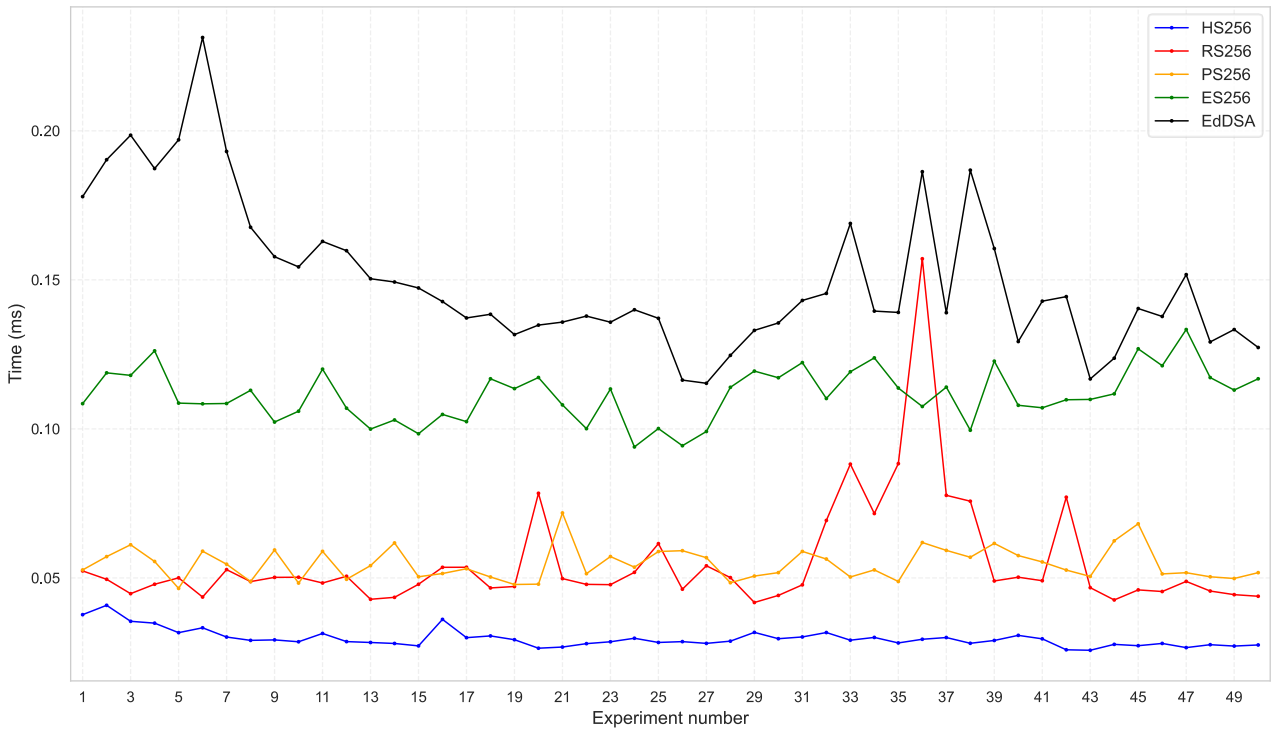


Figure 19. Comparison of average JWT signature verification time for different algorithms (HS256, ES256, EdDSA, PS256, RS256)

Source of the figure: original

A visual analysis of the data reveals that the HS256 column exhibits minimal delays in verify operations, suggesting that it is the lowest among the columns under consideration. The PS256 and RS256 columns are taller, but remain significantly lower than ES256 and especially EdDSA, which form the "slowest" group with the maximum signature verification time. The graph provides clear evidence that HS256 is the optimal choice in terms of signature verification efficiency. RSA algorithms offer an acceptable compromise, while ES256 and EdDSA are suitable when the priority is not speed but other requirements, such as cryptographic strength or standardization.

The table displays the throughput values during JWT signature verification (verify) for five cryptographic algorithms: ES256, EdDSA (Ed25519), HS256, PS256, and RS256. Each row in the table corresponds to a distinct experiment, and the numerical values represent the number of verification operations that can be performed per second (ops/s).

Table 17

16) Results of measuring JWT signature verification throughput for algorithms HS256, ES256, EdDSA, PS256, and RS256 (ops/s)

	ES256	EdDSA	HS256	PS256	RS256		ES256	EdDSA	HS256	PS256	RS256
1	2	3	4	5	6	7	8	9	10	11	12
1	9219.22	5620.3	26540.4	18981.34	19103.38	26	10597.24	8593.64	34929.71	16909	21636.67
2	8417.74	5254.75	24505.91	17498.91	20184.91	27	10087.4	8672.87	35661.29	17607.39	18493.35
3	8477.59	5037.01	28208.43	16358.73	22358.92	28	8775.73	8021.07	34738.94	20663.79	19954.85
4	7925.47	5337.77	28706.97	18012.78	20886.01	29	8376.22	7515.93	31519.26	19754.25	23946.53
5	9203.74	5076.57	31610.29	21511.18	19992.63	30	8535.06	7377.87	33774.21	19312.68	22660.2
6	9224.08	4323.31	30060.76	16958.41	22939.44	31	8180.57	6988.48	33110.04	16979.06	20973.43
7	9215.3	5179.19	33150.1	18315.9	18946.81	32	9074.35	6874.87	31570.08	17747.81	14434.13
8	8855.14	5965.33	34428.6	20495.08	20504.35	33	8391.55	5919.82	34346.9	19872.66	11342.31
9	9775.31	6338.51	34238.04	16843.51	19924.94	34	8076.2	7167.23	33292.54	18978.47	13965.51
10	9440.43	6478.58	34993.1	20712.84	19907.75	35	8794.47	7190.5	35471.95	20487.97	11320.07
11	8330.35	6138.77	31878.98	16968.58	20704.59	36	9300.3	5368.24	33990.1	16159.28	6366.34
12	9350.81	6258.59	34911.3	20187.51	19779.76	37	8772.01	7195.32	33329.36	16880.48	12868.06
13	10005.65	6650.41	35284.96	18474.37	23342.76	38	10044.87	5353.34	35622.52	17563.88	13202.79

Continuation of table 17

14	9710.08	6698.76	35700.08	16197.86	22994.01	39	8148.63	6230.43	34464.79	16239.86	20414.45
15	10166.14	6789.89	36766.19	19845.19	20890.27	40	9265.87	7733.38	32545.47	17390.96	19904.37
16	9537.87	7006.84	27709.64	19423.65	18671.51	41	9338.74	7000.33	33834.52	18062.89	20384.35
17	9761.68	7287.76	33376.23	18838.19	18663.54	42	9109.59	6926.59	38621.01	18995.7	12974.25
18	8560.66	7222.69	32743.18	19876.12	21441.63	43	9099.94	8564.68	38865.15	19802.69	21400.58
19	8809.73	7596.49	34165.35	20920.43	21220.54	44	8948.6	8082.69	36089.93	16016.47	23470.38
20	8529.56	7417.17	37853.95	20871.14	12754.59	45	7882.43	7123.55	36688.27	14671.79	21748.22
21	9254.64	7362.12	37301.02	13927.54	20092.89	46	8250.72	7261.22	35690.04	19472.53	22009.43
22	9992.09	7254.67	35783.27	19443.84	20902.18	47	7499.45	6589.21	37568.17	19324.12	20477.44
23	8819.87	7363.59	34983.76	17491.43	20948.74	48	8530.61	7741.79	36226.9	19848.83	21932.05
24	10643.79	7144.42	33605.67	18653.28	19280.64	49	8847.26	7499.95	36849.42	20075.96	22526.28
25	9988.66	7293.72	35272.02	16989.97	16260.37	50	8560.83	7855.1	36345.67	19315.02	22802.9

Across 50 runs under identical conditions, the highest verification throughput is consistently provided by HS256, with an average of approximately 35,000 ops/s. RS256 and PS256 form an intermediate group at approximately 18,000 – 20,000 ops/s, while RS256 shows higher variability (some runs yield lower values around 6,000 – 11,000 ops/s). The lowest verification throughput is observed for ES256 and EdDSA: ES256 typically achieves about 9,000 – 10,000 ops/s, whereas EdDSA achieves about 7,000 – 8,000 ops/s. Therefore, a clear hierarchy is observed: HS256 is the most efficient by verifications per second, RSA algorithms are intermediate, and ES256/EdDSA are the slowest in terms of verify throughput.

The highest graph (blue) describes HS256, reflecting its undisputed leadership in performance. RS256 (red) and PS256 (orange) demonstrate that RSA-based algorithms provide approximately half the throughput of HS256, yet still significantly outperform elliptic schemes. ES256 (green) exhibits a conspicuously lower bar height, and the EdDSA bar (black) demonstrates the lowest number of signature checks per second for this mode, thereby confirming the observations. The visual comparison offers a clear illustration of the trade-off between security and performance. In scenarios with high service loads, symmetric HS256 is optimal. RSA algorithms are

suitable as a balance between performance and compatibility requirements. ES256 and EdDSA are best used in scenarios where cryptographic strength and compliance with modern standards are prioritized over maximum throughput.

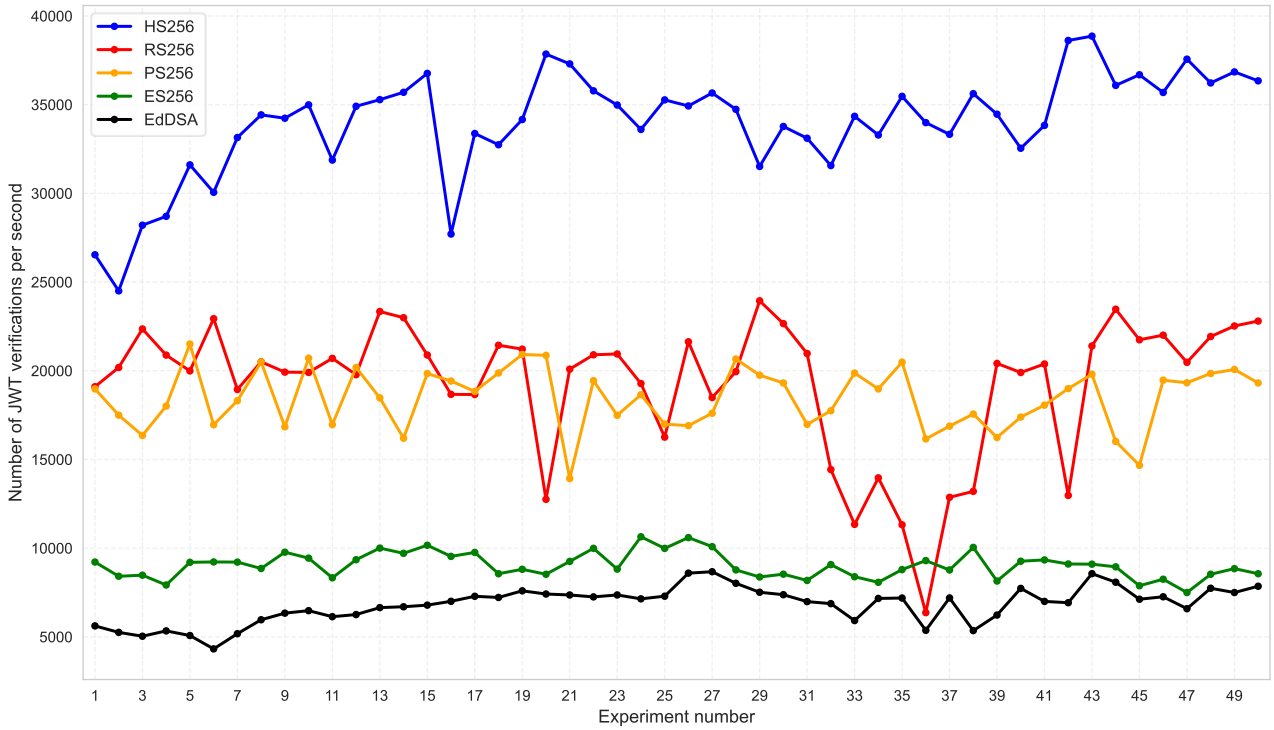


Figure 20. Comparison of JWT verification throughput by the number of verify operations per second

Source of the figure: original

Finally, Table 18 summarizes the averaged encode/verify results for HS256, ES256, EdDSA, PS256, and RS256 (averaged over 50 runs).

Table 18

17) Performance comparison of JWT encode/verify operations for different signing algorithms

Algorithm	Encode: average time, ms	Encode: throughput, thousand ops/s	Verify: average time, ms	Verify: throughput capacity, thousand ops/s
HS256	0,033	30,24	0,030	33,98
ES256	0,065	15,27	0,111	9,03
EdDSA	0,074	13,57	0,150	6,82
PS256	0,618	1,62	0,055	18,44
RS256	0,659	1,52	0,055	19,28

4.6. Assessment of the adequacy of experimental research into the method, model, algorithm.

The study examined a single security feature of web-oriented applications, namely, JWT tokens. However, it is noteworthy that a considerable number of critical mechanisms were not included in the scope of the experiment. The test client can be extended to evaluate multiple security features concurrently; however, the framework employed is intentionally simplified and straightforward to implement. The implemented model is deficient in key rotation and revocation mechanisms, which curtails its applicability in industrial scenarios. Furthermore, the study's emphasis on authentication overshadows the unaddressed matter of authorization. In the context of token-based authentication, users may be granted disparate access rights to a curated subset of resources or discrete microservices, a scenario that necessitates supplementary modeling and analysis.

The experiments were conducted using the Python programming language; however, the same approach to load testing can be implemented for other languages and technology stacks. This would allow for a comparison of the impact of platform features on performance. A notable constraint pertains to the absence of database replication, which entails the storage of data in a single copy. This configuration results in the establishment of a "single point of failure," thereby augmenting the system's

susceptibility to failures and security breaches. In order to enhance reliability and fault tolerance, it is imperative that such infrastructure be deployed in a cloud environment, employing replication, load balancing, and geographic resource distribution mechanisms.

Conclusion

During the experiments, it was observed that as the number of requests increases, network interactions between microservices introduce millisecond-scale delays; consequently, the contribution of the core security functions to the overall request processing time becomes relatively insignificant. To obtain a more realistic picture, it is advisable to perform extended load testing in which the CPU operates under sustained high utilization and system behavior is analyzed under computationally intensive workloads. To reduce response time, caching mechanisms can be implemented at the API Gateway level or within the User Authentication Service, while ensuring that expired tokens are removed via an appropriate cache eviction policy once their validity period has elapsed.

It is possible to replace JWT validation in every microservice with validation performed only in a limited set of specialized services, thereby reducing overall load. If the request rate increases substantially, deploying multiple internal API gateways should be considered. Adding a unique request identifier to the JWT would enable finer-grained correlation of user activity logs and facilitate analysis of which specific functions were executed. Another optimization direction is selecting JWT signature algorithms with better performance characteristics; in particular, Edwards-curve schemes such as Ed25519 and Ed448 demonstrated high performance in the conducted measurements. To limit request volume and mitigate DoS attacks originating from a single user, device, IP address, or set of credentials, rate limiting and load balancing mechanisms can be applied at the API Gateway level.

Summary Table 19 provides recommendations for selecting JWT signature algorithms based on the experimental analysis of encode and verify modes and typical requirements of information systems. For each algorithm (HS256, ES256, EdDSA,

PS256, RS256), the table summarizes key performance and security characteristics, common usage scenarios, and a generalized recommendation.

Table 19

18) Recommendations for Using Algorithms in JWT

Algorithm	Key Features (Performance/Security)	Recommended Use Cases	General Recommendation
HS256	Fastest in both encode and verify; very high throughput; requires secure storage of a shared secret key across all services.	High-load API gateways, microservices with a large volume of authentication requests, internal systems where a shared-secret model is acceptable.	Use as the primary option when the threat model allows symmetric keys and maximum performance is required.
ES256	Moderate performance (slower than HS256, but faster than EdDSA in verify); asymmetric keys; compact EC key material.	External services, B2B integrations where an asymmetric signature is required; modern mobile and web applications emphasizing standardized EC algorithms.	Recommended as a balance between performance, modern standards, and asymmetric cryptography.
EdDSA (Ed25519)	High cryptographic strength and modern design; encode/verify slower than HS256 and ES256; lowest throughput among the evaluated verify modes.	Systems with increased security requirements, long-lived tokens, services where modern cryptographic primitives (Ed25519) are prioritized and load is moderate.	Use where security and modern primitives are more important than maximum performance.
PS256	Very slow encode (tens of times slower than HS256), but verify is noticeably faster and comparable to RS256; asymmetric RSA signature (RSASSA-PSS).	Systems where signing is infrequent (e.g., performed by a dedicated service) but verification is frequent; integrations that require RSASSA-PSS due to security policies.	Apply when policies/standards mandate PSS; not recommended for mass token signing under high load.
RS256	Very slow encode; fast verify (second most efficient after HS256); widely supported across libraries and infrastructure.	Legacy systems, enterprise platforms, and identity federation (OIDC, SSO) where RSA is historically used and maximum compatibility is required.	Recommended for compatibility with existing infrastructure; for new high-load systems, consider HS256 or ES256 instead.

The table indicates that HS256 should be treated as the baseline choice for high-load services due to minimal encoding and verification time and maximum throughput, provided that a symmetric key model is acceptable. ES256 is positioned as a compromise between performance and modern asymmetric-cryptography requirements, making it suitable for external integrations and modern web and mobile applications. EdDSA (Ed25519) is recommended for systems with higher cryptographic-strength requirements where reduced throughput is acceptable in exchange for the use of modern primitives.

RSA-based algorithms are considered separately. PS256 and RS256 have significantly slower encode performance, but provide acceptable verification speed and remain relevant due to widespread deployment and security-policy requirements in existing infrastructure. For these algorithms, the table recommends use primarily in scenarios where compatibility with established solutions (SSO, OIDC, enterprise platforms) is more critical than maximum throughput. Overall, the table serves as a practical checklist for engineers and architects, enabling a rapid mapping of algorithm properties to project requirements and supporting an informed choice of JWT signing mode.

DOI: 10.46299/ISG.2026.MONO.TECH.2.3.2

3.2 Мультимодальні біометричні системи

3.2.1 ВСТУП

Біометричні системи застосовують для автоматизованого підтвердження або встановлення особи за ознаками, які пов'язані з тілом, поведінкою або фізіологічними сигналами людини. Упродовж тривалого часу практичні рішення здебільшого будувалися навколо однієї модальності: відбитка пальця, обличчя, райдужної оболонки, голосу чи підпису. Такий підхід зручний, але він виявився недостатнім у середовищах, де якість даних змінюється, користувачі мають різні фізичні можливості, а зловмисник може імітувати або перехоплювати окрему ознаку.

Мультимодальна біометрія виникла як відповідь на цю обмеженість. Її сутність полягає у поєднанні кількох біометричних джерел, які разом формують надійніше рішення, ніж кожне джерело окремо. При цьому мультимодальність не зводиться до простого додавання ще одного датчика. Вона вимагає узгодження процедур реєстрації, попереднього оброблення, виділення ознак, ф'южну, встановлення порогів, захисту еталонів і перевірки стійкості до атак. Саме тому сучасні огляди розглядають такі системи як комплексні інформаційні рішення, а не як сукупність незалежних класифікаторів [40, с. 2].

Актуальність теми посилилася через поширення мобільного банкінгу, дистанційної ідентифікації, розумних будинків, пристроїв інтернету речей, електронних адміністративних послуг і систем неперервної автентифікації. У цих сценаріях користувач часто взаємодіє з системою поза контрольованим приміщенням, а якість зразків залежить від освітлення, камери, шуму, положення тіла, швидкості руху або стану датчика. Одномодальна система може в таких умовах або помилково відмовляти законному користувачу, або, навпаки, приймати підроблений зразок.

Водночас мультимодальність не є самодостатньою гарантією безпеки. Якщо система збирає більше біометричних даних, то зростає й відповідальність за їх оброблення. Витік пароля можна компенсувати його заміною, тоді як біометрична ознака є пов'язаною з людиною протягом значної частини життя. Тому сучасні дослідження дедалі частіше поєднують питання точності з питаннями приватності, пояснюваності, відкличності еталонів, виявлення атак пред'явлення та справедливості щодо різних груп користувачів [45-48].

Метою цього розділу є систематизація теоретичних засад і практичних підходів до побудови мультимодальних біометричних систем. Для цього розглянуто основні поняття, причини переходу до мультимодальності, загальну архітектуру, рівні та методи ф'южну, найпоширеніші поєднання ознак, способи оцінювання, типові вразливості й механізми захисту. Окрему увагу приділено тому, що висока лабораторна точність має значення лише тоді, коли вона підтверджена коректним протоколом тестування та не досягається ціною надмірного збирання чутливих даних.

3.2.2 ТЕОРЕТИЧНІ ЗАСАДИ МУЛЬТИМОДАЛЬНОЇ БІОМЕТРІЇ

3.2.2.1 Основні поняття та визначення

Біометричною модальністю називають тип даних, за допомогою яких система описує людину для подальшого порівняння. До фізіологічних модальностей належать обличчя, райдужна оболонка, відбиток пальця, венозний рисунок, форма долоні, сітківка й електрокардіографічні або фотоплетизмографічні сигнали. До поведінкових модальностей належать голос, динамічний підпис, хода, клавіатурний ритм, рухи курсора та спосіб взаємодії з сенсорним екраном. Межа між цими групами не завжди є жорсткою: голос має анатомічну основу, але істотно залежить від поведінки, емоційного стану й умов запису.

Мультимодальною біометричною системою доцільно вважати систему, що інтегрує щонайменше дві різні біометричні модальності для формування

спільного рішення про особу. Поруч із цим у літературі використовують ширше поняття багатобіометричної системи. Воно може охоплювати кілька датчиків для однієї ознаки, кілька алгоритмів аналізу одного зразка, кілька екземплярів однієї ознаки або кілька різних модальностей. У цьому розділі основна увага зосереджена саме на мультимодальному випадку, де система працює з різними джерелами біометричної інформації [40].

Робота біометричної системи зазвичай складається з двох режимів: реєстрації та перевірки. Під час реєстрації система отримує один або кілька зразків, перевіряє їх якість, виконує попереднє оброблення, виділяє ознаки й формує еталон. Під час перевірки новий зразок перетворюється на ембеддінг або інше числове представлення, після чого порівнюється з еталоном. У мультимодальній системі цей процес повторюється для кожної модальності, а остаточне рішення отримують шляхом ф'южну даних.

Еталон не слід ототожнювати з початковим зображенням чи сирим сигналом. У сучасних системах це здебільшого математичне представлення, яке має бути достатнім для порівняння, але не повинно розкривати зайвої інформації про користувача. Проте навіть перетворений еталон залишається чутливим об'єктом. Він може бути пов'язаний з однією особою в різних сервісах, а його компрометація створює довготривалий ризик. Тому до еталона висувають вимоги незворотності, відкличності, відокремленості між застосуваннями й стабільності після захисного перетворення [47, с. 17].

Важливим є також розмежування ідентифікації та верифікації. Ідентифікація відповідає на питання, кому з множини зареєстрованих осіб відповідає поданий зразок. Верифікація перевіряє заявлену тотожність і порівнює поточні дані з еталоном конкретного користувача. У першому випадку критичними стають масштабованість і правильне ранжування кандидатів, у другому - вибір порога прийняття рішення та баланс між хибним допуском і хибною відмовою.

3.2.2.2 Причини переходу до мультимодальності

Перша причина переходу до мультимодальності пов'язана з якістю даних. Зразки обличчя залежать від освітлення, пози, віку та наявності маски; відбитки пальців погіршуються через вологість, пошкодження шкіри або забруднення сенсора; голос змінюється під впливом шуму, хвороби чи емоційного стану. Якщо система спирається лише на одну ознаку, будь-яке погіршення її якості безпосередньо впливає на рішення. Поєднання кількох ознак дає змогу частково компенсувати слабкість одного каналу інформацією з іншого.

Друга причина полягає у підвищенні стійкості до атак. Для одномодальної системи зловмиснику достатньо зосередитися на одній поверхні атаки: фотографії обличчя, відтворенні голосу, штучному пальці або текстурній лінзі для райдужної оболонки. Мультимодальна система ускладнює атаку, оскільки потребує узгодженої підміни кількох ознак або впливу на модуль ф'южну. Втім, це не скасовує потреби у перевірці живості й виявленні атак пред'явлення, що підкреслюється у сучасних оглядах з біометричної безпеки [45, с. 14].

Третя причина стосується зручності та доступності. Не всі користувачі можуть однаково легко подати певну ознаку. Частина людей має пошкоджені відбитки, порушення мовлення, обмеження рухливості або умови праці, які ускладнюють використання конкретного датчика. Мультимодальна система може запропонувати резервний канал або зважити доступні модальності відповідно до якості поданих зразків. Це особливо важливо для масових державних і фінансових сервісів, де система не повинна відкидати користувача лише через непридатність однієї ознаки.

Четверта причина пов'язана з контекстом використання. Для низькоризикової операції може бути достатньо обличчя або відбитка пальця. Для фінансової операції, доступу до критичної інфраструктури чи віддаленої реєстрації доцільним стає поєднання кількох джерел. Тому сучасні архітектури переходять від фіксованої схеми до ризик-орієнтованого добору модальностей,

коли кількість і вага ознак залежать від типу операції, історії користувача, якості сигналів та виявлених ознак атаки [49, с. 16].

Рисунок 1. Логіка переходу від одномодальної до мультимодальної біометрії

Джерело рисунка: авторська розробка.

Отже, перехід до мультимодальності має не тільки технічну, а й організаційну природу. Він вимагає переосмислення всього життєвого циклу біометричних даних: від отримання згоди користувача до видалення або відкликання еталона. Перевага мультимодальної системи виникає лише тоді, коли додаткові модальності справді зменшують невизначеність рішення, а не збільшують складність без помітної користі.

3.2.3 АРХІТЕКТУРА МУЛЬТИМОДАЛЬНИХ БІОМЕТРИЧНИХ СИСТЕМ

3.2.3.1 Загальна структура системи

Типова мультимодальна біометрична система містить кілька взаємопов'язаних модулів. Перший модуль відповідає за збирання даних: камеру, мікрофон, сканер відбитка, інфрачервоний сенсор, носимий пристрій або програмний датчик поведінки. Другий модуль оцінює якість зразка і відхиляє

явно непридатні дані. Третій виконує попереднє оброблення: нормалізацію освітлення, вирівнювання обличчя, сегментацію райдужної оболонки, фільтрацію шуму або виділення області інтересу.

Після попереднього оброблення система формує ознаки. У класичних рішеннях використовувалися локальні дескриптори, текстурні ознаки, мінуції відбитка, спектральні характеристики голосу або геометричні параметри підпису. У сучасних працях переважають глибинні моделі, які будують ембеддінги зображень, сигналів і часових послідовностей. Для зображень поширеними є згорткові нейронні мережі та трансформерні архітектури, а для голосу, ходи й клавіатурного ритму - рекурентні, темпоральні або послідовні моделі [42-44; 50].

Наступним етапом є ф'южн. Він може виконуватися до класифікації, після отримання оцінок подібності або вже на рівні рішень. У якісно спроектованій системі модуль ф'южну не просто додає результати окремих каналів, а враховує їхню якість, надійність, кореляцію та можливу наявність атаки. Наприклад, низька якість зображення обличчя не повинна автоматично призводити до відмови, якщо інші модальності підтверджують особу, але вона має зменшувати вагу відповідного каналу.

Останніми елементами є модуль прийняття рішення, база еталонів, журналювання подій і засоби захисту. База еталонів повинна зберігати не сирі зразки, а захищені представлення. Журналювання має фіксувати критичні події без надмірного накопичення чутливих даних. Для високоризикових сценаріїв важливими є локальне зіставлення на пристрої, криптографічний захист, контроль доступу до еталонів і окремий модуль виявлення атак пред'явлення [47, с. 9, с. 17].

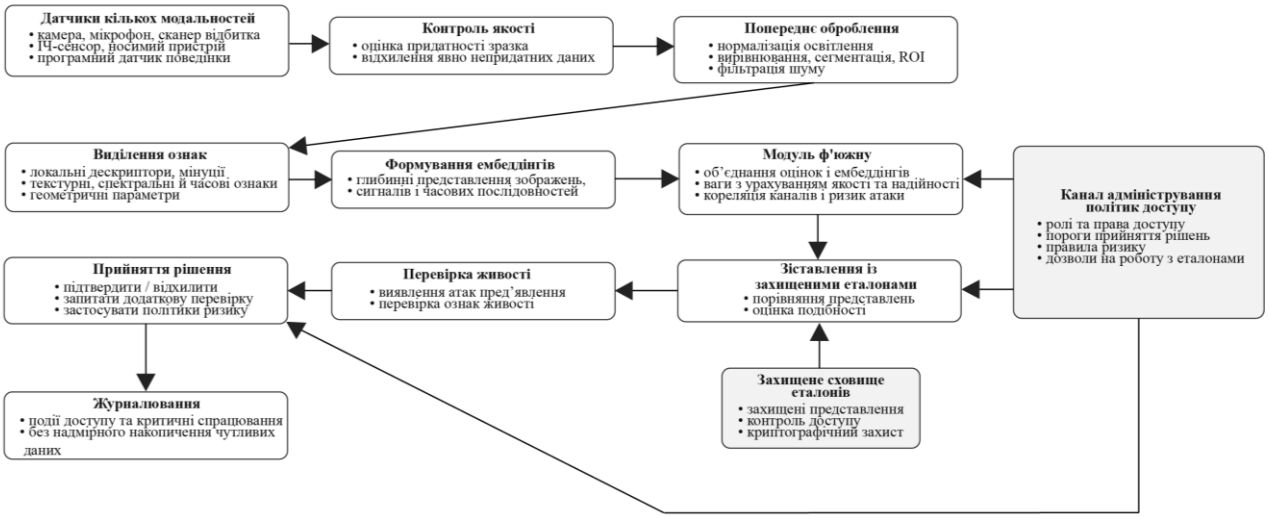


Рисунок 2. Узагальнена архітектура мультимодальної біометричної системи

Джерело рисунка: авторська розробка.

3.2.3.2 Рівні злиття даних

Ф'южн на рівні первинних даних передбачає об'єднання необроблених або мінімально оброблених зразків. Він може бути корисним тоді, коли кілька датчиків спостерігають один і той самий об'єкт у сумісних умовах, наприклад у поєднанні різних зображень руки. Перевагою є збереження максимальної кількості інформації, а недоліком - потреба в синхронізації, однаковій якості вимірювань і високих обчислювальних ресурсах.

Ф'южн на рівні ознак працює з векторами ознак або ембеддінгами, отриманими для кожної модальності. Він дає змогу сформувати спільний простір, у якому система навчається враховувати взаємодію між джерелами. Саме цей рівень часто застосовують у глибинних моделях, зокрема в архітектурах із механізмами уваги. Недоліком є висока розмірність, ризик перенавчання та потреба у достатньому обсязі справді мультимодальних даних.

Ф'южн на рівні оцінок поєднує числові результати окремих модальностей, наприклад оцінки подібності або ймовірності належності до заявленого користувача. Цей рівень вважають практично зручним, оскільки окремі підсистеми можна розробляти незалежно, а потім об'єднувати їхні оцінки. Він

також полегшує додавання нової модальності без повного перенавчання всієї системи. Багато прикладних робіт з обличчям і голосом або обличчям і підписом використовують саме цей рівень [42, с. 4].

Ф'южн на рівні рішень об'єднує вже готові відповіді окремих каналів: прийняти, відхилити або передати на повторну перевірку. Його перевага полягає у простоті та прозорості, проте частина інформації втрачається, адже система не бачить, наскільки впевненою була кожна підсистема. Тому такий рівень доцільний у простих або нормативно обмежених системах, але менш придатний там, де потрібне тонке налаштування ризику.

Окремо виділяють гібридний ф'южн, у якому поєднуються кілька рівнів. Наприклад, обличчя та райдужну оболонку можна об'єднати на рівні ембеддінгів, а результат поєднати з голосом на рівні оцінок. Гібридні підходи особливо корисні у складних системах, де модальності мають різну природу: зображення, звук, часові сигнали й поведінкові послідовності. Проте їхня складність вимагає ретельного протоколу тестування, інакше приріст точності може бути наслідком перенавчання.

3.2.3.3 Методи злиття

Найпростішими методами є правила суми, добутку, мінімуму, максимуму та зваженого голосування. Вони мають перевагу у прозорості: можна пояснити, чому певна модальність вплинула на рішення більше або менше. Для оцінкового ф'южну ці методи потребують нормалізації, оскільки різні підсистеми можуть видавати оцінки у різних шкалах. Якщо нормалізацію виконано неправильно, сильніша шкала може домінувати над справді інформативнішою модальністю.

Статистичні методи розглядають оцінки окремих модальностей як ознаки для метакласифікатора. У такій ролі можуть виступати логістична регресія, метод опорних векторів, дерева рішень або баєсівські моделі. Їхня перевага полягає у здатності навчати ваги на даних, а не встановлювати їх вручну. Недоліком є залежність від репрезентативності навчального набору: якщо він не

містить складних умов або атак, модель може помилково оцінювати надійність каналів.

Глибинні методи ф'южну формують спільне представлення за допомогою нейронних мереж. Вони можуть об'єднувати ембеддінги, виділяти міжмодальні залежності, застосовувати механізми уваги або адаптивно змінювати ваги залежно від якості зразка. Наприклад, системи з камерною фотоплетизмограмою та відбитком пальця використовують один смартфонний сценарій знімання, а ф'южн допомагає поєднати текстурні й фізіологічні ознаки [44]. Такі підходи зручні, але потребують обережності щодо перенавчання і пояснюваності.

Адаптивні методи враховують контекст. Система може зменшувати вагу голосу в шумному середовищі, підвищувати вимоги до обличчя під час фінансової операції або використовувати поведінкові ознаки лише як додатковий сигнал після первинного входу. У неперервній автентифікації контекстно-керований ф'южн є особливо важливим, оскільки доступні дані змінюються в часі, а рішення має накопичуватися поступово, без нав'язування користувачу постійних перевірок [49].

Метод ф'южну слід обирати не лише за найкращим показником точності, а й за придатністю до впровадження. Для банківської системи важливими є контроль порогів, пояснюваність і можливість аудиту. Для мобільного пристрою - швидкість, енергоспоживання та локальне оброблення. Для державної системи - справедливість щодо різних груп користувачів і мінімізація даних. Тому універсального методу ф'южну не існує; його ефективність визначається конкретним сценарієм.

3.2.4 ОСНОВНІ КОМБІНАЦІЇ БІОМЕТРИЧНИХ МОДАЛЬНОСТЕЙ

3.2.4.1 Комбінації на основі зображень

До найпоширеніших належать комбінації обличчя, райдужної оболонки, відбитка пальця, вен пальця, вен долоні та геометрії руки. Їхня перевага полягає у високій інформативності зображень і наявності розвинених методів

глибинного виділення ознак. Обличчя забезпечує зручність і швидкість, райдужна оболонка - високу індивідуальність, відбиток пальця - компактність і поширеність датчиків, а венозний рисунок - додатковий захист від поверхневих підробок.

Комбінація обличчя і райдужної оболонки часто розглядається для високоточних систем, оскільки ці ознаки мають різні джерела помилок. Обличчя може погіршуватися через освітлення або позу, тоді як райдужна оболонка потребує якісного знімання очей. Якщо обидві модальності доступні, ф'южн здатний зменшити неоднозначність. Водночас така система повинна враховувати зручність користувача, адже примусове наближення до датчика може бути неприйнятним у масових сервісах.

Комбінації відбитка пальця з венозним рисунком або геометрією пальця є прикладом hand-based біометрії. Огляд таких систем показує, що їхня перевага полягає у спільному зніманні кількох ознак руки, однак це потребує спеціалізованого обладнання та контролю якості зображень [41]. Для реального застосування важливо, щоб додатковий канал не лише підвищував лабораторну точність, а й допомагав протидіяти атакам і зменшував кількість відмов за поганого стану шкіри.

Окрему групу становлять тримодальні системи на основі обличчя, райдужної оболонки та вен пальця. Сучасні праці використовують для них попередньо навчені згорткові мережі, трансформери зору та ф'южн на рівні оцінок [51]. Такий підхід демонструє потенціал глибинних моделей, але потребує незалежної перевірки на різних датчиках і сеансах. Без цього висока точність може відображати особливості конкретного набору даних, а не справжню узагальнювальну здатність.

3.2.4.2 Комбінації з поведінковими ознаками

Поведінкові ознаки цінні тим, що можуть збиратися природно під час взаємодії користувача із системою. Голос, підпис, хода, клавіатурний ритм і рухи

сенсорного екрана не завжди мають таку саму стабільність, як райдужна оболонка або відбиток пальця, проте вони корисні як додаткові джерела інформації. Їхня роль особливо помітна у віддалених сервісах і неперервній автентифікації, де система не повинна постійно вимагати активного знімання фізіологічної ознаки.

Поєднання обличчя та голосу є природним для мобільних і веб-сервісів, оскільки камера і мікрофон є доступними на більшості пристроїв. Дослідження із застосуванням FaceNet для обличчя та гаусових сумішей для голосу показує, що ф'южн оцінок може знижувати коефіцієнт рівних помилок порівняно з окремими модальностями [42]. Водночас така система має бути стійкою до фотографій, відеозаписів, синтезованого голосу та повторного відтворення аудіо.

Поєднання обличчя і динамічного підпису доцільне для фінансових і юридично значущих операцій. Обличчя підтверджує присутність заявленої особи, а підпис фіксує поведінкову послідовність: швидкість, тиск, порядок штрихів і часові затримки. Праця 2024 року демонструє, що поєднання статичних і динамічних ознак підпису з даними обличчя може бути корисним для онлайн-безпеки, але потребує достатньої кількості зразків і перевірки на реалістичних віддалених умовах [43].

Клавіатурний ритм і хода придатні для неперервної автентифікації. Після початкового входу система може спостерігати за поведінкою користувача без окремої процедури перевірки. Контекстно-кероване поєднання цих ознак дає змогу враховувати, чи користувач набирає текст, рухається з телефоном або взаємодіє з носимим пристроєм [49]. Проблемою залишається мінливість поведінки через втому, стрес, травму або зміну робочого середовища.

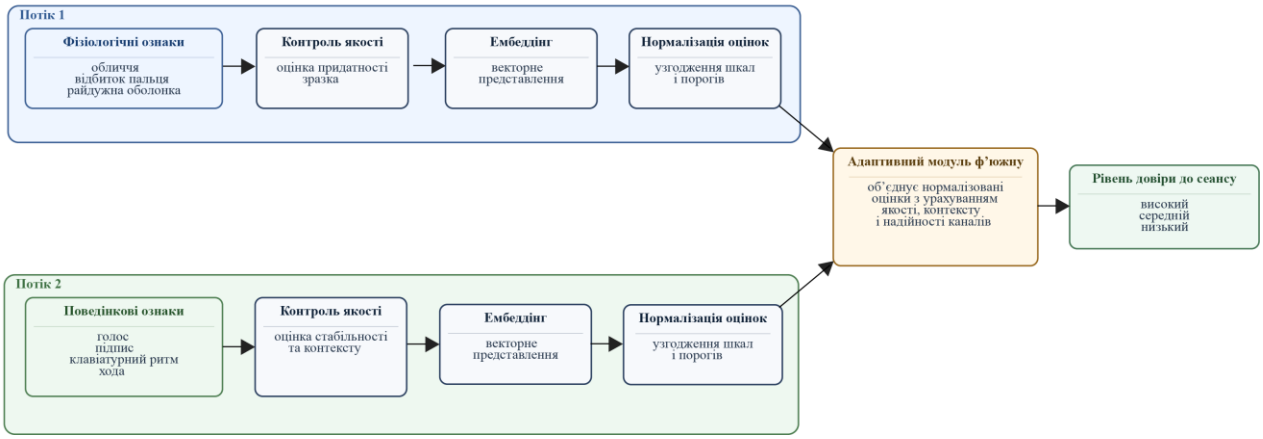


Рисунок 3. Комбінація фізіологічних і поведінкових модальностей

Джерело рисунка: авторська розробка.

3.2.4.3 Системи з трьома і більше модальностями

Системи з трьома і більше модальностями створюють для підвищення стійкості та резервування. Вони можуть поєднувати, наприклад, відбиток пальця, райдужну оболонку й електрокардіограму; обличчя, райдужну оболонку й венозний рисунок; відбиток, геометрію та вени пальця. Такий підхід збільшує інформаційне покриття, але водночас підвищує вартість, тривалість реєстрації, складність ф'южну та ризики для приватності.

Комбінація відбитка пальця, райдужної оболонки та електрокардіографічного сигналу цікава тим, що додає до двох зображувальних ознак біосигнал, пов'язаний із життєдіяльністю. У праці 2025 року для такої системи використано згорткові моделі та Swin Transformer [50]. Важливим є не лише отриманий показник точності, а й питання, чи справді електрокардіограма підвищує захист від підміни та чи не створює вона надмірної незручності для користувача.

Тримодальні системи на основі руки мають іншу логіку: кілька ознак можна отримати під час одного контакту або близьких процедур знімання. Відбиток, геометрія та венозний рисунок пальця можуть взаємно доповнювати одне одного, оскільки відображають різні рівні структури руки. Такі рішення є

перспективними для контролю доступу й промислових середовищ, але вони потребують спеціалізованих датчиків і стійких алгоритмів сегментації.

Головним принципом для багатомодальних систем має бути мінімальна достатність. Якщо дві модальності забезпечують прийнятний рівень безпеки для певного сценарію, додавання третьої потрібно обґрунтувати: підвищенням стійкості до атаки, резервуванням, доступністю для користувачів або вимогами ризик-орієнтованої політики. Інакше система лише збирає більше чутливих даних без пропорційної користі.

3.2.5 МЕТОДИ ОЦІНЮВАННЯ ЯКОСТІ ТА ПРОДУКТИВНОСТІ

3.2.5.1 Метрики оцінювання

Оцінювання мультимодальної біометричної системи не можна зводити до загальної точності. Для верифікації ключовими є частота хибного допуску (FAR), частота хибної відмови (FRR), коефіцієнт рівних помилок (EER), істинне прийняття за заданого рівня хибного допуску та площа під кривою робочих характеристик. Для ідентифікації важливими є точність першого рангу, кумулятивна характеристика збігу та середній ранг правильної особи.

FAR показує, як часто система помилково приймає сторонню особу за заявленого користувача. FRR показує, як часто вона відхиляє справжнього користувача. Ці показники залежать від порога рішення: жорсткіший поріг зменшує хибні допуски, але може збільшити хибні відмови. Тому одна й та сама система може бути придатною для низькоризикового входу в застосунок і непридатною для високоризикової операції, якщо не змінити пороги.

EER є зручним узагальненим показником, але його не слід абсолютизувати. У точці рівних помилок система має однакові FAR і FRR, проте реальні сервіси рідко працюють саме в цій точці. У банківській або прикордонній системі може бути важливішим низький FAR, навіть якщо це збільшує кількість повторних перевірок. У соціальних або медичних сервісах надто високий FRR може створювати неприйнятні бар'єри для законних користувачів.

Для виявлення атак пред'явлення застосовують окремі показники, зокрема APCER, BPCER і ACER. Вони показують, як часто система пропускає атаку, як часто помилково відхиляє справжнього користувача і якою є середня помилка. Це важливо, тому що система може добре розпізнавати зареєстрованих осіб, але бути слабкою проти фотографій, масок, штучних пальців або синтезованого голосу. Огляди 2024-2026 років підкреслюють, що найбільшою проблемою залишається узагальнення на невідомі типи підробок і нові датчики [45, с. 14].

Коректне оцінювання має показувати не лише середнє значення метрики, а й поведінку системи у підгрупах, за різних датчиків, сеансів і умов середовища. Для мультимодальних систем важливо виконувати абляційний аналіз: оцінювати кожну модальність окремо, всі модальності разом і варіанти з відсутніми або погіршеними каналами. Лише так можна зрозуміти, чи ф'южн справді додає стійкості, а не приховує слабкість окремої моделі.

3.2.5.2 Бази даних і бенчмарки

Бази даних визначають межі доказовості експерименту. У біометрії широко використовують набори для окремих модальностей: CASIA та IITD для райдужної оболонки, FVC і SOCOFing для відбитків пальців, CASIA-B для ходи, а також численні набори для обличчя, голосу, підпису, вен пальця й електрокардіографії. Мультимодальні дослідження часто формують спільні набори шляхом поєднання кількох одномодальних баз, але такий підхід має методологічні обмеження.

Головне обмеження штучного об'єднання полягає у відсутності реального зв'язку між модальностями однієї особи. Якщо відбиток пальця, райдужну оболонку й електрокардіограму взято з різних наборів, система фактично не бачить справжнього мультимодального профілю користувача. У такому разі результати можуть переоцінювати стійкість ф'южну. Найбільш цінними є набори, у яких кілька ознак знято в одних і тих самих осіб, у різні сеанси та на кількох пристроях.

Друге обмеження пов'язане з розміром вибірки. Глибинні моделі мають багато параметрів, а мультимодальний ф'южн додає ще один рівень складності. Якщо кількість користувачів і сеансів невелика, висока точність може бути наслідком перенавчання. Тому важливими є суворе розділення користувачів між навчальною і тестовою частинами, міжсесійна перевірка, міжбазове тестування та опис усіх процедур попереднього оброблення.

Бенчмарки мають також враховувати справедливість. Усереднена точність може приховувати різні показники FAR і FRR для вікових, гендерних або інших груп. У мультимодальній системі одна модальність може компенсувати зміщення іншої, але може й посилити його, якщо модуль ф'южну неправильно зважує джерела. Тому сучасна оцінка повинна містити стратифікований аналіз, а не лише одну підсумкову таблицю результатів.

3.2.6 ВРАЗЛИВОСТІ ТА МЕТОДИ ЗАХИСТУ

3.2.6.1 Типи атак

Найвідомішими є атаки пред'явлення, коли зловмисник подає датчику підроблену або відтворену ознаку. Для обличчя це можуть бути фотографії, відео, маски або згенеровані зображення. Для відбитків пальців - штучні пальці з силікону, желатину чи інших матеріалів. Для голосу - запис або синтезована мова. Для райдужної оболонки - друковані зображення, контактні лінзи з текстурою або екранне відтворення. У мультимодальній системі можлива як атака на один канал, так і комбінована атака на кілька каналів.

Другий тип становлять атаки на канал передавання і програмну інфраструктуру. Зловмисник може перехопити зразок між датчиком і модулем оброблення, підмінити ембеддінг, повторно подати раніше записані дані або вплинути на нормалізацію оцінок. Якщо система вважає, що дані вже перевірені на справжність, така підміна може бути небезпечнішою за фізичну атаку на датчик.

Третій тип пов'язаний із базою еталонів. Викрадення еталонів створює ризик повторного використання в іншій системі, реконструкції частини біометричної інформації або побудови атак на конкретного користувача. Мультимодальна база є ще ціннішою для зловмисника, оскільки містить кілька пов'язаних ознак однієї особи. Саме тому захист еталонів є центральним елементом безпеки, а не додатковою функцією після розпізнавання [47, с. 9].

Четвертий тип становлять атаки на модель. До них належать перенавчання на отруєних даних, зворотне відновлення ознак, підбір зразків для обходу порога, атаки на пояснюваність і маніпулювання модулем ф'южну. У системах із глибинними моделями окремим ризиком є слабка переносимість детекторів атак на нові матеріали підробок або інші датчики. Тому тестування має включати не лише відомі атаки, а й умови, які відрізняються від навчальних [45, с. 14].

3.2.6.2 Механізми захисту

Перший рівень захисту - контроль якості та перевірка живості. Система повинна визначати, чи зразок є придатним для розпізнавання і чи має ознаки живого джерела. Для обличчя використовують аналіз рухів, глибини, текстури, відблисків і мікродинаміки. Для відбитків пальців - властивості шкіри, потових пор, кровонаповнення або оптичні ознаки. Для голосу - виявлення повторного відтворення і синтезованої мови. Живість не повинна бути зовнішнім фільтром; він має впливати на саме рішення ф'южну.

Другий рівень - захист біометричного еталона. До основних підходів належать біометрія, що скасовується, біометричні криптосистеми, незворотні перетворення, фільтри Блума, випадкові проєкції та шаблони, які можна відкликати. Сутність біометрії, що скасовується, полягає в тому, що еталон формується через контрольоване перетворення. Якщо він скомпрометований, система змінює параметри перетворення і створює новий еталон із тих самих біометричних даних [47, с. 17].

Третій рівень - криптографічне зіставлення. Гомоморфне шифрування дає змогу виконувати окремі операції над зашифрованими еталонами без їх розкриття. Дослідження щодо мультимодального розпізнавання райдужної оболонки та обличчя показує, що зіставлення в зашифрованому домені може поєднувати точність і захист приватності [48]. Водночас такі методи потребують значних обчислювальних ресурсів і ретельного налаштування, щоб не погіршити зручність користувача.

Четвертий рівень - навчання із збереженням приватності. Федеративне навчання дає змогу тренувати моделі без централізованого збирання сирих біометричних даних. У біометрії воно є перспективним, оскільки дані часто належать різним організаціям або зберігаються на пристроях користувачів [46]. Для мультимодальних систем проблема складніша: одні клієнти можуть мати обличчя і голос, інші - відбиток і поведінкові дані, а якість датчиків може відрізнятися.

Рисунок 4. Багаторівневий захист мультимодальної біометричної системи

Механізми захисту потрібно оцінювати разом із продуктивністю. Надто сильне перетворення еталона може погіршити точність, а складне шифрування може збільшити затримку до неприйнятного рівня. З іншого боку, висока

точність без захисту еталонів створює довготривалий ризик для користувача. Тому проєктування мультимодальної системи є пошуком балансу між точністю, безпекою, приватністю, швидкодією й пояснюваністю.

3.2.7 ПРАКТИЧНІ ЗАСТОСУВАННЯ

У мобільних і банківських сервісах мультимодальна біометрія використовується для зменшення ризику шахрайства та підвищення зручності користувача. Типовими є поєднання обличчя, голосу, відбитка пальця, динамічного підпису або поведінкових ознак під час взаємодії з пристроєм. Такі системи мають працювати з різними телефонами, камерами, шумовими умовами й мережевими затримками. Тому важливими є локальне оброблення, контроль якості зразків і чітке розділення операцій із різним рівнем ризику.

У системах контролю фізичного доступу частіше застосовують обличчя, райдужну оболонку, відбиток пальця, вени пальця або долоні. Мультимодальність тут корисна тим, що дозволяє зменшити хибні допуски та забезпечити резервування, коли один датчик тимчасово недоступний. Для промислових і медичних середовищ важливо враховувати рукавички, забруднення рук, маски, змінне освітлення та потребу у швидкому проходженні великої кількості людей.

У розумних будинках і системах інтернету речей біометрія поєднується з поведінковими шаблонами. Наприклад, обличчя може використовуватися для первинного входу, а хода або поведінка у приміщенні - для подальшого підтвердження присутності користувача. Праця 2026 року щодо розумного будинку розглядає поєднання глибинного розпізнавання обличчя з аналізом ходи, що відображає загальну тенденцію до контекстної автентифікації [52].

У медичних і соціальних сервісах мультимодальність може підвищити доступність, але потребує особливої обережності. Поєднання обличчя, голосу, підпису або біосигналів має враховувати вік, стан здоров'я, тимчасові порушення мовлення чи руху. Для таких сценаріїв важлива не лише низька помилка, а й

можливість альтернативної перевірки без дискримінації користувачів, для яких певна ознака є нестабільною.

У корпоративних системах дедалі більшого значення набуває неперервна автентифікація. Вона не обмежується одноразовим входом, а періодично оцінює, чи з пристроєм працює той самий користувач. Для цього можуть використовуватися клавіатурний ритм, рухи миші, хода, положення пристрою або інші поведінкові дані. Такий підхід зменшує ризик захоплення сеансу, але потребує прозорої політики приватності та мінімізації зібраної інформації.

3.2.8 ТЕНДЕНЦІЇ ТА ВІДКРИТІ ПРОБЛЕМИ

Першою помітною тенденцією є перехід від фіксованого ф'южну до адаптивного. Система майбутнього повинна враховувати якість кожної модальності, ознаки атаки, контекст операції, доступні датчики й попередню історію взаємодії. Це означає, що ваги модальностей мають бути не сталими, а залежними від ситуації. Водночас адаптивність ускладнює пояснення рішення, тому вона має супроводжуватися журналюванням і прозорими правилами аудиту.

Другою тенденцією є розвиток моделей з увагою та трансформерних архітектур. Вони дають змогу виявляти взаємозв'язки між модальностями й автоматично виділяти найбільш інформативні частини даних. Для мультимодальної біометрії це корисно, адже якість зразків змінюється від сеансу до сеансу. Однак такі моделі потребують великих і різноманітних наборів даних, а також перевірки на стійкість до перенавчання й демографічних зміщень [54].

Третьою тенденцією є захист приватності як частина архітектури, а не зовнішній додаток. Біометрія, що скасовується, гомоморфне шифрування, локальне зіставлення, федеративне навчання та диференційна приватність формують новий підхід, у якому точність не розглядається окремо від ризику витоку. Сучасні праці з гомоморфного зіставлення та захищених еталонів

показують, що цей напрям швидко розвивається, але ще потребує зниження затримок і спрощення впровадження [48, с. 18].

Четвертою проблемою є неповні дані. У реальній системі не всі модальності доступні завжди: користувач може бути в темному приміщенні, мікрофон може працювати з шумом, датчик відбитка може бути забруднений, а носимий пристрій - розряджений. Тому мультимодальна система повинна мати політику часткового рішення. Вона має визначати, коли достатньо підмножини ознак, коли потрібно повторити знімання, а коли слід перейти до альтернативної процедури.

П'ятою проблемою є справедливість і демографічні зміщення. Якщо навчальні дані нерівномірно представляють різні групи користувачів, система може демонструвати різні показники помилок для цих груп. Мультимодальність може зменшувати такі зміщення, якщо різні модальності компенсують одна одну. Проте вона може й посилювати нерівність, якщо ф'южн навчено на незбалансованих даних. Тому стратифіковане оцінювання має стати обов'язковою частиною протоколу.

Шостою відкритою проблемою є пояснюваність. Користувач, адміністратор або регулятор мають розуміти, чому система прийняла або відхилила рішення, особливо якщо наслідком є відмова в доступі до послуги. Для простих правил голосування пояснення очевидніше, але точність може бути нижчою. Для глибинного ф'южну точність часто вища, але механізм рішення складніший. Тому перспективними є довірчо-адаптивні моделі, які поєднують ф'южн із показниками якості, локальними поясненнями та захищеним обробленням [55].

3.2.9 ВИСНОВКИ

Мультимодальні біометричні системи є закономірним етапом розвитку біометричної автентифікації. Вони відповідають на обмеження одномодальних рішень, пов'язані з шумом, нестабільністю ознак, недоступністю окремих датчиків і вразливістю до атак пред'явлення. Поєднання кількох джерел може

зменшити невизначеність рішення, підвищити стійкість і забезпечити гнучкість у різних сценаріях застосування.

Разом з тим мультимодальність не слід розглядати як автоматичне вирішення всіх проблем. Додавання нових модальностей збільшує обсяг чутливих даних, ускладнює архітектуру, підвищує вимоги до ф'южну й створює нові поверхні атаки. Тому ефективна система повинна поєднувати розпізнавання з контролем якості, виявленням живості, захистом еталонів, аудитом і мінімізацією даних.

Найбільш практичним рівнем ф'южну для багатьох прикладних систем залишається рівень оцінок, оскільки він поєднує гнучкість і відносну прозорість. Водночас у системах із достатньою кількістю даних перспективними є ознаковий і гібридний ф'южн, особливо якщо вони враховують якість зразків і контекст. Вибір рівня ф'южну має залежати від сценарію, ризику операції, доступних датчиків і вимог до пояснюваності.

Оцінювання мультимодальної біометрії повинно охоплювати не лише точність, а й FAR, FRR, EER, криві робочих характеристик, показники атак пред'явлення, міжсесійне та міжбазове тестування, абляційний аналіз і перевірку справедливості. Без такого протоколу високі показники можуть бути наслідком перенавчання або особливостей конкретної бази даних, а не реальної надійності системи.

Отже, майбутній розвиток мультимодальної біометрії визначатиметься балансом між точністю, приватністю, зручністю, доступністю й довірою. Найперспективнішими є системи, які не лише поєднують кілька ознак, а й уміють оцінити їхню якість, захистити еталони, працювати з неповними даними, пояснювати рішення та адаптувати рівень перевірки до реального ризику.

DOI: 10.46299/ISG.2026.MONO.TECH.2.4.1

4.1 Current approaches to multimodal data storage in AI-driven information systems

Modern information systems face increasingly complex requirements for data storage and processing. The diversity of data types – structured tables, unstructured documents, complex networks of relationships – forces developers to go beyond a single database model and combine several models simultaneously. This approach is known as Polyglot Persistence [58, 59] and has become the standard in building scalable systems. The emergence of vector databases, which have become an integral part of modern AI systems, deserves special attention. Unlike traditional databases, which search for exact matches, vector databases [60] operate on mathematical representations of content – so-called embeddings – and allow for the retrieval of semantically similar data. It is precisely this property that makes them a key component in AI-based architectures.

The rapid development of large language models and transformer-based architectures (in particular, models for generating deep semantic embeddings) has radically changed the requirements for the design of information systems. Traditional approaches to systematic data analysis are increasingly confronted with the challenge of effectively combining strict relational transactional requirements with semantic search. In contemporary scientific literature, the issue of integrating vector stores and optimising Retrieval-Augmented Generation (RAG) architecture is explored in numerous works [61]. In particular, recent works have devoted significant attention to optimising RAG pipelines to improve information retrieval quality metrics (such as recall, precision and F1-score) in specialised domains [62].

However, a detailed analysis of the literature shows that most current studies focus on the isolated use of vector databases or consider exclusively the two-component interaction ‘Vector Database + LLM’ [63]. At the same time, the problem of complex orchestration of heterogeneous databases within the Polyglot Persistence paradigm in

high-load microservice architectures remains insufficiently addressed. For critical sectors (such as medical information systems), a fundamental scientific and practical contradiction arises: on the one hand, there is an urgent need to ensure strict ACID guarantees for structured data, and on the other, a need for flexible semantic search of unstructured data using approximate nearest neighbour (ANN) search algorithms. Resolving this contradiction requires the development and justification of comprehensive data integration patterns at the level of system architecture [64].

The aim of this work is to investigate the characteristics of integrating various database models – relational, document-oriented, graph and vector – into a single system, as well as to analyse their interaction in the context of building AI applications.

The choice of database model is one of the key architectural decisions when designing any information system. An incorrect choice can lead to significant problems with performance, scalability, or the complexity of maintaining the system in the future. To make an informed choice, it is necessary to understand the strengths and weaknesses of each model according to key criteria. The CAP theorem [65] provides the fundamental theoretical basis for understanding these constraints in distributed computing environments. This theorem states that any distributed system can simultaneously guarantee only two of the three basic properties: data consistency, system availability, and partition tolerance. Relational database models are traditionally geared towards ensuring consistency and availability (the CA model), deliberately sacrificing partition tolerance. In contrast, most non-relational (NoSQL) systems opt for availability and partition tolerance (the AP model), which allows them to scale easily [66]. At the same time, they allow for so-called ‘eventual consistency’ – a state of the system where data across different nodes in the cluster may temporarily diverge, but is guaranteed to synchronise over time. Understanding the principles of the CAP theorem is a critically important step in the design of modern distributed systems. This is because different database models within a single complex system can provide completely different consistency guarantees, which must be taken into account when developing integration logic at the application level.

To systematise and clearly illustrate the characteristics of each of the models under consideration, it is advisable to conduct a detailed comparative analysis of them based on the most important criteria. These criteria include data structure, schema flexibility, query language support, the ability to process ACID transactions, and scaling strategies. The results of the comparison are presented in Table 1.

Table 1.

Comparison of database models

Criterion	Criterion	Criterion	Criterion	Criterion
Data structure	Tables	JSON documents	Nodes and edges	Numeric vectors
Schema	Rigid	Flexible	Flexible	Fixed dimension
Query language	SQL	MQL / JSON queries	Cypher	ANN search
ACID transactions	Full support	Partial	Partial	None
Scaling	Vertical	Horizontal	Limited	Horizontal
Strength	Data integrity	Schema flexibility	Complex relationships	Semantic search
Weakness	Rigid schema	Complex relationships	Scaling	Similarity search only
Examples	PostgreSQL, MySQL	MongoDB, CouchDB	Neo4j, Neptune	Pinecone, Weaviate

The correct choice of a specific model is determined primarily by the nature of the data itself and the type of operations that the information system will perform most frequently. It is worth noting that there is no single universal solution – each model has its own clearly defined scope of application. Relational data models undoubtedly remain the optimal and reliable choice in those cases where data has a clear, unchanging structure, and transactional integrity and complete consistency are an absolute priority. Typical examples of such tasks include processing financial transactions, maintaining patient records in healthcare facilities, or managing personnel – all these processes are naturally reflected in a typical tabular model. Furthermore, if

a system makes intensive use of complex analytical queries involving multidimensional aggregation and data grouping, the standardized SQL language remains the most convenient and powerful tool.

On the other hand, *document-oriented databases* are the appropriate choice when the data structure is irregular, dynamic, or subject to frequent changes during system development. If different records belonging to the same entity have different sets of attributes, the flexible document model is the most natural and logical choice. This model performs significantly better in high-load scenarios with intensive write and read traffic, where rapid horizontal scaling of the cluster (distribution across nodes) is critical.

Graph databases are indispensable in situations where the primary value of an information system lies not so much in the entities themselves as in the complex network of relationships between them. Building recommendation systems, fraud detection algorithms, constructing deep medical ontologies, or analysing social graphs – all these specific tasks require efficient traversal of complex networks of relationships. Graph databases handle such operations orders of magnitude more efficiently than their classical relational counterparts.

At the same time, *vector data models* are an essential and indispensable component of any modern system that utilises artificial intelligence and requires semantic content-based search. It is precisely scenarios involving intelligent semantic search across large document repositories, the search for visually similar images, the generation of recommendations based on implicit preferences, and the creation of AI assistants capable of ‘remembering the context of a conversation’ that can be successfully implemented using a vector store.

To support the adoption of sound architectural decisions, we will compare the traditional SQL approach with a range of NoSQL alternatives. The results of the comparison are shown in Table 2.

Table 2.

Comparison of SQL and NoSQL approaches: use cases

Aspect	SQL (Relational)	NoSQL (Non-relational)
Data model	Structured, tabular with a fixed schema and relationships	Flexible data models (document, key-value, columnar, graph)
Schema	Rigid; must be defined in advance	Schema-less or flexible; supports dynamic and unstructured data
Scaling	Vertical (increasing server capacity)	Horizontal (distribution across nodes)
Consistency	Strong consistency via ACID transactions	Eventual consistency, although managed in some systems (e.g. MongoDB)
Query language	SQL (standardised, powerful for complex queries)	Varies (e.g. MongoDB query language, Cassandra Query Language)
Performance	Optimised for complex queries and joins, slower with large write volumes	Optimised for high read/write throughput, particularly in distributed systems
Use cases	Best for structured data with strong consistency (financial systems, ERP, CRM)	Best suited for unstructured data and large-scale distributed systems (Big Data, analytics, social networks)
Scaling complexity	Difficult to shard across servers	Designed for easy sharding and distribution
Data integrity	High, ensured through constraints and relationships	Varies; integrity is often ensured at the application level
Relationships and JOINs	Supports complex JOINs between tables and foreign keys	Relationships are either built-in (documents) or handled differently (e.g. graph databases)
Examples	MySQL, PostgreSQL, Oracle, SQL Server	MongoDB, Cassandra, DynamoDB, Redis, Neo4j
Examples	PostgreSQL, MySQL	MongoDB, CouchDB

It is important to understand that in real-world industrial systems, these approaches do not compete with one another in any way; on the contrary, they effectively complement one another. It is precisely this logical interaction and complementarity that underpin the concept of Polyglot Persistence, which serves as the modern standard in the development of complex distributed information systems.

The term Polyglot Persistence was introduced by M. Fowler and P. Sadalage in the book "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence" [66]. The concept is that different parts of a single system can and should use different types of data models – each where they are most effective. The name of the term is an analogy to the concept of 'polyglot programming' – the use of different programming languages in a single project. Figure 1 illustrates a typical Polyglot Persistence architecture, where each microservice/system component interacts with the type of database that best suits its needs. It is important to understand that this is not a haphazard use of different technologies, but a deliberate architectural decision with a clear rationale for each choice.

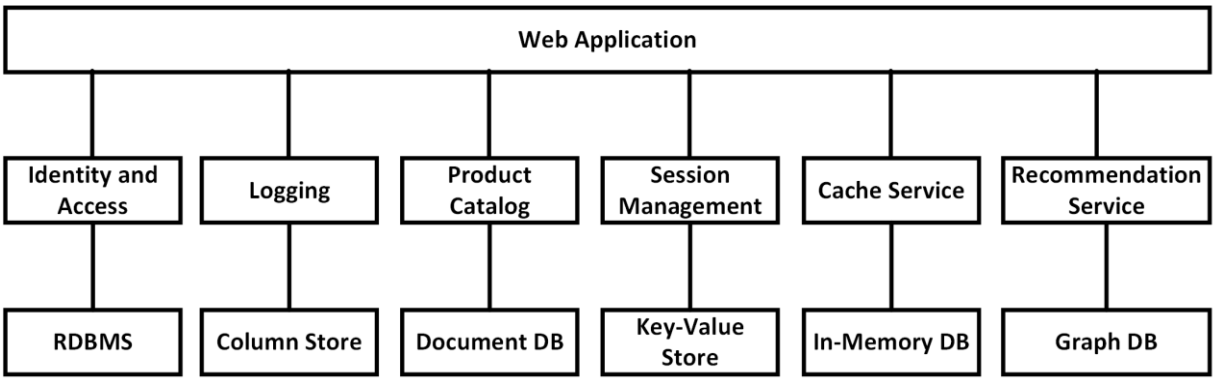


Figure 1. Polyglot Persistence architecture

Source of the figure: [59].

The core idea of Polyglot Persistence aligns well with microservices architecture – an approach where the system is broken down into small, independent services. Each service owns its own data and selects the most suitable storage. For instance, the authentication service uses a relational database model to store user accounts. The directory service uses a document-oriented model for flexible storage of heterogeneous

records. The recommendation service uses a graph database model for analysing relationships, whilst the AI search service uses a vector database for semantic search [67]. Currently, there are several patterns for organising interaction between different database models within a single system. The choice of pattern depends on requirements regarding latency, data consistency and system complexity.

- ETL (Extract, Transform, Load) is the simplest and oldest integration pattern. Data is extracted from one database, transformed into the required format, and loaded into another. ETL is usually performed on a schedule – for example, every night. This approach is simple to implement but has a significant drawback: data between synchronisations may be out of date. For a medical system, this can be critical – if a vector database is synchronised with MongoDB once a day, the AI search module will not be aware of discharge records created today.
- CDC (Change Data Capture) is a more modern approach that tracks changes in the database in real time. Instead of copying all data on a schedule, CDC records every insert, update or delete operation and instantly broadcasts it to other systems. This ensures near-real-time synchronisation between individual repositories.
- Event-Driven Architecture is an architectural approach where all data changes are published as events in a message queue. Each database subscribes to events relevant to it and updates its data accordingly. This approach ensures loose coupling between components, meaning that no single database is directly aware of the existence of others.
- API Gateway – a pattern where all requests to different databases pass through a single entry point. The gateway receives the request, determines which database is the source of the required data, executes the query and returns the result. If necessary, the gateway can aggregate data from several sources in a specific sequence.

The most complex challenge when integrating multiple databases is ensuring data consistency. In a monolithic system with a single relational database, a transaction is either executed in full or rolled back – the ACID principles resolve this issue. In a multi-model system, a transaction may span several different stores, each with its own set of guarantees. The Saga pattern is used to solve this problem. Instead of a single

distributed transaction, Saga breaks down a complex operation into a sequence of local transactions in each database separately. If an error occurs at any step, compensating transactions are triggered – operations that undo the steps already performed. For example, when saving a new medical report: first, the document is written to MongoDB, then the status is updated in PostgreSQL, and then the vector is indexed in Weaviate. If indexing in Weaviate fails, compensating transactions delete the document from MongoDB and undo the update in PostgreSQL.

Another issue is data duplication. In a multi-model system, the same data can be stored in several databases, but in different formats. For example, a patient's name appears in both PostgreSQL and in MongoDB documents containing discharge summaries. When a name changes, the data must be updated simultaneously in both places, which requires the use of synchronisation mechanisms. For the practical implementation of the patterns described, there are a number of specialised tools that simplify the integration of heterogeneous database models:

- Apache Kafka is a distributed platform for streaming event processing and is the de facto standard for building event-driven architectures. Kafka acts as a reliable message bus between various database models and services. It guarantees the durability and ordered delivery of messages even when consumers are temporarily unavailable [68, 69].

- Debezium is a CDC implementation tool that connects to the transaction logs of relational databases and publishes all changes to Kafka. Debezium supports PostgreSQL, MySQL, MongoDB and other popular systems. The combination of Debezium and Kafka is the most common way to ensure real-time synchronisation between different database models.

- Apache Spark is a framework for distributed processing of large volumes of data, often used to implement ETL processes between heterogeneous data stores. Spark supports reading and writing data to and from relational databases, MongoDB, and file stores [70, 71].

However, with the emergence of large language models and machine learning systems, the requirements for data storage infrastructure have changed radically.

Traditional approaches to information retrieval, based on exact keyword matching, have proved insufficient for artificial intelligence tasks. It is vector databases that have provided the answer to this challenge and have become an integral component of modern AI systems.

The central concept of vector databases is embedding – a numerical vector representation of an object in a multidimensional space. The process of creating an embedding involves a specialised neural network – a so-called encoder model – taking an arbitrary object (text, image, audio) as input and returning a vector of fixed dimension that encodes the semantic content of that object. Modern models generate vectors with dimensions ranging from 384 to 1536 numerical values.

A key property of embeddings is that semantically similar objects receive mathematically close vectors. The texts “the patient complains of a headache” and “the patient experiences pain in the head” will have vectors that are close to one another in vector space – despite the fact that there is not a single common word of meaning between them. It is precisely this property that forms the basis of semantic search.

Mathematical metrics are used to measure the similarity between vectors. Cosine similarity measures the angle between two vectors – the smaller the angle, the more similar the objects. Euclidean distance measures the geometric distance between points in space. It should be noted that cosine similarity is the most common metric for text embeddings, as it does not depend on the length of the text. The cosine similarity between two vectors is determined as follows:

$$\cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \cdot \|\vec{B}\|}$$

Where, $\vec{A}$ and $\vec{B}$ - are embedding vectors;

$\vec{A} \cdot \vec{B}$ - the dot product of the vectors;

$\|\vec{A}\|, \|\vec{B}\|$ - the Euclidean norms of the vectors.

The cosine similarity value lies in the range $[-1,1]$, where values close to 1 indicate a high degree of semantic similarity between objects.

The search for nearest neighbours in vector space (Approximate Nearest Neighbour, ANN) is a fundamental operation in vector databases. A traditional exact search requires calculating the distance between the query and every vector in the system, which is computationally infeasible for large datasets. Instead, when using ANN, upon receiving a query, it is instantly converted into a numerical vector, after which the database finds the k vectors closest to it in the storage, sacrificing absolute accuracy for the sake of speed. The task of finding the nearest neighbours can be formalised as finding a set of k vectors that minimise the distance to the query:

$$argmin_{x_i \in D}\, d(q, x_i), \qquad i = 1, \dots, k$$

where, q is the query vector;

D is the set of vectors in the database;

$d(q, x_i)$ - the distance function (e.g., cosine or Euclidean);

k is the number of nearest neighbours.

In the case of ANNs, the problem is solved approximately, which allows for a significant reduction in the computational complexity of the search with a negligible loss of accuracy. Special spatial index structures are used to perform this operation efficiently on sets of vectors. In particular, one of the most modern is the HNSW (Hierarchical Navigable Small World) algorithm, which constructs a multi-level navigable graph and ensures logarithmic search time.

To objectively assess the effectiveness of integrating vector databases into the complex architecture of an information system, it is advisable to rely on quantitative metrics of performance and the quality of semantic search. The use of optimised encoder models in combination with approximate search algorithms (ANN) allows the necessary balance to be achieved between latency and the relevance of results. The application of such approaches reduces the processing delay for complex semantic queries, which is a critical metric for high-load microservice systems. At the same time, despite the lack of an exact match, the quality of searching for relevant documents remains high. The effectiveness of finding semantically similar vectors (for example, similar clinical cases) is traditionally assessed using the metrics of precision, recall,

and their harmonic mean – the F1-score. Formally, the F1-score is defined as the harmonic mean between precision and recall:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

where, $Precision$ is the proportion of relevant items among those found;

$Recall$ - the proportion of relevant items found out of all relevant items.

For the sake of completeness, we note that:

$$Precision = \frac{TP}{TP+FP}, \qquad Recall = \frac{TP}{TP+FN}$$

де, TP - true-positive results;

FP - false positives;

FN - false negatives.

The F1-score metric provides a balanced assessment of model quality in cases where both precision and recall must be taken into account simultaneously. The use of the F1-score is particularly appropriate in information retrieval tasks and RAG systems, where striking a balance between the comprehensiveness of the context provided and its relevance is crucial. The practical implementation of modern encoders in specialised domains demonstrates the ability to maintain the F1-score at a consistently high level (above 0.85–0.90), which guarantees high-quality context passed to generative models in the RAG architecture.

RAG (Retrieval-Augmented Generation) is a modern architectural pattern that has become the de facto standard for building AI systems based on large language models. The main problem that RAG solves lies in the limited context of neural networks: large language models are trained on publicly available datasets and, by design, do not possess the closed, specific information of a particular organisation – such as patients' personal medical records, internal treatment protocols or commercial knowledge bases. Furthermore, the knowledge of the base model is fixed at the moment its training is completed. RAG circumvents these limitations by allowing up-to-date contextual information to be added to the language model directly during response generation. Figure 2 shows the complete RAG pipeline [63].

For example, in a medical system, the RAG architecture allows a doctor to ask a query such as ‘what treatment methods have been used for patients with similar symptoms?’ and receive a response based on real data from a medical records database, rather than on the model’s general knowledge.

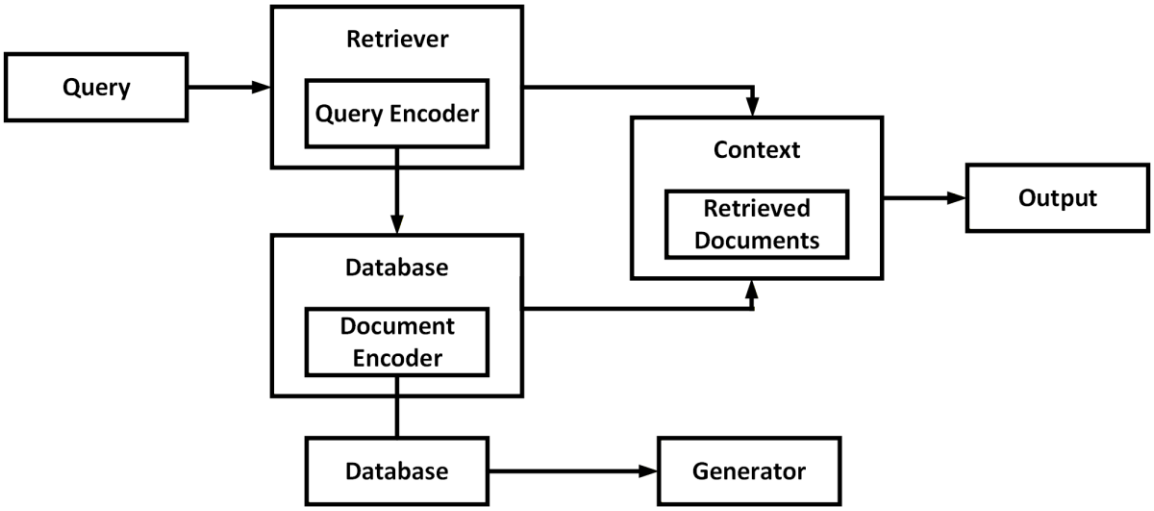


Figure 2. System-level RAGPipeline

Source of the figure: [63].

A vector database model rarely exists in isolation – it is almost always part of a larger multimodal system. There are several typical patterns for its integration with other types of storage. The most common is the “dual-entry” pattern: when a new document is saved in MongoDB, the system simultaneously generates its embedding and stores the vector in Weaviate. During a search, a vector query is first executed to return the identifiers of relevant documents, and then the full documents are retrieved from MongoDB using these identifiers. In this way, the vector database is responsible only for searching, whilst MongoDB handles the storage of the full content.

Another interesting solution is the pgvector extension for PostgreSQL, which adds support for vector search directly to the relational database. This allows vectors to be stored as a regular table column and enables hybrid queries that combine traditional SQL filtering with vector search. For example: finding medical records of patients over 50 years old (SQL filter) with similar symptoms (vector search) – in a single query.

Figure 3 illustrates how pgvector extends the capabilities of PostgreSQL by enabling hybrid queries.

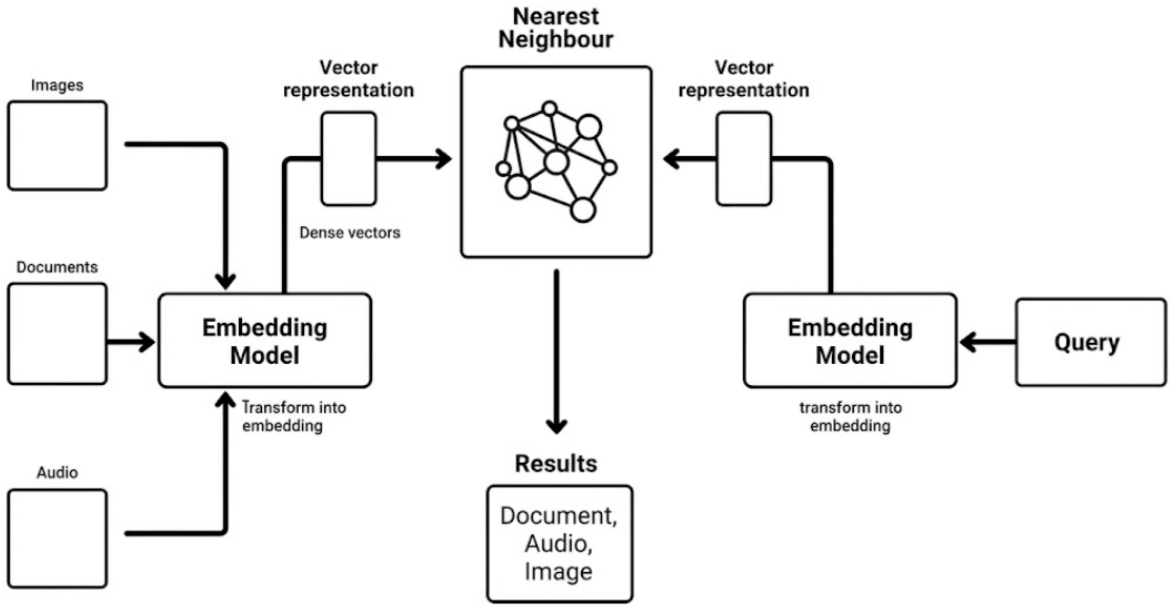


Figure 3. Hybrid search with pgvector

Source of the figure: [67].

Another important integration scenario is the combination of graph and vector database models. In particular, Neo4j has native support for vector indexing. This allows queries that simultaneously traverse the graph of relationships and search for semantically similar nodes – for example, finding all diseases associated with a specific symptom (graph traversal) that have a description similar to a given text (vector search). To demonstrate the concepts discussed, let us consider the architecture of an abstract medical information system. This system automates hospital operations – from patient registration to AI-assisted diagnosis. It is precisely the diversity of tasks in such a system that makes it an ideal example for the application of Polyglot Persistence (Fig. 4). All requests from doctors and nurses arrive via a single-entry point – the API Gateway, which routes them to the relevant service. Each service is responsible for its own domain and works with only one database. The central element of the integration is Apache Kafka – a message bus through which all services exchange events in real

time. Each database in the system is responsible for a clearly defined data domain for which it is best suited.

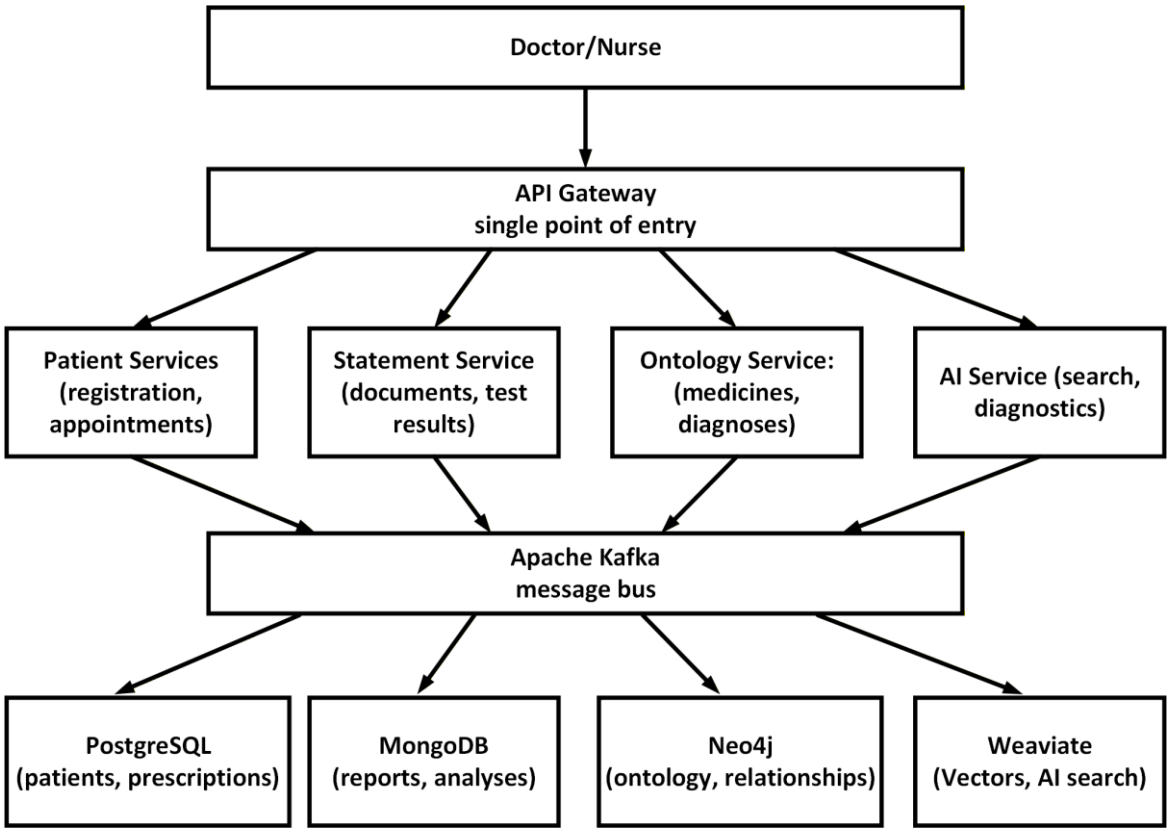


Figure 4. System architecture: integration via Apache Kafka.

Source of the figure: author's development.

PostgreSQL stores all structured and mission-critical data: information about patients, doctors, appointment schedules, prescriptions and payments. This is where ACID transactions are essential – we cannot allow a situation where a prescription is issued but the appointment record is not saved.

MongoDB stores medical discharge summaries and test results as documents. Each discharge summary has an arbitrary structure – a surgical operation, a routine check-up and a laboratory test have completely different sets of fields. The document model ensures efficiency for such irregular information. Neo4j stores the medical ontology: the relationships between diseases, symptoms and medications. When a doctor prescribes medication, the system queries Neo4j to check for any contraindications given the patient’s other diagnoses.

Weaviate stores vector representations of all medical records. This allows the AI assistant to find similar clinical cases across the entire patient database based on a free-form description of symptoms.

Let's look at the workflows in more detail. Workflow 1 – the doctor concludes the consultation and saves the discharge summary. The doctor completes the discharge summary in the interface. A request via the API Gateway reaches the discharge summary service, which stores the document in MongoDB. Debezium tracks this change and instantly publishes the event to Kafka. Next, two services simultaneously receive this event: the patient service updates the appointment status in PostgreSQL, whilst the AI service generates a vector representation of the discharge summary and stores it in Weaviate. In this way, a single action by the doctor automatically updates three different databases without any manual intervention.

Workflow 2 – a doctor prescribes a new medication. Before saving the prescription, the ontology service sends a query to Neo4j – checking the graph of relationships between the medication and all of the patient's diagnoses. If Neo4j returns any contraindications, the system alerts the doctor before the prescription is saved. If everything is in order, the prescription is saved in PostgreSQL and the event is sent back to Kafka for synchronisation. It is precisely this division of responsibilities between the four databases via a single message bus that constitutes a practical implementation of the Polyglot Persistence concept.

In summary, this paper, in accordance with its stated objective, analyses modern approaches to organising data storage in AI-oriented information systems and demonstrates the limitations of traditional monomodel solutions in the context of increasing data heterogeneity. The paper demonstrates the feasibility of using the Polyglot Persistence paradigm, which ensures the effective combination of different database models in accordance with the specifics of the tasks.

Particular attention is paid to the integration of vector databases as a key component of modern AI systems and RAG architectures. It has been established that the use of embeddings and approximate search algorithms ensures effective semantic search, whilst the use of Precision, Recall and F1-score metrics allows for an objective

assessment of the quality of the results obtained. The main patterns of data integration in multimodal environments have been systematised, and mechanisms for ensuring consistency have been analysed, in particular using the Saga approach. The proposed information system architecture demonstrates the practical implementation of these approaches and confirms their effectiveness in complex application scenarios.

The results obtained demonstrate the feasibility of integrating multimodal databases in combination with AI technologies as a basis for building modern scalable information systems and identify prospects for further research in the direction of optimising RAG architectures and semantic search.

DOI: 10.46299/ISG.2026.MONO.TECH.2.4.2

4.2 Assessment of critical components of autonomous systems using variable parametric identification

The modern development of autonomous systems and long-range unmanned aerial vehicles (UAVs) places stringent demands on their reliability and survivability. One of the main causes of failure of power plants and onboard avionics is a violation of the thermal operating conditions. Under conditions of changing atmospheric density and dynamic loads, traditional temperature monitoring methods are ineffective, as they do not take into account the inertia of thermal processes and the noise level of sensor signals.

To ensure predictive or forecast flight safety, a pressing task is the development of computational models that enable real-time parametric identification of the thermal state of critical components of unmanned aerial vehicles. Parametric identification in technical objects involves obtaining, based on experimental data, optimal estimates of a certain vector of sought-after heat transfer parameters included in the mathematical model of the latter [72]. However, the accuracy of such identification directly depends on the methodology of experimental design and the assessment of heat transfer identification errors. This paper examines a methodology for planning parametric identification for monitoring the thermal state of a UAV power plant and, if necessary, other components of unmanned aerial vehicles. The study builds on the author's previous work on metrological support for thermophysical experiments and digital processing of temperature transducer measurement signals.

Inverse heat conduction problems (IHCPs) form the basis of modern information technologies for experimental and computational studies of heat transfer processes in technical objects (TOs), which determine their thermal state.

Interest in IHCPs is constantly growing, driven by both practical needs and the rapid development of computer technology methods and tools. For example, in thermal power engineering, including heat engines, IHCP methods are widely used both in experimental studies of the thermal state of working fluids and structural elements

(IHCP diagnostics) and in the optimal design of the latter (IHCP optimization).

In diagnostics, boundary thermal management techniques are most common, aimed at determining local heat fluxes and resistances, heat transfer coefficients, etc., on the surface of the vehicle. Separate classes of thermal management techniques include coefficient-based techniques, aimed at determining the thermophysical characteristics of the material, geometric techniques, and combined or complex techniques, in which disparate thermal quantities are simultaneously determined. It should be emphasized that the tasks of constructing (identifying) mathematical models that adequately describe heat transfer in the vehicle also relate to thermal management techniques.

The reliability of long-endurance unmanned aerial vehicles (UAVs) is inherently linked to the thermal stability of their propulsion and avionic systems. Recent studies emphasize that effective thermal management is a cornerstone for ensuring mission success in autonomous platforms. For example, Smith et al. [73] provide a comprehensive review of thermal management systems for high-altitude long-endurance UAVs, highlighting that traditional cooling methods often struggle with the dynamic heat loads encountered during complex flight profiles. To address real-time monitoring challenges, various estimation techniques have been proposed. Gao et al. [74] demonstrated the efficacy of the Extended Kalman Filter (EKF) for robust thermal state estimation in electric propulsion systems, focusing on the nonlinear heat dissipation of motors. Similarly, the problem of sensor reliability in harsh environments—where vibration and electromagnetic interference are prevalent—was addressed by Zhang and Wei [75]. They proposed adaptive sensor fusion algorithms that enhance fault diagnosis capabilities, though their approach primarily focuses on high-level system failures rather than deep parametric identification of heat transfer processes.

The accuracy of onboard sensors remains a critical bottleneck. Wang [76] explored Kalman filter-based error compensation specifically for high-temperature sensors in aerospace applications, proving that stochastic noise filtering is essential for meaningful diagnostics. However, as noted by Liu et al. [77], the mere filtering of

signals is often insufficient without a precise underlying mathematical model. Their work on parametric identification of thermal models for avionic components emphasizes the need for high-fidelity identification techniques in autonomous systems.

Despite these advances, a significant gap remains in the methodological planning of the identification process itself. While existing literature focuses on state estimation, there is a lack of focus on the quantitative assessment of identification accuracy through the Fisher information matrix and joint confidence intervals (JCI) in the context of UAV diagnostics. This paper aims to bridge this gap by integrating a discrete-recursive thermal model with a precision-planned Kalman filtering framework, building upon established methods of parametric identification to enhance the thermal resilience of autonomous aerial platforms.

Unlike conventional monitoring systems, the proposed method is based on the analysis of the Fisher information matrix, which is constructed from the sensitivity functions of measured temperatures to the desired parameters. In the work [72], which is in a sense a generalization for a number of long-term studies conducted at the National Aerospace University “Kharkiv Aviation Institute”, an approach to assessing the accuracy of results and planning parametric identification is proposed, based on the analysis of the Gram matrix.

The Gram matrix consists of sensitivity functions of the sought parameters to the obtained measurements and is analogous to the Fisher information matrix. The Gram matrix is used [78] to evaluate the accuracy of the obtained measurement results. This allows for the accuracy of each sought parameter to be considered separately.

As indicated in the monograph [79], the most effective approach to solving the heat transfer problem is an extremal one, which consists of minimizing quadratic objective functionals. This approach can be implemented as a method of parametric identification of heat transfer in heat transfer, which involves obtaining optimal estimates of a certain vector of sought heat transfer parameters, included in its mathematical model, based on experimental data.

It is known that heat transfer problems are ill-posed problems in mathematical physics, which are characterized by the possibility of solution instability. Therefore,

issues of the reliability and accuracy of the solutions obtained, particularly those obtained using the parametric identification method, are of paramount importance.

Technical literature has repeatedly emphasized that parametric identification of heat transfer is the most complex type of indirect measurement, in which mathematical models of heat transfer in heat transfer serve as the measurement equations. This circumstance, in accordance with the requirements for measurement tools and methods, also leads to the need to establish quantitative indicators of the accuracy of IHT solutions.

For inverse heat conduction problems (IHT), [72] proposed using joint confidence regions (JCRs) of the obtained solutions as indicators of parametric identification accuracy, as well as joint confidence intervals (JCIs) of the parameter estimates, defined as projections of the JCRs onto the axes of the multidimensional space of the estimated parameters. It has been shown that JCIs are practical indicators, particularly effective for cases with a significant number of unknown parameters.

Let us consider the basic principles of the parametric identification method for heat transfer in heat transfer, with the aim of obtaining the above-mentioned accuracy indicators. We will assume that there is an adequate mathematical model of heat transfer in a technical object (hereinafter referred to as the model), which allows for the moments of time $\tau_k=k\cdot\Delta\tau$ ($k = 1, 2, \dots n$) to calculate the temperature values t_{zk} at z=1, 2, …l points, which make up the $(l\times 1)$-vector $\mathbf{T}_k = [t_{zk}]_{z=1}^{l}$ of the temperature state of the object.

We will also assume that the $(r\times 1)$-vector $\mathbf{\Theta} = [\theta_j]_{j=1}^{r}$ of the sought-after heat transfer parameters can be identified in the heat transfer model. Its components θ_j must either be constant or can be approximated based on known functions of time with constant unknown (desired) coefficients, i.e., satisfying the condition Θ=const. The procedure for identifying Θ in the heat transfer model will be called the parameterization of inverse heat conduction problems.

We assume that the experiment measures temperatures at $m\leq l$ points of the object or their linear combinations, constituting the $(m\times 1)$-measurement vector $\mathbf{Y}_k = [y_{ik}]_{i=1}^{m}$,

whose relationship with the object's temperature vector $\mathbf{T}_k$ is described by the measurement model.

$$\mathbf{Y}_k=\mathbf{C}\ \mathbf{T}_k + \boldsymbol{\varepsilon}_k, \quad (1)$$

where C – the $(l \times m)$-dimension measurement matrix;

$\boldsymbol{\varepsilon}_k = [\varepsilon_{ik}]_{i=1}^{m}$ – the $(m \times 1)$-dimension vector of random errors or noise in measurements.

Measurement noise is known to be one of the factors that can cause instability in the solution of inverse heat conduction problems. Therefore, taking its influence into account at the stages of formulating and solving the inverse heat conduction problem seems essential. In this case, we will use the assumption, generally accepted for temperature measurements, that the components ε_{ik} of the vector ε_k in model (1) are normally distributed random variables with zero mathematical expectations, the same variance σ^2, and are uncorrelated with each other. This assumption seems entirely justified for homogeneous temperature measurements performed using the same primary and recording transducers. This allows us to represent the main characteristic of the random vector σ^2 its covariance (m m)-matrix R as

$$\mathbf{R}=\sigma^2\ \mathbf{I}\ , \quad (2)$$

where $\mathbf{I}$ – the identity $(m \times m)$- dimension matrix.

We will also assume that, using the heat transfer model in the heat transfer system, it is possible to calculate forecasts (analogues) of the measurement vector $\hat{\mathbf{Y}}_k(\boldsymbol{\Theta}) = [y_{ik}(\boldsymbol{\Theta})]_{i=1}^{m}$ depending on the vector of sought parameters $\boldsymbol{\Theta}$ at $\varepsilon=0$.

In this case, parametric identification consists of determining optimal estimates $\hat{\boldsymbol{\Theta}}$ of the vector $\boldsymbol{\Theta}$ based on n values of the measurement vector $\mathbf{Y}_k$ in model (1) and the heat transfer model in the heat transfer in TO. For these purposes, the most common method is minimization of the following quadratic residual function with respect to Θ [78]:

$$\mathbf{\Phi}(\mathbf{\Theta}) = \sum_{k=1}^{n} \left[\mathbf{Y}_k - \hat{\mathbf{Y}}_k(\mathbf{\Theta})\right]^{\mathrm{T}} \mathbf{R}^{-1} \left[\mathbf{Y}_k - \hat{\mathbf{Y}}_k(\mathbf{\Theta})\right]. \quad (3)$$

Thus, parametric identification of heat transfer in heat transfer systems can be reduced to the generalized least squares (GLS) method [78], which has been well studied in the mathematical and technical literature. Its advantage is that it does not require a priori knowledge of the statistical properties of the initial estimates $\hat{\mathbf{\Theta}}_0$, yields unbiased estimates $\hat{\mathbf{\Theta}}$, and ensures minimal variance of the estimates if the model is linear and the measurement noise is normally distributed.

It is known that when minimizing the residual function (3), the possibility of obtaining unstable solutions cannot be ruled out. This problem is one of the fundamental ones in the theory of heat transfer systems and is beyond the scope of this study. Therefore, we will assume that, using minimization procedures, stable optimal or near-optimal estimates $\hat{\mathbf{\Theta}}$ can be obtained, the errors of which can be estimated within the framework of GLS theory.

In the case where there is a linear dependence $\hat{\mathbf{Y}}_k(\mathbf{\Theta})$ on the vector of the sought parameters Θ, the following well-known solution of the linear GLS method for the estimates $\hat{\mathbf{\Theta}}$ and the covariance ($m \times m$) matrix P of the errors of these estimates can be used [73].

$$\hat{\mathbf{\Theta}} = \mathbf{P} \sum_{k=1}^{n} \left(\frac{\partial \vec{Y}_k}{\partial \mathbf{\Theta}} \right)^{\mathrm{T}}_{\hat{\mathbf{\Theta}}} \mathbf{R}^{-1} \mathbf{Y}_k; \quad (4)$$

$$\mathbf{P} = \mathbf{A}^{-1} = \left[\sum_{k=1}^{n} \left(\frac{\partial \mathbf{Y}_k}{\partial \mathbf{\Theta}} \right)^{\mathrm{T}}_{\hat{\mathbf{\Theta}}} \mathbf{R}^{-1} \left(\frac{\partial \mathbf{Y}_k}{\partial \vec{\mathbf{\Theta}}} \right)_{\hat{\mathbf{\Theta}}} \right]^{-1}. \quad (5)$$

If the noise covariance matrix **R** satisfies condition (2), then the solution (4)–(5) is transformed to the form

$$\hat{\mathbf{\Theta}} = \mathbf{P} \frac{1}{\sigma^2} \sum_{k=1}^{n} \mathbf{H}_k^{\mathrm{T}} \mathbf{Y}_k; \quad (6)$$

$$\mathbf{P} = \sigma^2 \mathbf{A}^{-1} = \sigma^2 \left[\sum_{k=1}^{n} \mathbf{H}_k^{\mathrm{T}} \mathbf{H}_k \right]^{-1}, \quad (7)$$

where $\mathbf{H}_k = \left[\frac{\partial \mathbf{Y}_k}{\partial \boldsymbol{\Theta}} \right]_{\hat{\boldsymbol{\Theta}}}$ – the $(m \times r)$-matrix of sensitivity functions;

$u_{ijk} = \frac{\partial y_{ik}}{\partial \theta_j}$ – the functions of sensitivity of the components of the measurement vector to the sought parameters;

y_{ik} – the components of the measurement vector;

vector $\boldsymbol{\theta}_j$ – the vector of the sought parameters (i=1, 2,…, m; j=1, 2,…, r; k=1,2,…, n), where the vector of the sought parameters is calculated for the values $\hat{\boldsymbol{\Theta}}$ at the k-th moment of time;

$\mathbf{A} = \left[\sum_{k=1}^{n} \mathbf{H}_k^{\mathrm{T}} \mathbf{H}_k \right]$ – the $(r \times r)$ Gram matrix of sensitivity functions (also known as Fisher information matrix).

The matrix of sensitivity functions Hk can be written in the form

$$\mathbf{H}_k = \begin{bmatrix} u_{11k} & u_{12k} & \dots & u_{1rk} \\ u_{21k} & u_{22k} & \dots & u_{2rk} \\ \dots & \dots & \dots & \dots \\ u_{m1k} & u_{m2k} & \dots & u_{mrk} \end{bmatrix}_{\hat{\boldsymbol{\Theta}}}. \quad (8)$$

Then the Gram matrix **A** in accordance with (6)–(8) can be represented as

$$\mathbf{A} = \begin{bmatrix} \sum_{k=1}^{n} \sum_{i=1}^{m} u_{i1k}^2 & \dots & \sum_{k=1}^{n} \sum_{i=1}^{m} u_{i1k} u_{irk} \\ \dots & \dots & \dots \\ \sum_{k=1}^{n} \sum_{i=1}^{m} u_{irk} u_{i1k} & \dots & \sum_{k=1}^{n} \sum_{i=1}^{m} u_{irk}^2 \end{bmatrix} = \begin{bmatrix} a_{11} & \dots & a_{1r} \\ \dots & \dots & \dots \\ a_{r1} & \dots & a_{rr} \end{bmatrix}. \quad (9)$$

In the theory of the linear least squares method, it has been shown [78] that under the adopted assumptions (2) about the noise vector ε_k, the estimates $\hat{\boldsymbol{\Theta}}$ are optimal – unbiased, efficient, sufficient, and consistent.

We will use the inverse Gram matrix A^{-1}, which can be obtained from (9) using a standard computational procedure, in the form

$$\mathbf{A}^{-1} = \begin{bmatrix} a_{11}^* & \dots & a_{r1}^* \\ \dots & \dots & \dots \\ a_{r1}^* & \dots & a_{rr}^* \end{bmatrix}. \quad (10)$$

The direct A and inverse A^{-1} Gram matrices are the basis for constructing the JCR and JCI.

JCR is usually understood [78] as a quadratic form

$$(\mathbf{\Theta}_0 - \hat{\mathbf{\Theta}})^{\mathrm{T}} \mathbf{A}(\mathbf{\Theta}_0 - \hat{\mathbf{\Theta}}) = \sigma^2 \mathrm{B} \quad (11)$$

in the space of the sought parameters $\mathbf{\Theta}$, describing in the neighborhood of the obtained estimates $\hat{\mathbf{\Theta}}$ an *r*-dimensional ellipsoid, which with a confidence probability ν contains the true values θ_0 of the vector $\mathbf{\Theta}$. In (11)

$$\mathrm{B}=r\mathrm{F}_{\nu}(r,\, n-r),$$

where $\mathrm{F}_{\nu}(r,\, n-r)$ – the tabular values of the Fisher distribution quantile for *r* parameters and *n* measurements in the evaluation area.

Let's consider several possible cases of constructing JCR based on a known Gram matrix **A** and the values $\sigma^2\mathrm{B}$ for the experiment conducted.

In the most illustrative case of two estimated parameters (r=2), denoting

$$\mathbf{\Theta}_0 - \hat{\mathbf{\Theta}} = \begin{bmatrix} \Delta\hat{\theta}_1 \\ \Delta\hat{\theta}_2 \end{bmatrix},$$

where $\Delta\hat{\theta}_1 = \theta_{1,0} - \hat{\theta}_1$ and $\Delta\hat{\theta}_2 = \theta_{2,0} - \hat{\theta}_2$, we obtain from (11) the following equation, which describes the ellipse of the joint confidence region in the error space $\Delta\hat{\theta}_1$ and $\Delta\hat{\theta}_2$ with the center at the point $\Delta\hat{\theta}_1 = \Delta\hat{\theta}_2 = 0$:

$$a_{11}(\Delta\hat{\theta}_1)^2 + 2a_{12}\Delta\hat{\theta}_1\Delta\hat{\theta}_2 + a_{22}(\Delta\hat{\theta}_2)^2 = B\sigma^2. \quad (12)$$

To construct an ellipse, one of the two equations that follow from (12) can be used:

$$(\Delta\hat{\theta}_1)_{1,2} = \frac{-a_{12}\Delta\hat{\theta}_2 \pm \sqrt{a_{12}(\Delta\hat{\theta}_2)^2 - a_{11}[a_{22}(\Delta\hat{\theta}_2)^2 - B\sigma^2]}}{a_{11}}; \quad (13)$$

$$(\Delta\hat{\theta}_2)_{1,2} = \frac{-a_{12}\Delta\hat{\theta}_1 \pm \sqrt{a_{12}(\Delta\hat{\theta}_1)^2 - a_{22}[a_{11}(\Delta\hat{\theta}_1)^2 - B\sigma^2]}}{a_{22}}. \quad (14)$$

For example, by substituting one coordinate $\Delta\hat{\theta}_1$ into equation (14), we obtain two coordinates $\left(\Delta\hat{\theta}_2\right)_{1,2}$ for two points of the ellipse, and so on.

The form of the dynamic range equation for the case r=2 is shown in Fig. 1.

In the case of one (r=1) estimated parameter θ_1, if we introduce the notation $\Delta\hat{\theta}_1 = \theta_{1,0} - \hat{\theta}_1$, then the SDO degenerates to a straight line segment bounded by the values $\Delta\hat{\theta}_1 = \theta_{1,0} - \hat{\theta}_1$. The latter are determined from the equation

$$(\Delta\hat{\theta}_1)^2 \sum_{k=1}^{n}\sum_{i=1}^{m} u_{i1k}^2 = \sigma^2 F_{\nu}(1, n-1),$$

which follows from (11) taking into account (9).

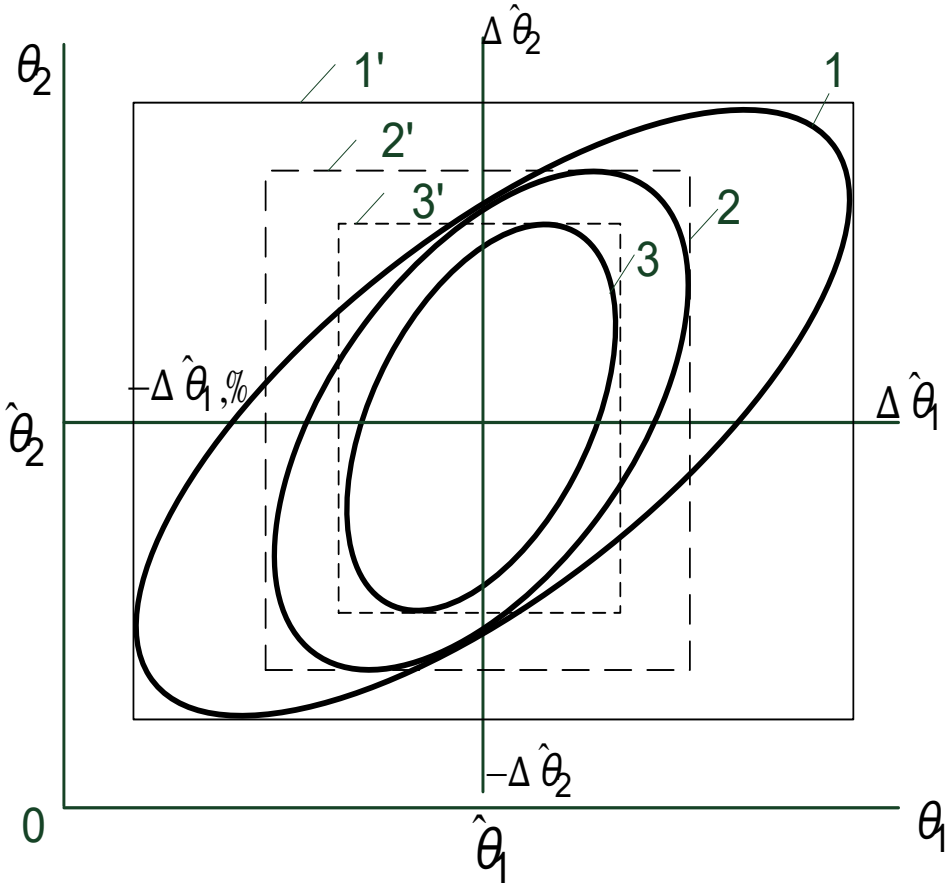


Figure 1. Elliptical JCR (1-3) and rectangular regions (1′-3′) of estimates described around them, formed by two pairs of boundary values of the JCI for three variants of the solution of the JHT of different accuracy

Source of the figure: [87].

In the case of three or more estimated parameters ($r \geq 3$), to study the properties of a multidimensional ellipsoid-shaped JCR, it is necessary to resort to a rather complex canonical analysis of the quadratic form (12).

Since the estimated parameters θ_j can be heterogeneous and have significantly different physical values, it is desirable to use relative values of the parameters $\bar{\theta} = \frac{\theta_j}{\hat{\theta}_j}$ and errors $\delta\bar{\theta}_j = \frac{\Delta\theta_j}{\hat{\theta}_j}$ for a convenient representation of the joint confidence regions (except in cases $\hat{\theta}_j \to 0$).

For the case of a higher-dimensional vector $\boldsymbol{\Theta}$ ($r>2$), using joint confidence

regions in the form (11) is associated with significant computational difficulties. Therefore, in [80, 81], joint confidence intervals (JCIs) were introduced for each *j*-th sought parameter θ_j (j = 1, 2, … r) as an indicator of the error in solutions to inverse heat conduction problems. These are projections of the JCIs onto the corresponding *j*-th coordinate axes of the *r*-dimensional space θ_j, which is equivalent to replacing the elliptical JCI (11) with a parallelepiped circumscribed around it.

First, let us consider the case of two sought parameters (r=2) considered above, for which Fig. 1 shows the relative positions of the JCRs and JCIs of the estimates. It follows from the figure that, compared to the JCIs, the JCRs are an approximate and more conservative indicator of the error in solving the IHT, with a larger area (in the case of $z \geq 2$ – volume) of dispersion of the estimates.

In this case, to construct a rectangular region of the JSI, for example, 1', described around an ellipse 1, two pairs of vertical and horizontal JSIs were used with boundary values $\pm\Delta\theta^*_1$ and $\pm\Delta\theta^*_2$, respectively, which are determined from equations (13) and (14) when the square roots in their right-hand sides are equal to zero. Then, by equating the corresponding radical expressions to zero, the following formulas can be obtained for the specified boundary values of the JSI:

$$\Delta\theta_1^* = \pm\sigma\sqrt{\frac{a_{22}B}{\det A}} \tag{15}$$

$$\Delta\theta_2^* = \pm\sigma\sqrt{\frac{a_{11}B}{\det A}}, \tag{16}$$

where $\det \mathrm{A} = a_{11}a_{22} - a_{12}^2$.

As stated above, it is advisable to present the obtained results in the form of relative boundaries of the SD, namely

$$\delta\theta_j^* = \pm\frac{\Delta\theta_j^*}{\hat{\theta}_j}, (j = 1, 2). \tag{17}$$

In [80, 81], several methods for determining the boundary values $\Delta\theta_j^*$ of the JSI for estimating the sought parameters θ_j (j=1, 2, … r) were proposed. Of these, in our experience, the following dependence obtained in [80] for the OST, using the diagonal elements a_{jj}^* of the inverse Gram matrix (10), should be recommended:

$$\Delta\theta_j^* = \pm\sigma\sqrt{a_{jj}^* \mathrm{B}} = \pm\sigma\sqrt{a_{jj}^* r \mathrm{F}_v(r, n-r)}\,, \tag{18}$$

All elements included in equation (18) have been discussed above. Note that formulas (15) and (16), based on known relationships $a_{11}/\det \mathrm{A} = a_{22}^*$ and $a_{22}/\det \mathrm{A} = a_{11}^*$ between the elements of the direct and inverse second-order Gram matrices, are reduced to form (18).

Relationship (18) is also useful for applying to the relative boundaries of the JSIs, namely:

$$\delta\theta_{\mathrm{j}}^* = \pm\frac{\Delta\theta_{\mathrm{j}}^*}{\hat{\theta}_{\mathrm{j}}} = \pm\frac{\sigma}{\hat{\theta}_j}\sqrt{a_{jj}^* \cdot r\mathrm{F}_v\left(r, n\text{-}r\right)}\,. \tag{19}$$

In the monograph [78], an approach to constructing approximate DROs for quadratic residual functions of type (3) that are nonlinear with respect to the sought parameters is proposed. It is based on the assumption that there is a minimization method (3) for obtaining estimates $\hat{\boldsymbol{\Theta}}$ close to the "true" value of Θ_0. This assumption requires confirmation by simulation modeling of solutions to control inverse problems using the specified method under real conditions.

With confirmed closeness $\hat{\boldsymbol{\Theta}}$ to Θ_0, approximate JCRs and JSIs of estimates $\hat{\boldsymbol{\Theta}}$ can be obtained by linearizing the vector of measurements $\mathrm{Y}_k(\Theta)$ included in the least squares residual function (3) near the predicted estimates. Then, in the linearized domain, the covariance matrix $\mathbf{P}$ is determined by formula (7) for values $\hat{\boldsymbol{\Theta}}$, and the approximate JCRs and JCIs are determined by formulas (12)–(19) given above for linear problems. It should be noted that the validity and effectiveness of this approach

have been confirmed in practice in [80, 81], where nonlinear IHT and IHT diagnostics were considered in many cases.

Since the JCR and JCI depend on all significant factors of the parametric identification process of heat transfer in an object, the methodology presented in [72] can be transformed for a priori (before conducting experiments and solving the HR) studies of the influence of these factors, i.e., for optimal or, at least, rational planning of parametric identification.

The main significant factors are:

• the structure (composition) of the vector of sought parameters **Θ**;

• the structure of the measurement vector **Y**: the location and number m of sensors measuring the object's temperatures;

• the number of time instants n of recording Y_k (k=1, 2, ...n);

• the noise level in the measurements, in particular, its variance σ^2 and other characteristics.

It is proposed to use the JCR and/or the SCI of a priori estimates as the target planning function, depending on the plans under consideration—sets of significant factors and their quantitative indicators.

In planning, the JCR of a priori estimates represents an r-dimensional ellipsoid (4) in space **Θ** centered at $\mathbf{\Theta}_0$, which contains the planned a priori estimates $\hat{\mathbf{\Theta}}$ of the vector **Θ** with a confidence probability of vector Θ

$$(\hat{\mathbf{\Theta}} - \mathbf{\Theta}_0)^{\mathrm{T}} \mathbf{A} (\hat{\mathbf{\Theta}} - \mathbf{\Theta}_0) = \sigma^2 \chi_\nu^2(r), \tag{20}$$

where **A** – the Gram matrix of sensitivity functions calculated using formula (2), but for values $\mathbf{\Theta}=\mathbf{\Theta}_0$ over the observation interval $n\Delta\tau$, containing n measurements $\mathbf{Y}_k$;

$\mathbf{\Theta}_0$ – the given ("true") values of the vector of sought parameters;

$\hat{\mathbf{\Theta}}$ – a priori (planned) values of the vector of sought parameters;

$\chi_\nu^2(r)$ – thequantile of the χ2-distribution for r sought parameters with confidence probability ν.

In the common case of two estimated parameters (r=2), we denote

$$\boldsymbol{\Theta}_0 - \hat{\boldsymbol{\Theta}} = \left[\Delta\hat{\theta}_1 \;\; \Delta\hat{\theta}_2\right]^{\mathrm{T}},$$

where $\Delta\hat{\theta}_1 = \hat{\theta}_1 - \theta_{1,0}$ и $\Delta\hat{\theta}_2 = \hat{\theta}_2 - \theta_{2,0}$.

Then we obtain from (20) the following two equations (similar to those obtained in [72] for a posteriori estimates), which allow us to construct an JCR ellipse in space $\Delta\hat{\theta}_1$ and $\Delta\hat{\theta}_2$ with the center at the point $\Delta\hat{\theta}_1 = \Delta\hat{\theta}_2 = 0$:

$$(\Delta\hat{\theta}_1)_{1,2} = \frac{-a_{12}\Delta\hat{\theta}_2 \pm \sqrt{a_{12}(\Delta\hat{\theta}_2)^2 - a_{11}[a_{22}(\Delta\hat{\theta}_2)^2 - \sigma^2\chi_\nu^2(r)]}}{a_{11}} \tag{21}$$

$$(\Delta\hat{\theta}_2)_{1,2} = \frac{-a_{12}\Delta\hat{\theta}_1 \pm \sqrt{a_{12}(\Delta\hat{\theta}_1)^2 - a_{22}[a_{11}(\Delta\hat{\theta}_1)^2 - \sigma^2\chi_\nu^2(r)]}}{a_{22}} \tag{22}$$

For a higher dimensionality of the vector $\boldsymbol{\Theta}$ (r>2), the use of the JSR in form (4) is associated with significant difficulties in the canonical analysis of its features. Therefore, it is proposed to use the JCI for each j-th sought parameter θ_j (j=1, 2, …r) as the objective function, as a projection of the JCR onto the corresponding j-th coordinate axis of the r-dimensional space θ_j.

In the case of r=2 considered above, to construct a rectangular region of the JCI described around the JCR ellipse from equations (6) and (7), it is necessary to find two pairs of boundary values (hereinafter referred to as boundaries) $\pm\Delta\theta^*_1$ and $\pm\Delta\theta^*_1$ и $\pm\Delta\theta^*_2$, respectively, for the vertical and horizontal SDIs using a method similar to that used in [72], in the form of the following dependencies:

$$\Delta\theta_1^* = \pm\sigma\sqrt{\frac{a_{22}B}{\det\mathbf{A}}}; \tag{23}$$

$$\Delta\theta_2^* = \pm\sigma\sqrt{\frac{a_{11}B}{\det\mathbf{A}}}, \tag{24}$$

in which a_{11} and a_{22} are the elements of the Gram matrix, and a $\det A = a_{11}a_{22} - a_{12}^2$

is its determinant.

In [80] the author proposes the following relationship for determining the boundaries of the SDI, using the diagonal elements a^*_{jj} of the inverse Gram matrix (3):

$$\Delta\theta^*_j = \pm\sigma\sqrt{a^*_{jj}\chi^2_\nu(r)} = \pm\sigma\chi_\nu(r)\sqrt{a^*_{jj}}\,. \tag{25}$$

Developing this, the authors of the study [72], taking into account the possible significant difference in the values of a^*_{jj}, propose to consider relative a priori estimates of the parameters $\theta^*_j = \theta_j / \theta_{j,0}$, and to present dependence (10) for the relative boundaries $\delta\theta^*_j$ of the *JCI* in the following form:

$$\delta\theta^*_j = \pm\frac{\Delta\theta^*_j}{\hat{\theta}_j} = \pm\frac{\sigma\chi_\nu(r)}{\hat{\theta}_j}\sqrt{a^*_{jj}}\,. \tag{26}$$

We propose the following planning mechanism: the values of the JCI boundaries obtained from dependencies (25) or (26) for the plan under consideration are compared with their permissible values. If the comparison result is negative, the plan under consideration is rejected and replaced with a new one, with new JCI boundaries defined for it. This continues until a positive comparison result is obtained.

This is critical for UAVs, as the **A** matrix allows us to determine, even at the diagnostic system design stage, whether the installed sensors are sufficient to accurately detect overheating, as well as plan for other factors of interest. For example, we use the eigenvalues of the **A** matrix and its determinants to optimize the placement of sensors onboard the autonomous vehicle.

UAV diagnostics often require high-precision determination of a single parameter, such as the thermal resistance of the traction motor winding insulation. In such cases, the use of joint confidence intervals is also recommended.

Using the above methodology allows the design engineer to estimate a specific error value in degrees or watts. This allows the onboard control system to make the

correct decision when answering, for example, whether a temperature increase in a given case is an anomaly (i.e., a failure) or an acceptable measurement error.

To monitor critical UAV components (e.g., engine combustion chamber walls or substrates of high-power electronic components), we adapt the differential-difference model (DDM) proposed in [72] for heat transfer in one-dimensional gradient heat flux density (HFD) receivers (sensors), for example, consisting of 11 blocks (Fig. 2).

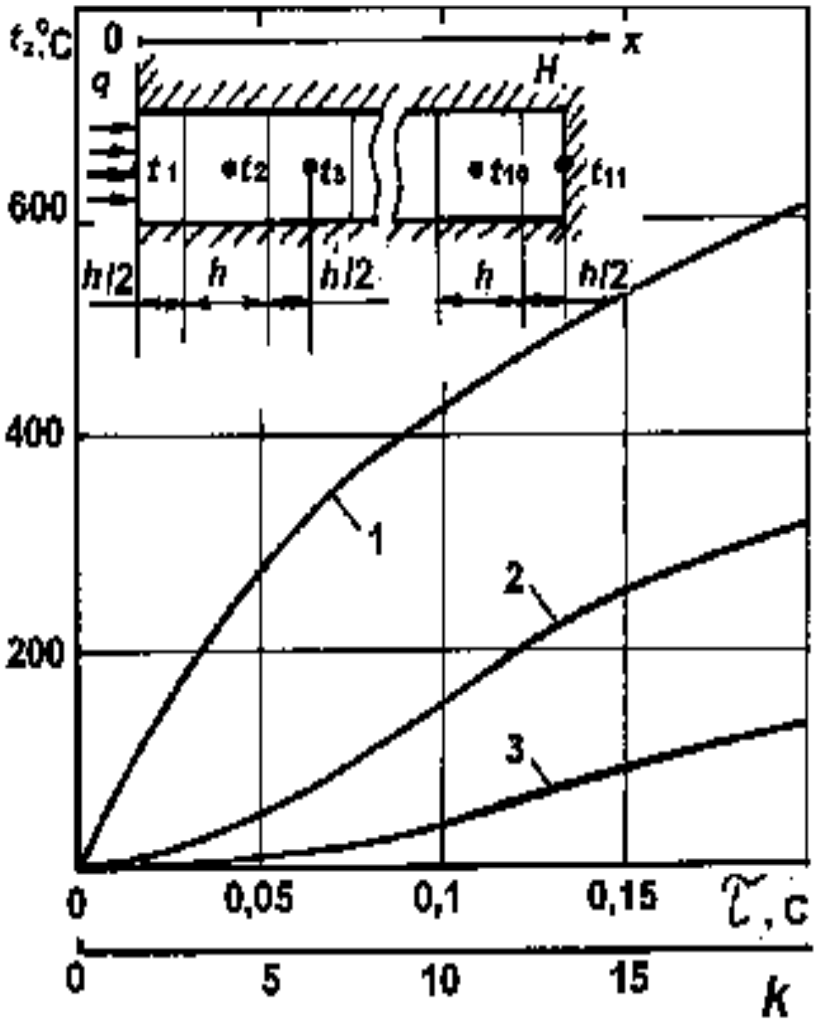


Figure 2. Thermal diagram of the differential-difference model of a one-dimensional gradient PTP and the course of the average temperatures of its blocks for the value $q=1\times10^7$ W/m^2: 1 – t_1; 2 – t_3; 3 – t_5

Source of the figure: [87].

They are typically located inside the object flush with the studied surface of its heat exchange with the environment and represent a rod of limited length H, thermally

insulated on three sides. Heat flux q enters the HFD through the working end (x=0), while the rear end (x=H) is assumed to be thermally insulated. The temperature course at one or more points along the HFD length or their difference is measured and serves as the basis for solving the boundary JHT problems – determining (restoring) the density values of the heat fluxes entering the HFD. The cases of either variable $q=q(\tau)$ or constant q=const are considered, but in the initial section of the transient process, the dynamic method of measuring constant heat flows is used.

The setting of the IHT is preceded by its parameterization is preceded by its parameterization—the identification of an ($r\times1$)-vector of the desired parameters Θ=const in the heat transfer model of the heat transfer station. We will consider two types of boundary temperature control, namely:

1) In the case of q=const (IHT-1 plan), parameterization is carried out naturally in the form $\Theta_1=[q]$=const;

2) in the more general case of $q=q(\tau)$ (IHT-2 plan), we will use a piecewise linear approximation of the function $q(\tau)$ for parameterization in successive approximation sections of equal duration $\Delta=n\Delta\tau$, each containing n discrete (with a step of $\Delta\tau$) values of the measurement vector $\vec{\mathbf{Y}}_k$ (k=1, 2, …n). In each section, the current values of the heat flow q_k (k=1, 2, …n), changing according to a linear law, can be represented by the following function of the left $q_л$ and right $q_{пр}$ of its values at the boundaries of the section Δ:

$$q_k = q_л + k\frac{q_{пр} - q_л}{n}. \qquad (27)$$

Thus, in this case, the (2×1)-vector of the sought parameters in the first and all subsequent sections has the form $\mathbf{\Theta}_2 = [q_л \quad q_{пр}]^T = \text{const}$;

3) in addition to the two boundary coefficients considered, the problem of determining or refining the thermophysical characteristics (TPC) of the material, in particular the thermal conductivity λ, may arise. This leads to the advisability of setting up two more parameterized combined (boundary-coefficient) IHTs with vectors of the

sought parameters $\mathbf{\Theta}_3 = [q \quad \lambda]^T$ (plan IHT-3) и $\mathbf{\Theta}_4 = [q_л \quad q_{пр} \quad \lambda]^T$ (plan IHT-4).

Thus, four experimental plans for the IHT, represented by the above vectors of the sought parameters Θ_1–Θ_4, will be subject to planning.

The following initial data were adopted for planning:

1) characteristics of the HTPS: length $H=4\cdot10^{-3}$ m, material – heat-resistant alloy with the following characteristics: density $\rho=7.8\cdot10^3$ kg/m3; thermal conductivity $\lambda=20$ W/m•deg; specific heat capacity C=0.43 kJ/kGK; thermal diffusivity $a=5.9\cdot10^{-6}$ m2/s. Initial condition $T_0=273$ K;

2) the DDM described above, consisting of $l=11$ blocks, is used as a model of the temperature state of the HTPS. In Fig. 2 shows the temperature history t_1, t_3 and t_5 of blocks 1, 3 and 5 obtained using the DDM for the case $q_0=q_{л0}=1\cdot10^7$ W/m2. The time discretization step here and below is 0,01 s. Thus, the heating of the UAV engine components was simulated at a peak load of $q=1\ 10^7$ W/m^2;

3) the given ("true") values of the estimated parameters included in the vectors $\mathbf{\Theta}_1$–$\mathbf{\Theta}_4$ are: $q_0=q_{л0}=1\cdot10^7$ W/m2; $q_{пр0}=1{,}8\cdot10^7$ W/m2 and $\lambda_0=20$ W/m·deg.

The following factors are considered in planning:

a) the structure (composition) of the measurement vector $\mathbf{Y}_k$, in particular, plans with single temperature measurements t_1, t_3 or t_5, as well as plans with simultaneous measurements of two indoor unit temperatures t_3 and t_5 (for the IHT-3 plan);

b) the variance σ^2 of the noise in the measurements, which characterizes the recording characteristics of the measured temperatures;

c) the number n (5, 10, or 20) of values of the measurement vector $\mathbf{Y}_k$ (k=1, 2, … n) used to obtain IHT solutions for each section Δ, since for the plan IHT -1 and IHT -2 plans, the choice of n is fundamentally important, determining the response time of the dynamic heat flux measurement method q= const. For the IHT-3 and IHT-4 plans, the choice of n is also very important, determining the optimal value $\Delta=n\Delta\tau$ for the sections of the piecewise linear approximation of the function $q(\tau)$: on the one hand, increasing n facilitates a successful solution of the IHT, but on the other hand, it leads to a deterioration in the quality of the approximation of the function $q(\tau)$;

d) it is also advisable to include the composition of the vector $\hat{\boldsymbol{\Theta}}$ of the sought parameters among the planning factors.

A priori relative boundaries of the JCI (26) are used as an indicator of the accuracy of estimates for the plans under consideration. Their values are compared with the permissible spreads of the sought parameters.

Planning for IHT-3 and IHT-4 is performed for the first section Δ, assuming the possibility of transferring its results to subsequent sections.

The following planning stages were completed.

Using numerical differentiation and the DDM of HFD, the sensitivity functions u_{zjk} of the z-th measured temperatures (z=1, 3 or 5) to the j-th sought-after parameters (j=1, 2, … r) at the k-th instants of time (k=1, 2, …, n) were calculated for the "true" values of the sought-after parameters given in the section above. For each u_{zjk}, the increment values $\Delta\theta_j$ were pre-selected, which, as a rule, did not exceed $\pm 0{,}05\theta_{j0}$.

Figure 3 shows the sensitivity functions of temperatures t_1, t_3 and t_5 to changes in the parameters of the sought-after parameters $q_л$ at the k-th instants of time.

Figure 4 shows the sensitivity functions of temperatures t_1, t_3 and t_5 to changes in the desired parameter q_r at the k-th time instant.

For plans IHT-3 and IHT-4, the current values of heat flow q_k, determined by formula (27), were substituted into the DDM. Figure 4 shows the sensitivity functions of temperatures t_1, t_3 and t_5 to changes in the desired parameters q, qл, qпр, and λ at the k-th time instant.

Figure 5 shows the sensitivity functions of temperatures t_1, t_3 and t_5 to changes in the desired parameters q, q_l, q_r and λ at the k-th time instant.

For each IHT plan, using the obtained sensitivity function values for n=5, 10, and 20, using formula (9), nine direct Gram matrices A were calculated, followed by nine inverse matrices $\mathbf{A}^{-1}$ for each of the nine variants (plans) – three variants of n for the four parameters q_0, q_l, q_r and λ_0.

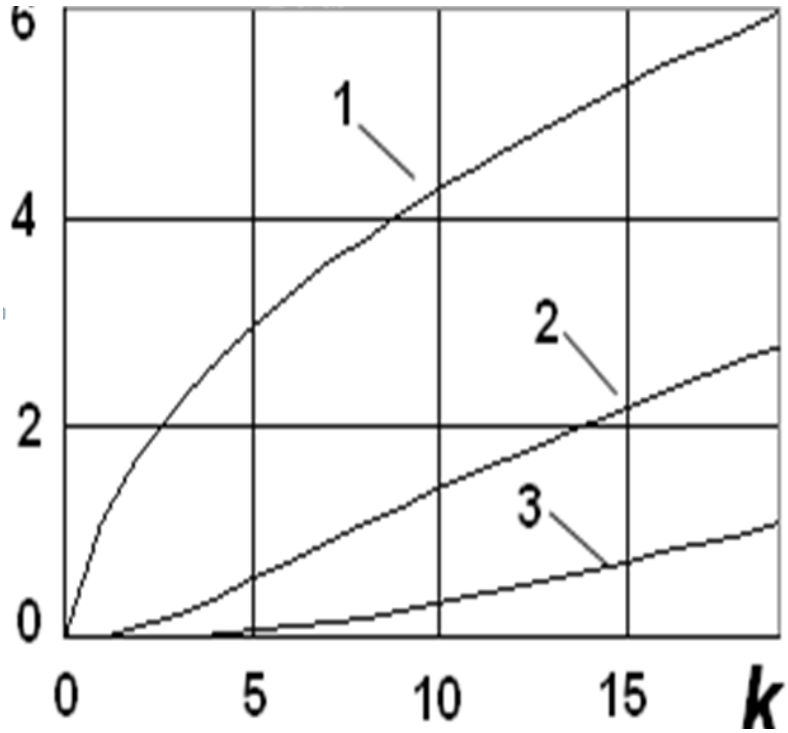


Figure 3. Sensitivity functions of the measured temperatures to changes in the desired parameter q_l: 1 – t_1; 2 – t_3; 3 – t_5

Source of the figure: [87].

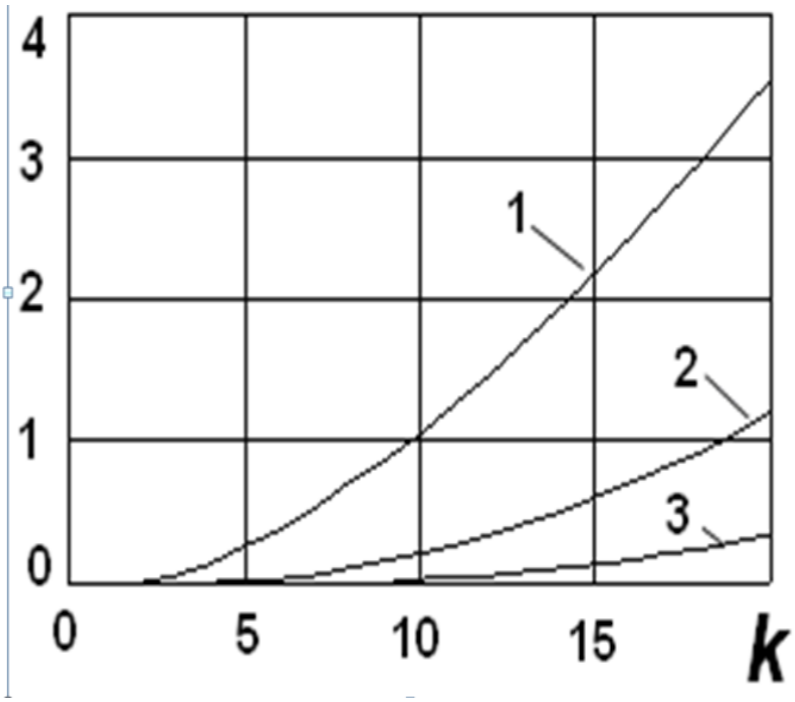


Figure 4. Sensitivity functions of measured temperatures to the desired parameter q_r; 1 – t_1; 2 – t_3; 3 – t_5

Source of the figure: [87].

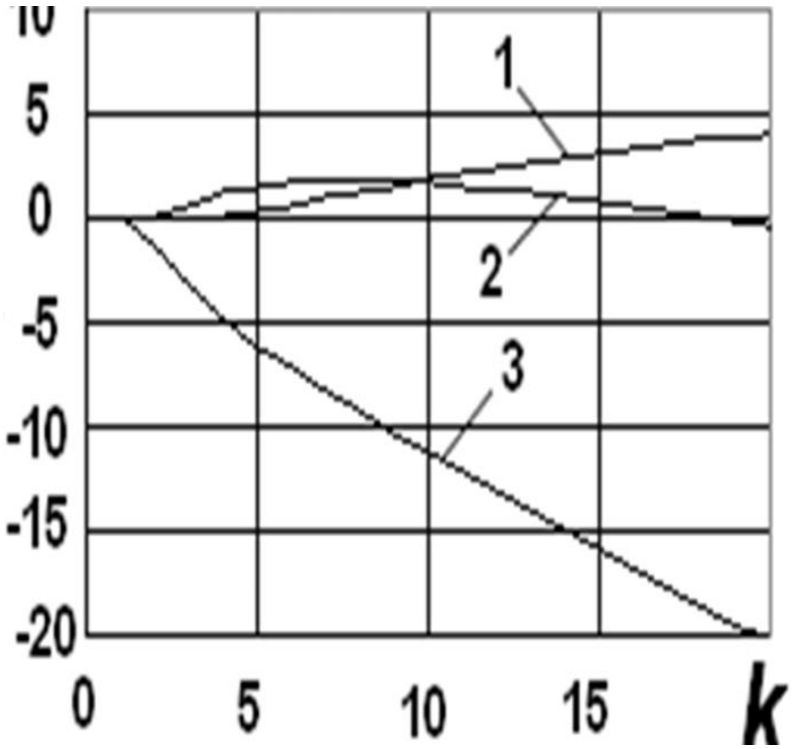


Figure 5. Sensitivity functions of measured temperatures to the desired parameter λ; 1 – t_1; 2 – t_3; 3 – t_5

Source of the figure: [87].

Furthermore, for the IHT-3 plan, two additional variants of the matrices **A** and A^{-1} were constructed for the plan with simultaneous temperature measurements t_3 and t_5 for $n = 10$ and $n = 20$.

The relative standard deviations $\delta\theta_j^*$ of the estimates were determined using formulas (24, 25, 27) for a variance of $\sigma^2=1$. Their values (in %) for each IHT are summarized in the table. If necessary, they can be recalculated for any other values of σ^2.

Thus, in this case, the key indicator of UAV reliability is the proposed metric – a priori relative boundaries – Joint Confidence Intervals. In the context of autonomous systems, this works as follows:

• If the current confidence interval of a parameter falls outside the acceptable limits, the UAV control system classifies this as a failure or structural damage to the component;

• The number of measurements (5, 10, or 20) is selected as a compromise between

response time, which is important for UAV control, and the quality of heat flow function identification.

During numerical modeling and planning of identification parameters for critical UAV components (based on the DDM of FHD), the following results were obtained, presented in Table 1.

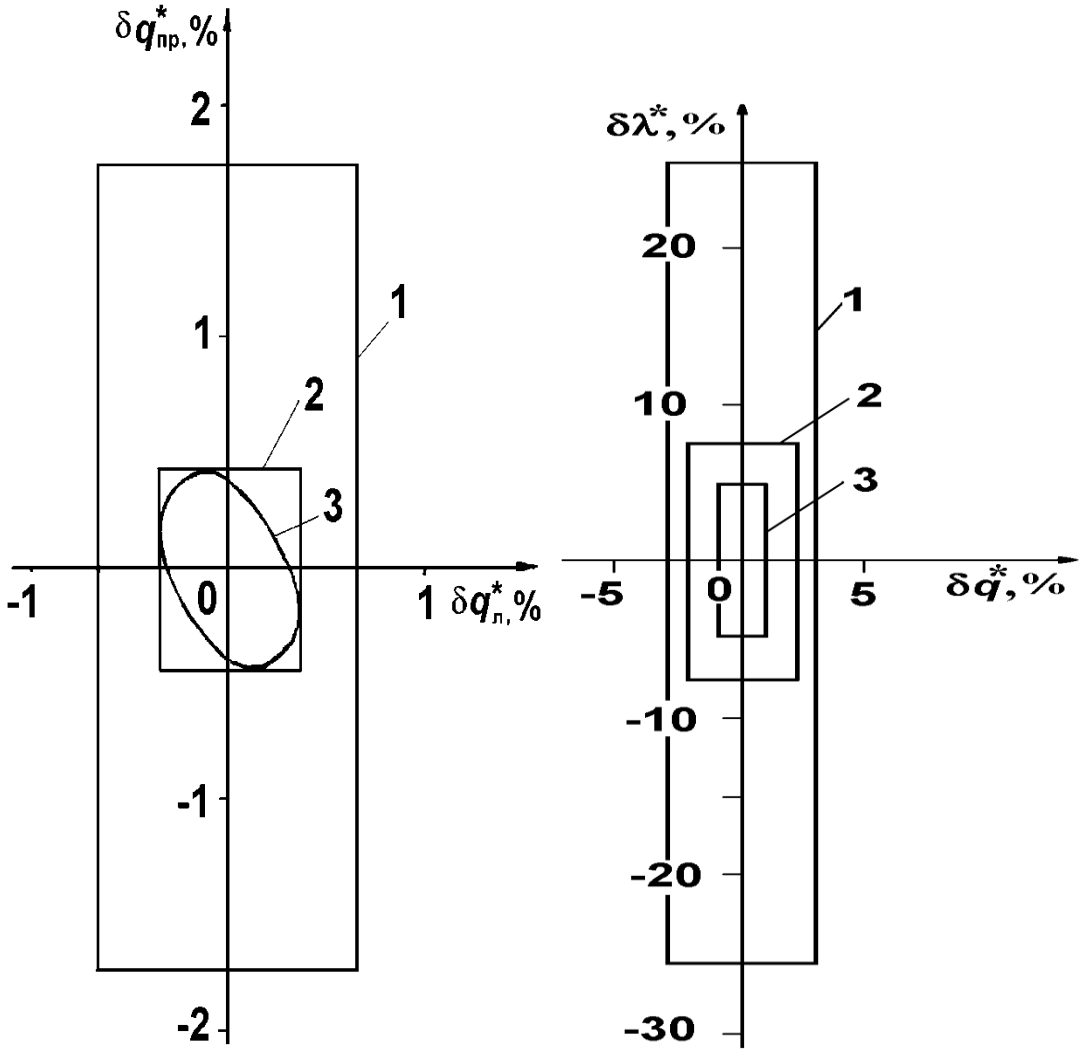
Table 1.

Summary results of parametric identification planning for solutions of the considered IHT (with $\sigma^2=1$)

Options IHT	Vector of desired parameters	№№ measurement blocks temperatures	Relative boundaries СДИ $\delta\theta_j^*$, %								
			$\pm\delta q^* = \pm\delta q_л^*$			$\pm\delta q_{пр}^*$			$\pm\delta\lambda^*$		
			Number of measurements n			Number of measurements n			Number of measurements n		
			5	10	20	5	10	20	5	10	20
Plan IHT–1	$\Theta=[q]$	1	0,39	0,20	0,10	–	–	–	–	–	–
		3	2,67	0,75	0,25	–	–	–	–	–	–
		5	60,20	3,77	0,71	–	–	–	–	–	–
Plan IHT–2	$\Theta=\begin{bmatrix} q_л \\ q_{пр} \end{bmatrix}$	1	1,16	0,67	0,36	6,66	1,74	0,44	–	–	–
		3	3,50	3,53	1,12	34,0	13,2	1,78	–	–	–
		5	192,0	30,0	4,40	3130	157	9,12	–	–	–
Plan IHT–3	$\Theta=\begin{bmatrix} q \\ \lambda \end{bmatrix}$	1	4,60	0,37	0,32	–	–	–	11,1	8,27	6,90
		3	15,80	2,00	0,32	–	–	–	25,2	7,00	2,80
		5	60,0	3,10	0,35	–	–	–	287	25,0	5,25
		3 и 5	–	0,83	0,16	–	–	–	–	5,10	2,30
Plan IHT–4	$\Theta=\begin{bmatrix} q_л \\ q_{пр} \\ \lambda \end{bmatrix}$	1	7,54	0,78	0,32	16,8	2,00	0,68	22,3	0,44	0,38
		3	192,7	32,6	3,80	883	32,6	5,50	158	59,9	8,40
		5	1150	530	200,2	5245	1114	187	426	236	133

In Fig. 6, for some experimental designs, the JCRs formed by two pairs of values of their relative boundaries are shown, and for one of the designs, a relative elliptical

JCR is shown.



а) б)

Figure 6. Examples of relative JCR and JCI: a) IHT–2: 1 – JCI for plan 2 (t_3 is measured at n=20); 2 – JCI for plan 1 – (t_1 is measured at n=10); 3 – JCI for plan 2; b) IHT–3: JCI for the following plans: 1 – t_5 is measured, n=10; 2 – t_3 is measured, n=10; 3 – t_3 and t_5 are measured, n=10

Source of the figure: [87].

Joint confidence interval (JCI) analysis revealed that the location of temperature sensors is critical to ensuring high performance of the UAV monitoring system. The optimal location and use of temperature sensor readings is in the surface layer. This ensures a minimum error of 0.39% even with a small number of measurements (n=5).

There are also limitations to sensor placement. Deepening the measurement point, i.e., moving to the third block, and especially to the fifth, leads to a sharp increase in uncertainty. Under dynamic UAV flight conditions, using sensors in these blocks is deemed impractical due to the actual instability of the JCI solution (errors of 100–1000% according to Table 1).

For implementation in the onboard control system, a sufficient number of measurements (n=10) is recommended for JCI-1 and JCI-3 plans (determination of constant heat loads), which allows for heat flux identification accuracy of 0.20–0.37%.

For integrated IHT-4 plans (simultaneous determination of heat fluxes and thermal conductivity), it is necessary to increase the sample size to n=20 when using a sensor in the first block, or to use a combination of sensors in the third and fifth blocks to stabilize the solution.

It was found that adding an additional measurement channel (a combination of sensors in the third and fifth blocks) for IHT-3 plan has a positive effect on the accuracy of a priori estimates. This confirms the hypothesis that for reliable autonomous systems, it is advisable to use sensor redundancy with subsequent processing of the results to compensate for the low information content of individual measurement points.

Thus, based on the results of the conducted research, a practical conclusion or recommendation can be drawn: placing sensors closer to the surface and using n=10 measurements to improve response time, or n=20 measurements to improve accuracy. However, not everything is so clear-cut. The closer the temperature sensor is to the surface, for example, of the combustion chamber, the higher the probability of its failure. Certain combinations of sensor configurations and other parameter values lead to solution instability, which is a key factor in reliability and safety for UAVs and other similar systems.

Thus, the parametric identification method in technical objects consists of obtaining, based on experimental data, optimal estimates $\hat{\Theta}$ of a certain vector Θ of sought-after heat transfer parameters included in the mathematical model of the latter [72]. However, the accuracy of such identification directly depends on the

methodology of experimental design and the assessment of heat transfer identification errors. Next, we will pose a problem very closely related to the problem of parametric identification accuracy discussed above and propose ways to solve it.

The issue is as follows. In modern research of various physical processes and phenomena, most measurement methods are indirect, i.e., the sought-after physical quantities are determined (reconstructed) from other physical quantities directly measured during the experiment using various techniques.

For example, measuring the temperature of a gas flow in an internal combustion engine with a thermocouple (thermocouple) involves recording the electrical voltage at the output of this measuring device, then calculating it based on certain temperature dependencies of the thermocouple's sensitive element, and then reconstructing the sought-after gas temperature, taking into account various corrections. These dependencies constitute the measurement equation of the measuring device.

It is well known that thermophysical experiments are one of the most complex areas of measurement technology. This is due to the difficulty (often impossible) of accounting for all factors involved in heat exchange between the measuring device and the medium being studied (especially gaseous media). It must be taken into account that measurement errors at high temperatures (up to 2000°C) due to all factors involved in such heat exchange can reach 200-300°C.

The measurement equations in thermophysical experiments are almost always nonlinear and often unstable, yielding negative results even with relatively small errors in direct measurements.

In modern heat engines, the following challenges are added to the above: extremely high operating temperatures, non-stationarity of the processes being studied, significant temperature gradients, and high demands on flow conditions and structural strength.

Indirect measurement problems, including those of thermophysical experiments, are often considered inverse problems. Interest in indirect measurement is constantly growing, driven both by practical needs and the rapid development of computer technology methods and tools. In particular, in thermal power engineering, including

heat engines, indirect measurement methods are widely used both in experimental studies of the temperature state of working fluids and structural elements and in the optimal design of the latter.

Considering all of the above, we propose reducing the process of indirect measurements to solving an inverse problem by applying the method of parametric identification of the measuring device. This method consists of obtaining, based on experimental data, optimal estimates $\hat{\mathbf{\Theta}}$ of a certain vector $\mathbf{\Theta}$ of the sought-after parameters of the interaction of the measuring device with the environment, included in its mathematical model. This approach is used in the monograph [79], which demonstrates that the most effective approach to solving inverse problems is the extremal approach, which consists of minimizing quadratic functionals.

However, in our opinion, this methodology is not widely used in the development and implementation of modern engineering measurement methods.

The objectives of this study are:

1) to present a consistent and accessible methodology for representing the indirect measurement process as a parametric identification of a measuring device;

2) to illustrate the specifics of applying this methodology, to perform a parametric identification of a device for measuring high temperatures of gas flows;

3) to implement this approach using a digital Kalman filter, which is dictated by the similarity of its mathematical apparatus with the Gram matrix analysis discussed above.

Most known parameter identification methods have been developed for linear, time-invariant systems. For nonlinear systems, the theory behind these methods is insufficiently developed to allow their selection based on established recommendations for a given system. Below is an example of identifying the parameters of a measuring device using a nonlinear mathematical model.

We will assume that there is a mathematical model of the measuring device (hereinafter, the model) that allows for calculating the values of the measured parameters for instants of time $\tau_k = k \cdot \Delta\tau$ (k=1, 2, … n):

$$\mathbf{Y}_k = f(\mathbf{\Theta}_k). \quad (27)$$

where $\mathbf{Y}_k$ – the vector of measurements;

$\mathbf{\Theta}_k$ - the vector of sought parameters at the *k*-th measurement step;

k – the measurement step number.

Information about the object at the *k*-th step is provided by the measurement vector

$$\mathbf{Y}_k = \mathbf{\Psi}_k + \boldsymbol{\varepsilon}_k,$$

where $\mathbf{\Psi}_k$ – the actual values of measured physical quantities;

$\boldsymbol{\varepsilon}_k$ – the measurement noise.

It is known that measurement noise is one of the factors capable of causing instability in the solution of the inverse problem, i.e., making successful measurements impossible. To account for the influence of noise at the stages of setting up and solving the inverse problem, we will use the assumption, generally accepted for most measurements, that the components ε_{ik} of the vector $\boldsymbol{\varepsilon}_k$ are normally distributed random variables with zero mathematical expectations, the same variance σ^2, and are uncorrelated with each other. This assumption seems entirely justified for homogeneous temperature measurements performed using the same primary and recording transducers. This allows us to represent the main characteristic of the random vector $\boldsymbol{\varepsilon}_k$ –its covariance ($m \times m$)-matrix **R**—in the form

$$\mathbf{R} = \sigma^2 \mathbf{I},$$

where **I** – the identity ($m \times m$)-matrix;

R – the measurement noise covariance matrix.

In the measuring device model, it is necessary to identify the ($r \times 1$)-vector $\mathbf{\Theta} = [\Theta_j]_{j=1}^{r}$ of the desired parameters. The procedure for identifying Θ in the model describing the measuring device in relation to the environment is called parameterization of the measuring device.

Based on the measuring device model, a prediction of the measurement vector $\mathbf{Y}_k$ is calculated depending on the vector of desired parameters $\mathbf{\Theta}$ at $\boldsymbol{\varepsilon}$=0:

$$\hat{\mathbf{Y}}_k(\mathbf{\Theta}) = \left[y_{ik}(\mathbf{\Theta})\right]_{i=1}^{m}$$

In this case, parametric identification consists of determining the optimal estimates $\hat{\mathbf{\Theta}}$ of the vector Θ based on n values of the measurement vector $\mathbf{Y}_k$ in model (27). For these purposes, the most common method is minimization with respect to $\mathbf{\Theta}$ of the following quadratic residual function [78]:

$$\mathbf{\Phi}(\mathbf{\Theta}) = \sum_{k=1}^{n}\left[\mathbf{Y}_k - \hat{\mathbf{Y}}_k(\mathbf{\Theta})\right]^{\mathrm{T}}\mathbf{R}^{-1}\left[\mathbf{Y}_k - \hat{\mathbf{Y}}_k(\mathbf{\Theta})\right]. \quad (28)$$

Thus, parametric identification of a measuring device can be reduced to the generalized least squares least squares method [78], which has been well studied in the mathematical and technical literature. Its advantage is that it does not require a priori knowledge of the statistical properties of the estimates $\hat{\mathbf{\Theta}}$, yields unbiased estimates $\hat{\mathbf{\Theta}}$, and ensures minimal variance of the estimates if the model is linear and the measurement noise is normally distributed.

For minimization $\mathbf{\Phi}(\mathbf{\Theta})$, it is proposed to use the following modified Kalman filter algorithm for the desired parameters $\mathbf{\Theta}$ [79]. In this algorithm, optimal estimates of the vector of the desired parameters $\hat{\mathbf{\Theta}}_k$ at the k-th measurement step are determined as follows:

$$\hat{\mathbf{\Theta}}_k = \hat{\mathbf{\Theta}}_{k-1} + \mathbf{K}\left[\mathbf{Y}_k - \hat{\mathbf{Y}}_k(\hat{\mathbf{\Theta}}_{k-1})\right], \quad (29)$$

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}_k^T\left[\mathbf{H}_k\mathbf{P}_{k-1}\mathbf{H}_k^T + \mathbf{R}\right]; \quad (30)$$

$$\mathbf{P}_k = \mathbf{P}_{k-1} - \mathbf{K}\mathbf{H}_k\mathbf{P}_{k-1}; \quad (31)$$

$$\mathbf{H}_k = \frac{\partial}{\partial\mathbf{\Theta}}\hat{\mathbf{Y}}_k(\hat{\mathbf{\Theta}}_{k-1}). \quad (32)$$

In the above algorithm, $\mathbf{P}_k$ and $\mathbf{P}_{k-1}$ are the covariance matrices of the estimation errors, the diagonal elements of which are the variances of the estimation errors of the desired parameters, and $\mathbf{H}_k$ is the matrix of the sensitivity functions of the measurement prediction vector to the desired parameters at the *k*-th step.

It is easy to see that we encountered similar calculations when solving the problem of estimating the accuracy of parametric identification procedures and planning experiments for UAVs. Only now, with minor modifications, we propose using this mathematical apparatus (a modified Kalman filter (29)-(32)) to conduct the necessary measurements.

A distinctive feature of the Kalman filter is that it does not require continuous storage of measurement data. Calculations are performed for each successive measurement step. Furthermore, for each measurement step, using the methodology for assessing the accuracy of parametric identification proposed at the beginning of this study, it is possible to simultaneously obtain estimates of the accuracy of the indirect measurements being performed.

We implemented the proposed approach when measuring high (up to 2200°C) gas flow temperatures using an adaptive pressure-reducing flow-through temperature transducer [88].

A temperature transducer, manufactured as a flow-through probe, is introduced into the gaseous medium (fuel combustion products in a jet engine). The temperature of the gas flow $T_г$ being measured at the inlet of the device's cooled channel is calculated (reconstructed) based on the device's mathematical model, based on directly measured gas temperatures T_1 and T_2 in the channel and at two points along its axis.

The measurement equation for this method is as follows:

$$\vartheta_г = \vartheta_1\left(\vartheta_1/\vartheta_2\right)^A, \tag{33}$$

where $\vartheta_г$ – the temperature of the gas flow being studied at the inlet to the temperature converter;

A – the method conversion coefficient;

ϑ_1 и ϑ_2 – the gas temperature in two sections of the cooled channel of the device.

By changing the gas flow rate in the device channel, it is possible to determine A:

$$A = \left[ln\left(\vartheta_{22}/\vartheta_{21}\right)/ln\left(\vartheta_{12}/\vartheta_{11}\right) - 1 \right]^{-1}, \quad (34)$$

where ϑ_{11}, ϑ_{21}, ϑ_{12} и ϑ_{22} – the gas temperatures in two channel sections for two values of gas flow velocity W_1 and W_2.

A distinctive feature of the method is that, for small changes in gas flow velocity in the device channel (a necessary condition for constant gas flow characteristics in the channel), temperatures ϑ_{11} and ϑ_{21} differ from ϑ_{12} and ϑ_{22} by amounts comparable to the measurement noise. Since the logarithms of the ratios of close quantities tend to zero, formula (34) exhibits instability due to zero-by-zero division. Furthermore, processing the measurement results using conventional statistical methods does not yield positive results.

To solve this problem when diagnosing high-temperature gas flows, we proposed applying the parametric identification algorithm (29)-(32) to the Adaptive Pressure-Reducing Flow-through Temperature transducer (ARPT) method, using the transformed measurement equations (33) for two values of gas velocity in the device channel as a model:

$$\begin{cases} ln\,\vartheta_{11} = ln\,\vartheta_{\Gamma} - A ln\left(\vartheta_{11}/\vartheta_{21}\right); \\ ln\,\vartheta_{12} = ln\,\vartheta_{\Gamma} - A ln\left(\vartheta_{12}/\vartheta_{22}\right). \end{cases} \quad (35)$$

The vector of the required parameters will have the following form:

$$\mathbf{\Theta} = \begin{bmatrix} ln\,\vartheta_{\Gamma} \\ A \end{bmatrix}.$$

We write the vector of measurements for the k-th step as

$$\mathbf{Y}_k = \begin{bmatrix} \vartheta_{11k} + \varepsilon_{11k} \\ \vartheta_{12k} + \varepsilon_{12k} \end{bmatrix}.$$

Then the matrix of sensitivity functions of the measurement vector to the sought parameters will take the form:

$$\mathbf{H}_k = \begin{bmatrix} 1 & -ln(\vartheta_{11k}/\vartheta_{21k}) \\ 1 & -ln(\vartheta_{12k}/\vartheta_{22k}) \end{bmatrix}.$$

The convergence of the optimal estimation algorithm as applied to model (35) and vectors of parameters and measurements was investigated by us in wide ranges of possible values of the measured gas temperature $\vartheta_г$ and the conversion coefficient A, initial estimates $\vartheta_{г_0}$ and A_0, the length of the measurement sample N, etc. Figure 6 shows samples of temperatures ϑ_{11}, ϑ_{21}, ϑ_{12} and ϑ_{22} in two sections of the temperature converter channel for two values of the gas flow velocity W_1 and W_2 and the movement of the determined estimate $\vartheta_г$.

To conduct a numerical experiment with various initial (reference) values of $\vartheta_г$, Coefficient A and variations in the gas flow velocity W in the channel, the gas temperatures ϑ_{11}, ϑ_{12}, ϑ_{21} and ϑ_{22} in the temperature transducer channel were found using formulas (34) and (35). The reference temperatures ϑ_{11}, ϑ_{21}, ϑ_{12}, ϑ_{22} and $\vartheta_г$ for Fig. 6 are equal to 1500, 1142, 1300, 900 and 1823 degrees, respectively. The number of measurements in this case was 40. By summing the reference values with the values ε_{11k}, ε_{12k}, ε_{21k} and generated by random number generators with different variances σ, the temperature samples were calculated, which served as the initial data for the numerical experiment.

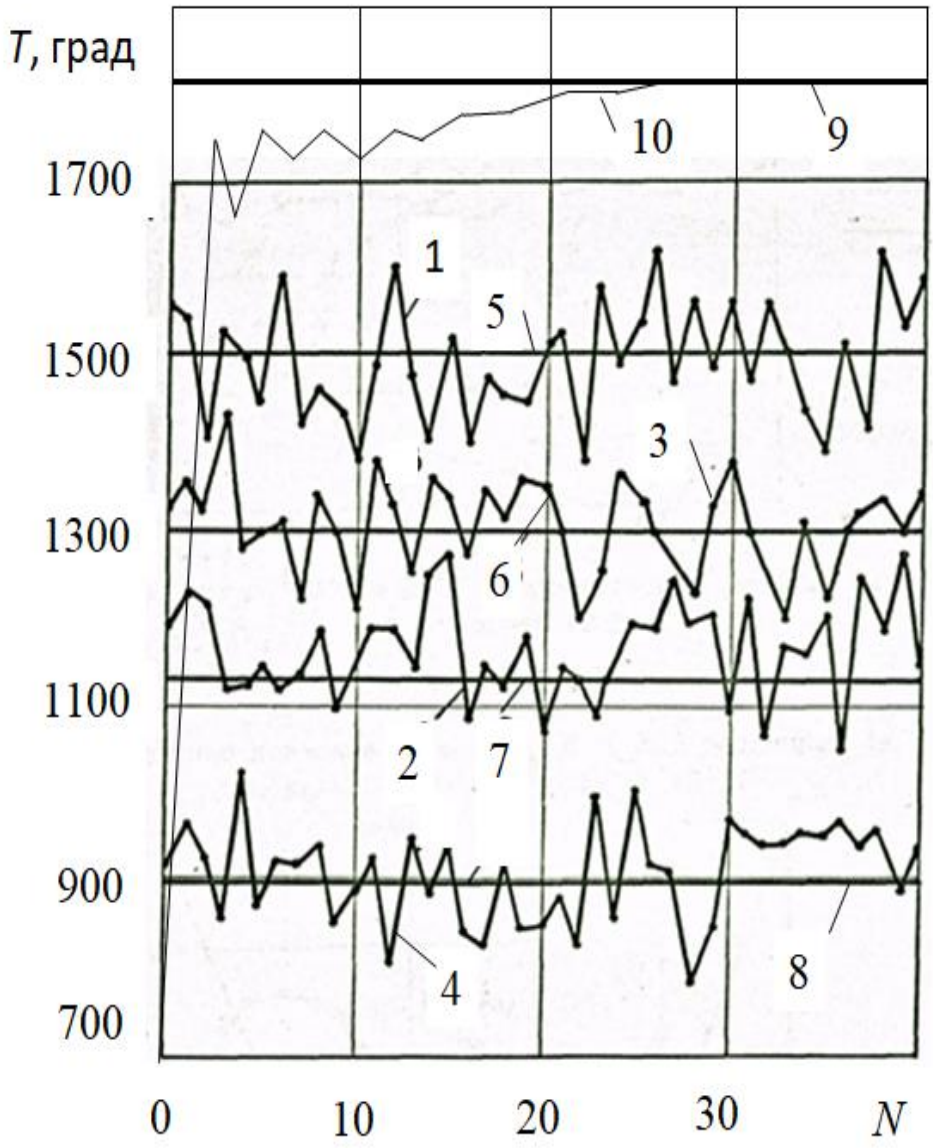


Figure 7. Samples of gas temperatures in the cross-sections of the temperature transducer channel for the numerical experiment: 1, 3 – gas temperatures in the device channel for sections 1 and 2 at the gas velocity in the channel; 2, 4 – the same at the gas velocity in the channel; 5, 6, 7, 8 and 9 – reference temperature values ϑ_{11}, ϑ_{21}, ϑ_{12}, ϑ_{22} and $\vartheta_{г}$; 10 – assessment movement $\vartheta_{г}$

Source of the figure: [88].

In the overwhelming majority of cases, stable convergence of the parametric identification procedure was observed for N≥35 with an accuracy of 0.2%. Thus, the parametric identification algorithm for a measuring instrument proposed for conducting indirect measurements allows for a significant reduction in the influence of

the stochasticity of the initial measurement information, as well as its volume. It can be argued that, for example, for this device, the proposed methodology is practically the only option for conducting measurements.

The results of numerical modeling showed that the proposed methodology for representing the measurement process as a parametric identification of the measuring device, using the example of an adaptive pressure-reducing flow-through temperature sensor, allows for highly accurate measurements with a minimal amount of measurement information. In some cases, this representation of the measurement process is practically the only option for conducting measurements.

To summarize the results of this study:

1) The proposed parametric identification method enables efficient identification of UAV component thermal parameters;

2) The use of joint confidence intervals as the primary accuracy metric provides a robust basis for assessing the reliability of the obtained solutions, allowing for distinguishing sensor noise from potential structural failures;

3) It has been established that for high-speed diagnostic systems, a measurement window of n=10 to n=20 provides a sufficient balance between computational efficiency and estimation accuracy;

4) Future research will focus on the use of the Kalman filter in the proposed methodology for parametric identification of UAV components, as well as for transmitting various information (measurement and control) under various interference conditions. This is precisely what the Kalman filter was originally designed for, some 60 years ago.

• Since this area entails extensive and long-term work with the Kalman filter, we consider it necessary to provide a brief historical background on this scientific instrument. This information is available from open sources [89]. Some believe that the filter's capabilities have not been fully explored, although it has been known and used for over half a century. The filter is named after the Hungarian mathematician Rudolf E. Kalman, who emigrated to the United States, although researchers Thiele and Swerling developed a similar algorithm earlier (Thiele considered only a particular

formulation, while Swerling's algorithm is virtually identical to Kalman's). Bucy of the University of Southern California contributed to the theory that led to the so-called Kalman-Bucy filter. Schmidt is credited with first implementing the Kalman filter during Kalman's visit to the Ames Research Center, leading Kalman to see the applicability of his ideas to the Apollo trajectory estimation problem, ultimately leading to its inclusion in the Apollo computer navigation system. The Kalman filter was first described and partially developed by Swerling (1958), Kalman (1960), and Kalman and Bucy (1961).

- Kalman filters have proven critical in the implementation of navigation systems for US Navy nuclear-armed ballistic missile submarines and in the navigation systems of cruise missiles such as the Tomahawk. It was also used in the navigation and control systems of the NASA Space Shuttle project and is used in the control and navigation systems of the International Space Station.

- The Kalman filter operates with the concept of a system's state vector (a set of parameters describing the state of the system at a given point in time) and its statistical description. In general, the dynamics of a state vector are described by the probability densities of the distribution of its components at each moment in time. Given a specific mathematical model of the observations made of the system, as well as a model of a priori changes in the state vector parameters, one can write an equation for the posterior probability density of the state vector at any moment in time. This differential equation is called the Stratonovich equation. The Stratonovich equation cannot be solved in general. An analytical solution can only be obtained under a number of constraints or assumptions:

- • Gaussian prior and posterior probability densities of the state vector at any moment in time (including the initial moment);

- • Gaussian shaping noise;

- • Gaussian observation noise;

- • white observation noise;

- • linearity of the shaping process model (which, we recall, must be a Markov process).

The classical Kalman filter is a set of equations for calculating the first and second moments of the posterior probability density function (in the sense of the vector of mathematical expectations and the matrix of variances, including cross-variances) under given constraints. Since for a normal probability density function, the mathematical expectation and variance matrix completely define the probability density function, the Kalman filter can be said to calculate the posterior probability density function of the state vector at each instant in time, thereby completely describing the state vector as a random vector variable.

The calculated values of the mathematical expectations are optimal estimates based on the mean square error criterion, which explains its widespread use.

There are several variations of the Kalman filter, each differing in some approximations that must be applied to reduce the filter to the form described and its dimensionality:

• Extended Kalman filter – convergence of nonlinear observation models and the underlying process using linearization via a Taylor series expansion;

• Sigma-point Kalman filter - used in problems where simple linearization leads to the destruction of useful relationships between the components of the state vector. In this case, "linearization" is based on the sigma-point transform;

• Ensemble Kalman filter (EnKF). Used to reduce the dimensionality of the problem;

• Variants with a nonlinear additional filter, allowing non-Gaussian observations to be normalized;

• Variants with a "whitening" filter, allowing for working with "colored" noise.

Furthermore, there are Kalman filter analogs that fully or partially utilize the continuous-time model:

• Kalman-Bucy filter, in which both the system evolution and measurements are functions of continuous time;

• Hybrid Kalman filter, which uses continuous time to describe the system evolution and discrete time instants for measurements.

The author of the study proposes using an extended Kalman filter for the desired parameters. Such experience has already been achieved, and it can be considered successful. Although there are some subtleties in setting up such a filter.

DOI: 10.46299/ISG.2026.MONO.TECH.2.4.3

4.3 Декларативні методи побудови адаптивних користувацьких інтерфейсів

На сучасному етапі розвитку технологій проектування користувацьких інтерфейсів (UI) та систем взаємодії людина-машина (HMI) трансформувалося з допоміжної задачі у складну інженерну дисципліну.

Сучасний UI більше не будується як статичний набір елементів. Застосування компонентного підходу дозволяє створювати інтерфейси як складні системи зі зворотним зв'язком. Використання сучасних нативних тегів, таких як <dialog>, <popover> чи <details>, дозволяє реалізувати складну логіку поведінки без надлишкового JavaScript, що підвищує продуктивність та доступність.

У промисловому секторі та робототехніці HMI є частиною ширшої екосистеми. Проектування інтерфейсу тут невід'ємне від: роботи з протоколами реального часу (наприклад, MQTT для Smart Home систем або OPC UA для виробничих ліній); візуалізації даних із сенсорів та датчиків у середовищах на кшталт LabVIEW або Multisim, де інтерфейс повинен відображати стан фізичних процесів з мінімальною затримкою.

Інженерія інтерфейсів тепер включає суворе дотримання стандартів WCAG (Web Content Accessibility Guidelines). Це означає, що на етапі архітектури закладається підтримка скрінрідерів, керування з клавіатури та семантична коректність коду. Це перетворює розробку на процес створення універсальних систем, здатних адаптуватися до різних фізичних обмежень користувача.

Зі збільшенням обсягів даних, що обробляються в реальному часі (особливо в системах автоматизації та інженерного моніторингу), традиційні підходи до керування станом системи стикаються з проблемою «архітектурної в'язкості».

В основі будь-якої розробки програмного забезпечення лежать дві фундаментальні парадигми, що визначають спосіб мислення інженера: імперативний підхід та декларативний.

Імперативний підхід є первинним. Він базується на детальному описі алгоритму: послідовності кроків та маніпуляцій, які комп'ютер має виконати для

досягнення цілі. У контексті веб-інженерії це пряме керування об'єктами DOM (Document Object Model), де розробник власноруч ініціює кожну зміну пікселя на екрані при зміні вхідних даних.

Декларативний підхід пропонує вищий рівень абстракції. Замість опису процесу «як зробити», інженер описує кінцевий результат - «що має бути». Реалізація технічних деталей перекладається на інтелектуальний рушій (браузер або фреймворк), який автоматично синхронізує модель даних із візуальним представленням [90].

Актуальність зумовлена тим, що сучасні стандарти стрімко рухаються в бік декларативності. Це дозволяє вирішити ключову проблему комп'ютерної інженерії - мінімізацію побічних ефектів при масштабуванні систем. Перехід до декларативних методів - це не просто зміна синтаксису, а стратегічний крок до створення відмовостійкого ПЗ, де інтерфейс є передбачуваним відображенням стану системи.

У сучасній інженерії інтерфейсів користувача декларативна парадигма стала однією з домінуючих підходів до створення, опису та управління графічними інтерфейсами. Вона радикально відрізняється від традиційної імперативної моделі, перетворюючи процес розробки з низькорівневого маніпулювання елементами на високорівневий опис бажаного стану системи.

Концептуальний аналіз цієї парадигми дозволяє розкрити її філософські, архітектурні та практичні виміри, зрозуміти її переваги, обмеження та вплив на еволюцію Human-Computer Interaction (HCI).

Декларативне програмування як парадигма сягає корінням у функціональне та логічне програмування (наприклад, Lisp, Prolog), де акцент робиться не на послідовності команд («як» виконати дію), а на описі результату («що» має бути отримано).

У контексті інтерфейсів ця ідея матеріалізувалася в моделях, таких як HTML/CSS (декларативний опис структури та стилю веб-сторінок), XAML у WPF, а пізніше - у сучасних фреймворках на кшталт React (з його JSX і Virtual DOM), Flutter (widget tree), SwiftUI та Jetpack Compose.

Ранні моделі Model-Based User Interface Development Environments вже у 1990-х роках пропонували використовувати декларативні моделі (abstract presentation model, concrete presentation model, тощо) для підвищення абстракції, повторного використання та автоматизації генерації інтерфейсів.

Декларативний підхід дозволяв описувати структуру, поведінку та взаємозв'язки UI-елементів на концептуальному рівні, делегуючи рутинні операції (рендеринг, оновлення, layout) фреймворку.

Імперативна парадигма, навпаки, домінувала в класичних інструментаріях (Win32 API, UIKit, Android View System): розробник вручну створював елементи, прив'язував обробники подій, управляв станом через прямі маніпуляції (наприклад, button.setText(), view.addSubview()). Це давало повний контроль, але призводило до «спагеті-коду», помилок синхронізації стану та складнощів підтримки.

Концептуально декларативна парадигма базується на кількох ключових принципах [91].

1. Опис стану, а не процесу - розробник декларує, яким має бути інтерфейс для заданого стану даних. Фреймворк самостійно обчислює різницю між попереднім і поточним станом та застосовує мінімальні зміни. У React це Virtual DOM, у Flutter - елементне дерево віджетів, у SwiftUI - структурний граф представлень.

2. Реактивність і односторонній потік даних - зміна стану автоматично тягне за собою оновлення UI (reactive programming). Це усуває проблему ручної синхронізації, типову для імперативних підходів.

3. Композиційність і абстракція - інтерфейс будується як композиція незалежних, декларативних компонентів (views, widgets, composables). Кожен компонент - це чисте відображення стану (pure function of state), що полегшує тестування, повторне використання та розуміння коду.

3. Розділення - логіка представлення відокремлюється від імперативної бізнес-логіки та низькорівневих деталей рендерингу. Фреймворк бере на себе «як» (як обчислити layout, анімувати зміни, оптимізувати перерисовку).

Ця парадигма тісно пов'язана з функціональним програмуванням: компоненти часто є чистими функціями, стан підіймається, а побічні ефекти контролюються явно.

Декларативний підхід радикально підвищує продуктивність і надійність інженерії інтерфейсів:

- зрозумілість і підтримуваність - код читається як опис бажаного результату, а не як послідовність маніпуляцій. Це зменшує когнітивне навантаження;

- менше помилок - автоматичне управління оновленнями усуває класичні баги «UI не оновився» або «стан розсинхронізовано»;

- кросплатформенність і адаптивність - один декларативний опис може трансформуватися в різні конкретні реалізації;

- автоматизація та інструменти - легше генерувати код з дизайн-систем, виконувати статичний аналіз, застосовувати формальні методи верифікації;

- масштабованість - у великих проєктах композиція компонентів і реактивний потік даних дозволяють ефективно керувати складністю.

Дослідження та практика показують, що перехід на декларативні фреймворки (React, SwiftUI, Compose) значно прискорює розробку та знижує кількість дефектів у порівнянні з імперативними попередниками.

Проте декларативна парадигма не є універсальною панацеєю. Концептуальний аналіз виявляє низку нюансів:

- прихована імперативність - під капотом фреймворки все одно виконують імперативні операції (diffing, patching DOM, traversal дерева). Розробник іноді втрачає контроль над точними кроками оптимізації;

- продуктивність у складних сценаріях - надмірна реактивність може призводити до зайвих перерисовок; потрібні тонкі механізми мемоїзації;

- крива навчання - перехід від імперативного мислення («як оновити цей елемент») до декларативного («яким має бути UI у цьому стані») вимагає зміни ментальної моделі;

- debugging - коли «магія» фреймворку не спрацьовує, важче зрозуміти, чому саме UI не відповідає декларації.

- гібридні системи - у реальних проєктах часто виникає потреба в імперативних втечах (наприклад, direct DOM access у React чи UIViewRepresentable у SwiftUI).

Крім того, модель-based підходи 1990–2000-х років, які були суто декларативними на рівні абстрактних моделей, часто страждали від недостатньої гнучкості та складності інструментів, що обмежило їх широке впровадження порівняно з сучасними «легкими» декларативними фреймворками.

Концептуальний аналіз декларативної парадигми в інженерії інтерфейсів дозволяє не лише порівняти її з імперативною, а й зрозуміти глибші філософські зрушення в програмуванні HCI: перехід від «програміст керує машиною» до «програміст описує намір, машина оптимізує виконання».

Це відображає ширшу тенденцію до підвищення рівня абстракції, автоматизації та гуманізації розробки - інтерфейси стають ближчими до природного опису, а не до машинних інструкцій.

У майбутньому декларативна парадигма, ймовірно, еволюціонуватиме в бік ще вищої абстракції: AI-assisted генерації UI з описів, формальних верифікацій декларативних специфікацій, інтеграції з domain-specific languages для дизайну та глибшого поєднання з реактивними та функціональними архітектурами.

Таким чином, декларативна парадигма - це не просто технологічний тренд, а фундаментальна зміна способу мислення про побудову інтерфейсів. Її концептуальний аналіз розкриває, як опис «що» замість «як» трансформує продуктивність, якість і креативність у сучасній інженерії користувацьких інтерфейсів, відкриваючи нові горизонти для досліджень і практики.

Для глибшого розуміння декларативної парадигми в інженерії інтерфейсів користувача важливо провести систематичне порівняння з класичною імперативною парадигмою [92]. Нижче розглянемо порівняння імперативної та декларативної парадигми інженерії інтерфейсів (табл. 1).

Таблиця 1.

Порівняння імперативної та декларативної парадигми

Аспект	Імперативна парадигма	Декларативна парадигма
Основна філософія	«Як» виконати дію	«Що» має бути отримано
Ментальна модель	Програміст керує машиною крок за кроком	Програміст описує намір, машина оптимізує виконання
Походження	Фон-нейманівська архітектура	Функціональне та логічне програмування
Спосіб опису UI	Прямі команди створення та модифікації елементів	Опис структури та вигляду як функції від стану
Оновлення інтерфейсу	Ручне	Автоматичне
Читабельність коду	Низька при зростанні проєкту (спагеті-код)	Висока (код читається як опис бажаного результату)
Продуктивність розробки	Повільніша в великих проєктах	Значно вища завдяки меншому обсягу коду та меншій кількості помилок
Кросплатформенність	Складна (потрібен окремий код для кожної платформи)	Легша (один декларативний опис для декількох платформ)
Керування станом	Програміст відповідає за мутацію та синхронізацію стану	Стан зазвичай незмінний (immutable), потік даних однонаправлений
Побічні ефекти	Поширені, важкопередбачувані	Мінімізовані або чітко ізольовані
Масштабованість	Складна через жорстку залежність від послідовності кроків	Вища, легше адаптувати до паралельних обчислень
Відлагодження (Debugging)	Фокусується на «де саме в ланцюжку кроків сталася помилка»	Фокусується на «чому опис стану не відповідає результату»
Тестування	Потребує перевірки всієї послідовності дій (Side-effects)	Легше тестувати окремі компоненти/функції (Pure functions)

Розглянемо на прикладах порівняння використання імперативної та декларативної парадигми. В якості першого прикладу розглянемо кнопку, яка при натисканні збільшує лічильник та змінює колір фону кнопки. Використання імперативного підходу, демонструє модель, де розробник бере на себе роль "диспетчера станів". В наведеному лістингу чітко простежується лінійна залежність, яка при ускладненні системи перетворюється на проблему.

```
<!doctype html>
<html lang="uk">
  <head>
    <meta charset="UTF-8" />
    <title>Імперативний лічильник</title>
    <style>
      button {
        padding: 15px 25px;
        font-size: 18px;
      }
    </style>
  </head>
  <body>
    <button id="btn">Лічильник: 0</button>
    <script>
      const button = document.getElementById('btn');
      let count = 0;

      button.addEventListener('click', () => {
        count++; // 1. Змінюємо стан

        button.textContent = `Лічильник: ${count}`;

        if (count % 2 === 0) {
        button.style.backgroundColor = '#e74c3c';
          button.style.color = 'white';
        } else {
        button.style.backgroundColor = '#3498db';
        button.style.color = 'white';
        }
      });
    </script>
  </body>
</html>
```

1. Мутація стану: count++: стан змінюється безпосередньо, без створення імутабельного знімка.

Пряме редагування значення змінної здається природною, але в контексті розробки ПЗ вона створює проблему непередбачуваності.

Коли ми мутуємо об'єкт або змінну, ми втрачаємо її попередній стан. Це робить неможливим впровадження функцій на кшталт «Undo/Redo» або ефективне налагодження через «Time-travel debugging».

Якщо програма велика, важко зрозуміти, яка саме функція і в який момент змінила значення. На противагу цьому, імутабельність (створення нового знімка даних) дозволяє системі легко порівняти «старе» та «нове» значення, щоб точно визначити, чи потрібно взагалі оновлювати екран.

2. Ручна синхронізація DOM: button.textContent = ... програма повинна "знати", який саме елемент потрібно оновити. Якщо ідентифікатор btn зміниться в HTML, скрипт перестане працювати.

Цей аспект створює жорстку залежність між логікою (JS) та структурою (HTML), що називається tight coupling.

Скрипт стає «заручником» структури документа. Зміна id або класу в HTML-шаблоні миттєво перетворює працюючий код на неробочий, причому помилка часто виявляється лише під час виконання програми (runtime).

Розробник змушений тримати в голові всю карту зв'язків: «Змінна X пов'язана з елементами A, B та C». При додаванні нових індикаторів кількість таких зв'язків зростає геометрично, що неминуче призводить до помилок синхронізації.

3. Логічне розгалуження стилізації: конструкція if (count % 2 = = = 0) змушує JS-код відповідати за візуальний вигляд, що порушує принцип розподілу відповідальності. Коли конструкції керують зовнішнім виглядом безпосередньо з JavaScript, порушується фундаментальний принцип Separation of Concerns (розподіл відповідальності).

Змішування бізнес-логіки та UI: обчислення парності - це математична логіка, а зміна кольору фону або тексту - це візуальна презентація. Коли вони переплітаються, дизайнер не може змінити вигляд елемента, не втручаючись у код програміста.

Замість того, щоб використовувати потужні декларативні можливості CSS, ми змушуємо браузер виконувати зайві маніпуляції зі стилями через JS-рушій. Це знижує продуктивність на малопотужних контролерах або мобільних пристроях.

Даний код демонструє три основні проблеми, що заважають масштабуванню.

1. Логіка оновлення розкидана по кількох місцях: припустимо, нам потрібно додати ще один напис, який показує, чи є число парним. Доведеться зайти всередину обробника події click і додати ще один рядок коду для елемента.

Ця проблема є фундаментальним обмеженням директної маніпуляції DOM, де інтерфейс і дані жорстко зчеплені. Кожна нова функція відображення змушує розробника втручатися в уже налагоджену логіку обробки подій, що експоненціально збільшує кількість потенційних точок відмови.

Коли обробник події виконує одночасно і розрахунки, і оновлення кількох елементів UI, стає важко гарантувати, що зміна одного рядка коду не зламає роботу інших частин інтерфейсу.

У складних системах (наприклад, панелях керування робототехнікою) одна зміна стану може вимагати оновлення десятків індикаторів; ручне керування кожним із них усередині одного click-обробника робить код практично нечитабельним.

Якщо значення числа може змінюватися не лише через клік (наприклад, через зовнішній сигнал від датчика або таймер), вам доведеться дублювати всю логіку оновлення написів у кожному новому місці, де змінюються дані.

Для вирішення цієї проблеми в сучасній інженерії використовують реактивне програмування та патерн Observer. У такому випадку обробник події лише змінює значення змінної (стан), а всі зацікавлені елементи інтерфейсу автоматично «підписуються» на цю зміну та оновлюються самостійно, незалежно один від одного.

2. Ручне керування залежностями: при додаванні нових елементів, які залежать від count, інженеру доведеться вручну прописувати оновлення для кожного з них у кожному місці, де змінюється count.

В імперативній парадигмі розробник бере на себе роль «двигуна синхронізації». При додаванні будь-якого нового елемента, стан якого залежить від змінної count (наприклад, індикатора прогресу, доступності кнопки або текстового опису), інженер змушений вручну ініціювати оновлення кожного з цих вузлів DOM.

Це створює низку критичних проблем:

1. Фрагментація логіки: код оновлення інтерфейсу розпорошується по всій кодовій базі. Якщо count змінюється в п’яти різних функціях, у кожній із них необхідно продублювати виклики для оновлення всіх залежних елементів.

2. Ризик втрати синхронізації (Stale UI): зі зростанням проєкту стає дедалі важче відстежити повний перелік залежних компонентів. Пропуск хоча б одного виклику призводить до ситуації, коли інтерфейс відображає застарілі або суперечливі дані.

3. Складність рефакторингу: будь-яка зміна структури DOM або назв ідентифікаторів вимагає переписування десятків прямих звернень, що робить підтримку коду надзвичайно трудомісткою та схильною до регресійних помилок.

4. Ефект «снігової кулі»: кожна нова залежність експоненціально збільшує кількість зв’язків, які програміст повинен тримати в ментальній моделі, що зрештою призводить до появи некерованого «спагеті-коду».

3. Ризик десинхронізації: найменша логічна помилка або порушення послідовності в ланцюжку імперативних команд неминуче призводить до розриву між фактичним станом системи та його візуальним представленням. У складних інтерфейсах це створює ефект «застиглого стану», коли внутрішні змінні програми вже оновилися, а користувач бачить застарілі дані. Для систем реального часу (RT-систем) така розбіжність є критичною вразливістю: затримка або помилка у відображенні показників датчиків чи системних сповіщень може призвести до прийняття хибних рішень оператором або автоматикою.

Проблема прямого маніпулювання властивостями: використання прямих низькорівневих маніпуляцій, таких як звернення до style або textContent у

JavaScript, є виправданим лише на етапі швидкого прототипування або при вирішенні ізольованих мікро-завдань.

У промисловій розробці такий підхід провокує появу «крихкого» коду, де логіка представлення (UI) занадто тісно переплетена з бізнес-логікою. Це унеможливлює масштабування системи, оскільки будь-яка зміна структури документа (DOM) вимагає повного перегляду методів доступу до його елементів.

В архітектурі сучасних комп'ютерних систем імперативне маніпулювання інтерфейсом вважається ресурсовитратним не лише з точки зору обчислювальних потужностей (через ризики частих перемальовувань - reflow/repaint), а й через високу вартість підтримки.

Когнітивне навантаження на інженера зростає експоненціально: замість опису того, що система має відображати, програміст змушений утримувати в пам'яті тисячі мікро-зв'язків між подіями та елементами. Це робить процес відлагодження (debugging) та впровадження нового функціоналу надто дорогим і тривалим, що суперечить сучасним стандартам гнучкої розробки та надійності.

Декларативний підхід, базується на концепції реактивності. Замість прямої маніпуляції властивостями DOM-елементів, розробник визначає залежність інтерфейсу від стану даних.

На відміну від прямого маніпулювання DOM (де кожна зміна - це дорога операція), цей код використовує Virtual DOM.

Коли викликається setCount, React не біжить одразу змінювати вузол у браузері. Він створює нове віртуальне дерево компонентів. Потім порівнює старе дерево з новим. Лише якщо він знайде різницю (наприклад, текст змінився з «0» на «1», а колір з червоного на синій), він застосує мінімально необхідний набір патчів до реального DOM.

style={{ ... }} - це приклад підходу CSS-in-JS. Стилі описані не як статичні рядки, а як об’єкт JavaScript. Це дозволяє використовувати всю потужність мови (тернарні оператори, змінні, логіку) прямо всередині властивостей стилю.

Це демонструє реактивний декларативний компонент.

```
import React, { useState, useCallback } from "react";
import ReactDOM from "react-dom";

function App() {
  const [count, setCount] = useState(0);
  const isEven = count % 2 === 0;
  const handleClick = useCallback(() => {
    setCount((prev) => prev + 1);
  }, []);
  return (
    <button
      style={{
        padding: "15px 25px",
        fontSize: "18px",
        backgroundColor: isEven ? "#e74c3c" : "#3498db",
        color: "white",
        border: "none",
        borderRadius: "6px",
        cursor: "pointer",
        transition: "background-color 0.2s",
      }}
      onClick={handleClick}
    >
      Лічильник: {count}
    </button>
  );
}
const rootElement = document.getElementById("root");
ReactDOM.render(<App />, rootElement);
```

Змінна isEven виступає в ролі перемикача. В імперативному коді довелося б писати element.style.backgroundColor = '...', що створює ризик помилки в назві властивості або кольору. Тут опис стилю є декларативним і стабільним.

Компонент App є функціональним. Це означає, що при однакових вхідних даних (props) та стані (state) він завжди поверне ідентичний HTML-результат. Кнопка «не знає» про зовнішній світ. Вона лише реагує на внутрішній стан count.

Такий код легко тестувати автоматизованими засобами, оскільки для перевірки логіки достатньо імітувати зміну стану та перевірити фінальну розмітку.

Використання useCallback з порожнім масивом залежностей [] та функціонального оновлення стану setCount((prev) => prev + 1) вирішує проблему замикань. Навіть якщо компонент перерендериться багато разів, handleClick завжди матиме доступ до найсвіжішого значення count через аргумент prev. Це запобігає ситуаціям «race conditions», коли декілька швидких натискань могли б призвести до некоректного підрахунку в імперативній моделі.

Використання хука useState дозволяє інкапсулювати дані всередині компонента. Зміна стану через функцію setCount ініціює автоматичний цикл перемальовування (re-rendering), керований ядром бібліотеки.

У коді відсутні виклики на кшталт button.textContent = ... або button.style.backgroundColor = ... Розробник лише декларує: "Колір кнопки залежить від змінної isEven". Система самостійно порівнює попередній стан із новим (процес Reconciliation) і вносить мінімально необхідні зміни в дерево відображення.

Обчислювальні властивості (Derived State): змінна isEven не є окремим станом, який треба синхронізувати вручну. Вона обчислюється автоматично при кожному оновленні компонента на основі базового стану count.

Таким чином, перехід до декларативних методів проектування адаптивних інтерфейсів є необхідною умовою для створення надійного та масштабованого програмного забезпечення. Така парадигма відповідає принципам автоматизованого керування, де зворотний зв'язок та стан об'єкта автоматично визначають керуючий вплив на виконавчі механізми.

Розглянемо другий приклад – фільтрацію даних.

Фільтрація даних - це процес створення підмножини елементів із більшої колекції на основі певного критерію (предиката). Це одна з найчастіших задач у веб-розробці, наприклад, при пошуку товарів у каталозі або сортуванні користувачів за ролями.

Фільтрація даних - це не просто технічна операція, а ключовий елемент забезпечення зручності користувача (UX) та ефективності системи. У сучасних

інтерфейсах вона трансформувалася з базового перебору масивів у складну систему динамічного відсіювання об’єктів у реальному часі.

Предикат виступає як логічний «фільтр-бар’єр», який для кожного елемента повертає булеве значення (true або false). Це дозволяє створювати гнучкі ланцюжки умов, де користувач може одночасно шукати за назвою, діапазоном цін та наявністю певних технічних характеристик.

У контексті сучасних веб-технологій фільтрація часто відбувається без перезавантаження сторінки. Це вимагає від розробника розуміння алгоритмічної складності: при роботі з великими наборами даних (Big Data) фільтрація на стороні клієнта може сповільнювати інтерфейс, тому часто переноситься на сторону сервера або оптимізується через індексацію.

Якісно реалізована фільтрація зменшує когнітивне навантаження на користувача, дозволяючи швидко виокремити релевантну інформацію з інформаційного шуму. В інженерних системах, наприклад у моніторингу сенсорів робототехніки, фільтрація за пороговими значеннями є критично важливою для виявлення аномалій та запобігання аварійним ситуаціям.

Основні характеристики фільтрації є:

- вхід: список елементів + функція-предикат (яка повертає True або False для кожного елемента). Дані: вхідний масив може містити будь-які типи даних - від примітивів (чисел, рядків) до складних об'єктів або структур.

Предикат: це чиста функція, яка діє як «фільтр» або «сито». Вона приймає поточний елемент і повертає булеве значення. Такий підхід дозволяє відокремити логіку відбору (що саме ми шукаємо) від механізму ітерації (як саме ми проходимо по списку), що робить код універсальним.

- вихід: новий список, що містить лише елементи, для яких предикат повернув True. Результатом операції є нова колекція, яка є підмножиною вихідної. Кількість елементів у вихідному списку варіюється від 0 (якщо жоден елемент не задовольнив умову) до n (якщо умову задовольнили всі), де n - довжина вхідного списку. Важливо, що типи елементів у вихідному списку

залишаються ідентичними вхідним; фільтрація змінює лише кількість об'єктів, але не їхню структуру.

- відсутність зміни оригінального списку. У декларативній парадигмі фільтрація не повинна мати побічних ефектів (side effects).

Оригінальний список залишається недоторканим у пам'яті. Це критично для багатопотокових систем та розробки інтерфейсів (наприклад, у React чи Redux), оскільки незмінність даних дозволяє легко відстежувати зміни та уникати важкопередбачуваних помилок, пов'язаних із неочікуваною мутацією стану.

Завжди можна повернутися до вихідних даних або застосувати до них інший фільтр незалежно від попередніх операцій.

- порядок елементів зберігається. Фільтрація - це стабільна операція щодо структури колекції. Процес зазвичай відбувається шляхом лінійного обходу вхідних даних (зліва направо). Елемент, який був третім у вхідному списку, ніколи не опиниться перед елементом, який був першим, якщо обидва вони пройшли фільтр.

Передбачуваність гарантує, що якщо дані були попередньо відсортовані (наприклад, за датою чи пріоритетом), цей порядок буде збережено і в результаті фільтрації, що позбавляє потреби у повторному сортуванні.

- відносний порядок елементів, що пройшли фільтр, не змінюється. Навіть якщо між двома елементами, що пройшли фільтр, було видалено десять інших, їхнє взаємне розташування (хто «старший», а хто «молодший» у списку) залишається незмінним.

Архітектурна значущість важлива для алгоритмів, де позиція елемента має значення (наприклад, при обробці кадрів відеопотоку або черги повідомлень у системах реального часу), оскільки фільтрація не порушує часову або логічну послідовність подій.

Проаналізуємо як можемо вирішити цю задачу використовуючи два різні підходи.

Імперативний підхід передбачає вказівку крок за кроком як потрібно робити фільтрацію. Потрібно створити порожній "контейнер" для результату.

Використовуємо цикли (for, while), умовні оператори та мутацію стану. Вручну звернутися до кожного елемента за індексом. Перевірити умову через if. Явно додати елемент, що підходить, у контейнер.

Декларативний підхід описує що хочемо отримати, а не як це робити. Використовуємо вбудовані функції вищого порядку: filter(), list comprehensions, або бібліотеки типу functools. Код коротший, читабельніший і менш схильний до помилок. Метод сам обходить масив і вирішує, як створити новий масив. Вихідний масив залишається незмінним, що є стандартом розробки.

Уявімо задачу: у нас є список студентів, і нам потрібно відфільтрувати тих, хто набрав 60 або більше балів.

Спочатку зробимо це за допомогою імперативного підходу:

```
const students = [
    { name: 'Ганна', score: 85 },
    { name: 'Дмитро', score: 45 },
    { name: 'Михайло', score: 72 },
    { name: 'Людмила', score: 58 }
];

const passingStudents = [];

for (let i = 0; i < students.length; i++) {
    // 3. Вручну перевіряємо умову
    if (students[i].score >= 60) {
        // 4. Явно додаємо елемент
        passingStudents.push(students[i]);
    }
}
```

Перші рядки створюють основу для роботи. В якості вихідних даних в нас створюється масив з 4 об’єктів-студентів.

Масив students: це наше джерело даних. Кожен об'єкт у масиві має дві властивості: name (ім'я) та score (бал).

Масив passingStudents: створюємо порожній «кошик». Оскільки працюємо в імперативному стилі, маємо заздалегідь підготувати місце, куди будемо власноруч складати результати.

Цикл - це «двигун» нашого алгоритму. В імперативному підході бере на себе повне керування ітерацією.

Конструкція циклу: for (let i = 0; i < students.length; i++).

let i = 0: починаємо з першого елемента (індекс 0).

i < students.length: працюємо, поки не дійдемо до кінця списку.

i++: після кожного кроку переходимо до наступного елемента.

Розглянемо логіку всередині.

Доступ за індексом: ми звертаємося до кожного студента через students[i].

Перевірка умови (фільтрація): if (students[i].score >= 60) - кажемо програмі зупинитися і подивитися на бал конкретного студента.

Дія (мутація): якщо умова істинна, ми викликаємо метод .push(). Буквально беремо об'єкт і «заштовхуємо» його в наш новий масив. Тобто, ми діємо як мікроменеджери. Не просто кажемо: «Дай мені список успішних студентів», ми покрового виконуємо всі етапи, наче прописуючи їх:

1. «Візьми порожній аркуш».

2. «Стань на початок списку».

3. «Подивись на бал першого студента».

4. «Якщо він більше 60, запиши ім'я на мій порожній аркуш».

5. «Перейди до наступного студента».

6. «Повторюй, поки список не закінчиться».

Імперативна фільтрація полягає в:

- створенні порожнього масиву const passingStudents = [], саме сюди ми будемо збирати студентів, які пройшли;

- описі циклу for крок за кроком; і починається з 0 і збільшується на 1 (i++) кожного разу. Цикл виконується стільки разів, скільки елементів у масиві students (4 рази);

- ручній перевірці умови хто набрав 60 або більше балів;

- додаванні елемента до нового масиву.

Після виконання коду масив passingStudents буде містити наступний результат:

```
[{
  name: "Ганна",
  score: 85
}, {
  name: "Михайло",
  score: 72
}]
```

Після завершення циклу, стан програми буде наступним:

Ганна (85): Умова 85 ≥ 60 є істинною (true) - додано до масиву.

Дмитро (45): Умова 45 ≥ 60 є хибною (false) - ігнорується.

Михайло (72): Умова 72 ≥ 60 є істинною (true) - додано до масиву.

Людмила (58): Умова 58 ≥ 60 є хибною (false) - ігнорується.

В імперативному підході створили механізм "сита". Кожен елемент масиву students пройшов через перевірку if. Наш масив passingStudents тепер містить посилання на ті самі об'єкти, що й оригінальний масив. Оскільки цикл йшов від i = 0 до кінця, Ганна йде першою, а Михайло другим - точно так само, як вони розташовувалися в початковому списку. Довжина масиву passingStudents.length дорівнює 2, хоча оригінальний масив students.length залишається рівним 4. Імперативний підхід не змінює вхідні дані, якщо ви самі цього не пропишете.

Розглянемо як буде виглядати декларативна версія цієї ж фільтрації.

```
const students = [
    { name: 'Ганна', score: 85 },
    { name: 'Дмитро', score: 45 },
    { name: 'Михайло', score: 72 },
    { name: 'Людмила', score: 58 }
    ];
const passingStudents = students.filter(student => student.score >= 60);

console.log(passingStudents);
```

Код створює масив студентів і одним рядком відфільтровує тільки тих, у кого бал (score) ≥ 60. Результат виводиться в консоль.

В якості вихідних даних в нас створюється масив з 4 об’єктів-студентів. Кожен студент має два поля: name (ім’я) та score (бал). Масив не змінюється надалі (const). Ми не пишемо, як саме фільтрувати (цикл, індекси, push, тощо). Ми просто описуємо, що хочемо отримати.

Метод filter() - це вбудований інструмент масивів у JavaScript, який створює новий масив з усіма елементами, що пройшли перевірку.

students: масив, який ми хочемо "просіяти".

.filter(...): функція вищого порядку. Вона сама запускає цикл "під капотом", перебираючи кожен елемент масиву.

student => ...: це стрілкова функція (callback). Ви кажете методу: «Для кожного окремого об'єкта в масиві (назвимо його student) виконай наступну дію».

student.score >= 60: Це логічна умова (предикт). Якщо умова повертає true - студент залишається в новому масиві. Якщо false - студент відкидається.

Коли ви викликаєте .filter(), JavaScript виконує наступні дії:

1. Створює новий порожній масив (внутрішньо).

2. Бере перший елемент (Ганна, 85). Питає: 85 ≥ 60? - Так. Кладе Ганну в новий масив.

3. Бере другий елемент (Дмитро, 45). Питає: 45 ≥ 60? - Ні. Ігнорує його.

4. Бере третій елемент (Михайло, 72). Питає: 72 ≥ 60? - Так. Кладе Михайла в новий масив.

5. Бере четвертий елемент (Людмила, 58). Питає: 58 ≥ 60? - Ні. Ігнорує її.

6. Повертає готовий масив і записує його в змінну passingStudents.

Тобто, результат присвоюється в нову константу passingStudents. Оригінальний масив students не змінюється, .filter() завжди повертає новий масив.

Вибір між імперативним та декларативним підходами (рис. 1) - це стратегічне рішення, яке впливає на продуктивність розробки, надійність системи та вартість її підтримки. Вибір часто залежить від того, на чому зосереджується увага: на контролі процесу чи на описі логіки [92].

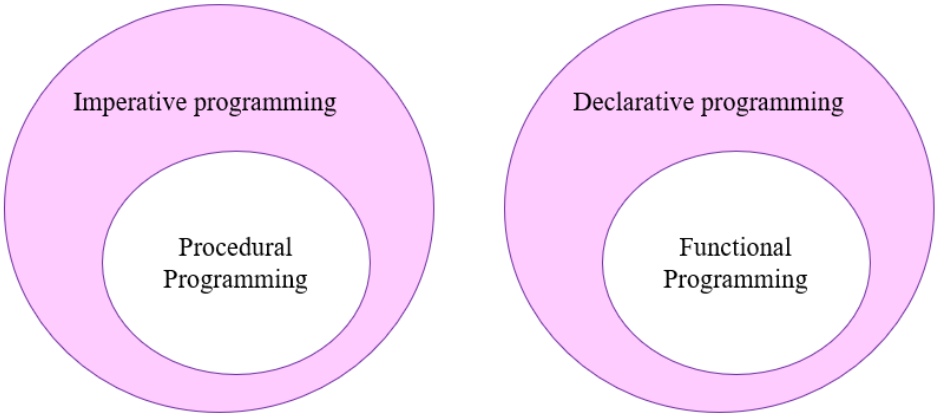


Рисунок 1. Імперативний та декларативний підхід

Джерело рисунка: розробка автора.

Ця діаграма демонструє ієрархічну класифікацію парадигм програмування, де зовнішні кола представляють загальні концепції (філософію написання коду), а внутрішні - їхні конкретні реалізації або відгалуження.

Ліва частина: імперативна вертикаль - частина діаграми ілюструє підхід, заснований на командах та зміні стану.

Imperative programming (зовнішнє коло) - це «наказовий» стиль. Програма сприймається як послідовність інструкцій, які крок за кроком змінюють стан комп'ютера (значення в пам'яті). Ви буквально кажете машині: «візьми це значення, додай до нього п'ять, збережи результат».

Procedural Programming (внутрішнє коло) - це еволюція імперативного підходу. Замість суцільного полотна команд код розбивається на блоки - процедури або підпрограми. Воно знаходиться всередині, бо повністю наслідує імперативну логіку (покроковість, змінні, цикли), але додає механізм повторного використання коду.

Права частина: декларативна вертикаль - частина описує підхід, що базується на визначенні результату та логічних зв'язках.

Declarative programming (зовнішнє коло): фокус зміщується з «як зробити» на «що отримати». Програміст описує властивості кінцевого результату або логічні умови, а система сама вирішує, якими алгоритмами це реалізувати. Це вищий рівень абстракції, де деталі керування пам'яттю чи процесором приховані.

Functional Programming (внутрішнє коло): це специфічна та сувора форма декларативного стилю. Програма будується як набір математичних функцій. Воно входить до складу декларативного програмування, оскільки замість зміни стану через команди, воно використовує обчислення виразів. Тут немає «змінних» у звичному розумінні - є лише вхідні дані та результат трансформації.

Те, що одне коло знаходиться всередині іншого, означає відношення «є видом». Тобто, будь-яка процедурна мова (як C або Pascal) за своєю суттю є імперативною. Будь-яка суто функціональна мова (як Haskell) є декларативною.

Діаграми навмисно рознесені в сторони. Це підкреслює фундаментальну різницю в мисленні: імперативний підхід керує процесом, а декларативний - даними та логікою.

Для розробника ця діаграма слугує картою: якщо ви пишете код на C, ви перебуваєте в лівій зоні. Якщо ви створюєте запити на SQL або верстаєте на HTML/CSS, ви працюєте в правій, декларативній зоні. Розуміння того, де саме ви знаходитесь, допомагає правильно обирати інструменти для вирішення конкретних інженерних завдань.

Визначимо критерії, які допоможуть зробити правильний вибір:

1. Складність бізнес-логіки та стану.

Якщо Ваш інтерфейс або система мають багато залежних елементів; зміна однієї змінної має оновити п'ять різних текстових полів, змінити колір графіків та розблокувати кнопку - декларативність врятує вас від помилок синхронізації.

Якщо логіка лінійна та проста, наприклад, разовий скрипт для обробки файлу або алгоритм, де кожен крок має виконуватися в суворо визначеному порядку з прямим доступом до пам'яті краще обрати імперативний підхід.

2. Продуктивність та ресурси

Якщо працюєте з мікроконтролерами або високонавантаженими обчисленнями потрібен повний контроль над кожним циклом процесора та виділенням пам'яті. Тут абстракції декларативних систем можуть бути занадто "важкими".

При розробці сучасних веб-інтерфейсів чи складних програмних комплексів сучасні рушії (браузери, віртуальні машини) оптимізовані настільки добре, що їхні внутрішні алгоритми оновлення часто працюють швидше, ніж ручний код середнього програміста.

3. Масштабованість та команда

Декларативний підхід кращий для великих команд. Код стає самодокументованим: ви читаєте "що має статися", а не "як ми бігаємо по масивах". Це зменшує кількість багів при передачі проєкту іншим розробникам.

Імперативний підхід вимагає детального коментування та суворого дотримання архітектурних патернів, інакше через пів року код перетвориться на "спагеті", де ніхто не знає, яка функція змінила глобальну змінну.

Використання нативних декларативних механізмів у сучасному вебі дозволяє значно спростити архітектуру інтерфейсу, переносячи логіку керування станом з JavaScript безпосередньо в HTML та CSS. Це підвищує продуктивність, оскільки браузер оптимізує ці переходи на нижньому рівні.

Розглянемо нативні декларативні механізми керування станом та вибором елементів.

Взаємодія з користувачем часто виходить за межі плоскої сторінки. Для керування увагою використовуються додаткові шари, які мають чітку ієрархію - від глобальних контейнерів до дрібних допоміжних елементів.

Традиційно для створення модальних вікон або контекстних меню розробники використовували складні JS-сценарії для маніпуляції властивостями display або z-index. Стандарти HTML та CSS пропонують декларативну альтернативу.

Popover - це будь-які спливаючі елементи в інтерфейсі: підказки, що розкриваються меню, превью товарів та інші.

У веб-дизайні є велика група оверлеїв (overlays). Далі, всередині оверлеїв, коло елементів звужується до конкретніших попапів (popups). Зазвичай, це невеликі вікна, які з'являються автоматично.

Оверлей - це найвищий рівень абстракції для будь-якого елемента, що рендериться «поверх» основного контенту. Його головна технічна особливість - керування за допомогою z-index або використання спеціального верхнього шару браузера. Оверлеї створюють візуальну глибину, дозволяючи користувачеві взаємодіяти з додатковими функціями, не залишаючи поточної сторінки.

Всередині сімейства оверлеїв виділяють попапи (popups) - це динамічні елементи, що з’являються у відповідь на певну подію (клік, наведення або системне сповіщення). На відміну від статичних оверлеїв (наприклад, закріпленої бічної панелі), попапи завжди мають тимчасовий характер. Вони існують лише доти, доки виконується завдання або поки користувач їх не закриє.

Всередині категорії попапів існує фундаментальний поділ за типом поведінки:

Діалоги (Dialogs): Це функціональні вузли, що вимагають від користувача прийняття рішення. Найчастіше вони є модальними: блокують взаємодію з рештою сторінки, поки питання не буде вирішено (наприклад, підтвердження видалення файлу або форма входу). Це «критична» точка інтерфейсу.

Поповери (Popovers): На відміну від діалогів, поповери зазвичай немодальні. Вони пропонують додаткові дії або інформацію в контексті конкретного елемента (наприклад, меню вибору дати або панель налаштувань профілю). Користувач може вільно ігнорувати поповер і продовжувати роботу з основним контентом.

Найбільш специфічним елементом є підказки. Це мікроінтерфейси, які є підмножиною поповерів. Їхня єдина мета - надати коротку текстову довідку про об'єкт, на який вказує користувач. Через свою лаконічність вони займають найнижчий щабель в ієрархії, але є критично важливими для доступності та навчання користувача всередині складних систем.

Ця структура відображає логіку "наслідування" властивостей:

Функціональність: кожна підказка технічно реалізується як поповер, що має властивості попапа, який своєю чергою є оверлеєм.

Семантика: для реалізації цих рівнів використовуються специфічні теги та атрибути, як-от <dialog> для діалогів або атрибут popover для контекстних вікон, що дозволяє браузеру правильно обробляти фокус та доступність для програм зчитування екрана.

Переваги та недоліки компонентів:

Діалоги забезпечують максимальний фокус і чистоту основної сторінки, виносячи складні форми в окремий шар. Проте вони створюють високий рівень «тертя» (UX friction), агресивно перериваючи потік роботи, і потребують ретельної технічної реалізації для підтримки доступності.

Поповери гнучкі та зручні для вибору параметрів (фільтри, меню), оскільки дозволяють користувачеві бачити контекст і легко закривати вікно кліком поза його межами. Головний недолік - ризик випадкового перекриття важливого контенту та складність адаптації під малі екрани смартфонів.

Підказки - ідеальні для економії простору та навчання «на ходу», особливо в мінімалістичних інтерфейсах з іконками. Водночас вони практично не існують на тач-пристроях (через відсутність стану hover) і при надмірному використанні створюють візуальний шум, який відволікає від основної мети.

Одним із пріоритетних напрямів розвитку сучасної вебверстки є використання вбудованих декларативних можливостей HTML. Розглянемо приклад реалізації вікна сповіщення про стан технічного пристрою за допомогою атрибута popover.

Цей підхід дозволяє створити функціональний інтерфейс для моніторингу даних датчика без використання JavaScript, що значно спрощує структуру коду та підвищує його продуктивність.

Обов'язкові атрибути для написання коду:

- popover - вказує, що блок тепер поводиться як поповер;

- id - унікальний ідентифікатор блоку, що використовується для показу поповера;

- popovertarget - пов'язує елемент, наприклад, кнопку або інший HTML-тег з поповером. Як значення використовується ID елемента, який хочемо показати.

Реалізуємо виведення даних датчиком.

```
<button popovertarget="info-box">Вивести статус датчика
   </button>

<div id="info-box" popover>
  <p>Статус: Активний</p>
  <p>Температура: 27°C</p>
</div>
```

Браузер самостійно обробляє клік по кнопці, відкриття вікна, закриття при кліку поза ним та клавішу.

Насправді атрибут popover має два значення:

auto - значення за замовчуванням не потрібно прописувати явно. Одночасно з'являється лише один. Поповер може бути закритий на кліку за межами елемента;

manual – можна показати кілька одночасно.

Поповер не може бути закритий на кліку за його межами:

```
<button popovertarget="popover-auto">
  <code>popover="auto"</code>
</button>

<div id="popover-auto" popover="auto">
  <p>Закриваю інші спливаючі вікна. Закриваюсь сам.</p>
</div>

<button popovertarget="popover-manual">
  <code>popover="manual"</code>
</button>

<div id="popover-manual" popover="manual">
  <p>Не закриваю інші. Без кнопки не закриюсь.</p>
</div>
```

Цей код демонструє роботу сучасного Popover API, де головна відмінність між атрибутами auto та manual полягає в способі керування станом вікна.

Коли ви використовуєте popover="auto", браузер застосовує логіку «м'якого закриття»: такий елемент автоматично зникає при натисканні клавіші Esc або

кліку в будь-якому місці поза межами вікна, а також він є ексклюзивним, тобто відкриття одного автоматичного поповеру автоматично закриває всі інші подібні вікна на сторінці.

Натомість popover="manual" ігнорує зовнішні кліки та клавішу Esc, залишаючись видимим до тих пір, поки не буде ініційована конкретна команда закриття через кнопку або скрипт; такий режим дозволяє декільком спливаючим вікнам співіснувати на екрані одночасно, не конфліктуючи між собою.

Обидва типи елементів при активації переміщуються у так званий верхній шар документа (top layer), що дозволяє їм відображатися поверх усіх інших об'єктів без необхідності налаштування z-index, при цьому зв’язок між кнопкою-тригером та самим вікном реалізується через пару атрибутів popovertarget та id.

Тег <dialog> - це сучасний стандарт HTML5 для створення модальних та немодальних діалогових вікон (спливаючих вікон, підтверджень, форм входу) [93].

Раніше розробники були змушені використовувати складні комбінації , абсолютного позиціонування та JavaScript для керування фокусом і доступністю. Тепер браузери надають нативний інструмент, який бере на себе більшість інженерних завдань щодо взаємодії з користувачем.

Використання <dialog> кардинально змінює підхід до UI-розробки. Це декларативний елемент, який за замовчуванням прихований і має вбудовану логіку поведінки.

Головна перевага - він автоматично відповідає стандартам доступності, забезпечуючи коректну роботу скрінрідерів та керування клавіатурою, що раніше потребувало написання десятків рядків коду.

Робота з <dialog> базується на трьох основних китах: стані, методах відкриття та стилізації підкладки.

1. Два режими відображення: елемент може працювати у двох принципово різних режимах, залежно від викликаного методу JS.

Немодальний режим (show()) - коли ви викликаєте діалог через .show(), він поводиться як звичайний абсолютно позиційований елемент. Користувач може

вільно прокручувати сторінку, натискати на посилання або заповнювати форми поза межами діалогу. Вікно залишається в межах того контексту накладання, де воно було створене. Це означає, що воно може опинитися під іншими елементами, якщо у них вищий z-index. Псевдоелемент ::backdrop у цьому режимі не відображається, тому ви не можете автоматично затемнити сторінку. Можливе використання для плаваючих панелей інструментів, чати у кутку екрана або вікна сповіщень, що не потребують негайної дії.

Модальний режим (showModal()) - метод .showModal() активує спеціальну поведінку браузера, яка робить вікно пріоритетним об'єктом. Модальний діалог автоматично переміщується у "Top Layer". Він завжди буде поверх усіх інших елементів сторінки, незалежно від налаштувань z-index у CSS. Решта сторінки стає "інертною". Браузер ігнорує будь-які кліки, скрол або виділення тексту поза межами діалогу. При натисканні клавіші Tab фокус циклічно переміщується лише між кнопками та полями всередині діалогу. Поки вікно відкрите, користувач не зможе випадково перескочити на меню сайту чи футер. Браузер автоматично додає обробник клавіші Esc для закриття вікна. В цьому режимі з'являється спеціальний шар (::backdrop), який можна стилізувати (наприклад, розмити або затемнити фон), щоб візуально відокремити діалог від контенту.

2. Псевдоелемент ::backdrop

При використанні showModal() за діалогом з'являється спеціальний шар, який перекриває сторінку. Раніше для цього створювали окремий фоновий div. Тепер можна стилізувати його напряму через CSS.

```
dialog::backdrop {
  background-color: rgba(0, 0, 0, 0.5);
  backdrop-filter: blur(5px);
}
```

Це дозволяє реалізувати сучасні ефекти розмиття фону лише кількома рядками коду, не перевантажуючи DOM-дерево зайвими елементами.

::backdrop - спеціальний псевдоелемент, який браузер автоматично генерує для модальних елементів. Він знаходиться у так званому Top Layer (найвищому шарі), безпосередньо під самим вікном <dialog>, але поверх усього іншого контенту сторінки. Він слугує візуальним бар'єром, який допомагає користувачеві зосередитися на модальному вікні.

background-color: rgba(0, 0, 0, 0.5); - надає фону колір та прозорість. Використовується чорний колір (0, 0, 0) із рівнем непрозорості 50% (0.5). Це створює ефект легкого затінення всієї сторінки, поки модальне вікно активне. Це стандартний прийом в UX-дизайні для акцентування уваги.

backdrop-filter: blur(5px); - розмиває все, що знаходиться під цим шаром (тобто основний контент сайту). Значення 5px визначає силу розмиття. Це створює сучасний ефект "матового скла".

На відміну від звичайного filter: blur(), який розмиває сам елемент, backdrop-filter впливає лише на те, що проглядається крізь нього.

3. Інтеграція з формами (method="dialog")

Однією з найбільш корисних особливостей є спеціальний атрибут для форм усередині діалогу. Якщо форма має method="dialog", то при натисканні на кнопку відправки (submit) вікно автоматично закриється, а значення кнопки буде передано у властивість returnValue об'єкта діалогу. Це ідеально підходить для вікон підтвердження ("Так/Ні").

4. Керування станом та подіями

Для гнучкого контролю розробник має доступ до специфічних подій:

close: спрацьовує, коли діалог закривається.

cancel: спрацьовує, коли користувач натискає Esc (дозволяє скасувати закриття, якщо форма не заповнена).

Розглянемо приклад реалізації модального вікна на базі тега <dialog>, який демонструє сучасний підхід: мінімум JavaScript, використання нативних методів та автоматичне керування фокусом.

Ми використовуємо атрибут id для доступу через JS та спеціальну форму з method="dialog", яка закриває вікно без перезавантаження сторінки.

```
<button id="openModal">Відкрити налаштування</button>

<dialog id="settingsDialog">
  <form method="dialog">
    <h2>Налаштування профілю</h2>
    <p>Виберіть тему інтерфейсу:</p>

    <select name="theme">
      <option value="light">Світла</option>
      <option value="dark">Темна</option>
    </select>

   <div class="controls">
   <button type="button" id="cancelBtn">Скасувати</button>
   <button type="submit" value="confirm">Зберегти</button>
    </div>
  </form>
</dialog>
```

HTML-частина визначає «скелет» компонента та його початкову поведінку.

Тег <dialog>: Це контейнер, який браузер за замовчуванням рендерить з display: none. Він має спеціальний внутрішній стан. Якщо додати атрибут open прямо в HTML, вікно буде видимим одразу, але без блокування фону.

Атрибут id="settingsDialog": Необхідний для ідентифікації елемента в JavaScript. В сучасній розробці важливо уникати дублювання ID, щоб не зламати логіку звернення до DOM.

Конструкція <form method="dialog">: це «кілер-фіча» тега. Звичайні форми при натисканні на кнопку submit перезавантажують сторінку.

method="dialog" каже браузеру: «Не відправляй дані на сервер, а просто закрий цей діалог і збережи значення натиснутої кнопки у властивість returnValue».

Кнопка <button type="submit" value="confirm">: Значення атрибута value - це те, що ми отримаємо в JS після закриття. Це дозволяє легко розрізняти, чи користувач підтвердив дію, чи просто закрив вікно [94].

CSS відповідає за те, як вікно взаємодіє з візуальним простором користувача.

```
dialog {
  border: none;
  border-radius: 8px;
  padding: 20px;
  box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);
  width: 300px;
}
dialog::backdrop {
  background: rgba(0, 0, 0, 0.6);
  backdrop-filter: blur(3px); /* Ефект розмиття */
}
.controls {
  margin-top: 20px;
  display: flex;
  justify-content: flex-end;
  gap: 10px;
}
```

На відміну від , модальний <dialog> автоматично центрується браузером по центру екрана (використовуючи внутрішні стилі margin: auto).

Псевдоелемент ::backdrop - це віртуальний шар, який знаходиться між діалогом та рештою сторінки. Він існує лише тоді, коли діалог відкритий через showModal().

Тут ми застосовуємо backdrop-filter: blur(), що створює ефект матового скла, фокусуючи увагу користувача виключно на контенті вікна.

Коли ми викликаємо модальне вікно, воно виноситься у спеціальний внутрішній шар браузера, який ігнорує будь-які z-index батьківських елементів. Це гарантує, що жодне інше меню на сторінці не перекриє модалку.

JavaScript у даному випадку не будує інтерфейс, а лише перемикає режими його роботи через native API.

Метод showModal(): - це критично важливий метод. Він не просто робить елемент видимим, а активує режим інертності для решти сторінки. Браузер починає ігнорувати кліки та фокус клавіатури на всіх елементах, крім тих, що всередині <dialog>.

```
const dialog = document.getElementById('settingsDialog');
const openBtn = document.getElementById('openModal');
const cancelBtn = document.getElementById('cancelBtn');

openBtn.addEventListener('click', () => {
  dialog.showModal();
});

cancelBtn.addEventListener('click', () => {
  dialog.close();
});

dialog.addEventListener('close', () => {
  console.log("Результат діалогу:", dialog.returnValue);
});
```

Метод close(): Програмно закриває вікно. Ви можете передати в нього аргумент, наприклад dialog.close("cancel"), щоб вручну встановити результат повернення.

Властивість returnValue: це «пам'ять» діалогу. Якщо форма була закрита через submit, ця властивість автоматично набуває значення натиснутої кнопки. Це дозволяє уникати створення зайвих глобальних змінних для збереження вибору користувача.

Подія close: Ми підписуємося на неї, щоб виконати фінальну дію (наприклад, застосувати обрану тему інтерфейсу), коли анімація закриття завершена та дані отримані.

В результаті отримаємо (рис. 2):

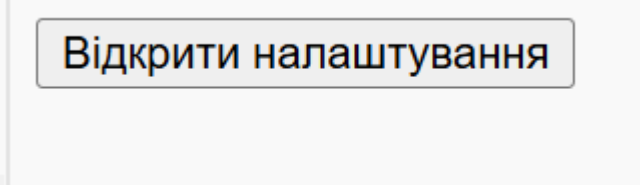


Рисунок 2. Стан «За замовчуванням»

Джерело рисунка: розробка автора.

Коли користувач заходить на сторінку, він бачить лише лаконічну кнопку «Відкрити налаштування». У цей момент браузер уже тримає в пам'яті структуру діалогового вікна, але завдяки вбудованій властивості display: none, тег <dialog> залишається «невидимкою». Він не займає жодного пікселя в макеті й, що критично важливо, повністю ігнорується скрінрідерами. Це гарантує, що допоміжні технології не перевантажують користувача зайвою інформацією завчасно.

Однак у мить натискання на кнопку запускається складний ланцюжок подій, що перетворює статичну сторінку на динамічний інтерфейс [94]. Ці процеси відбуваються паралельно:

1. Візуальна поява та позиціювання

Замість маніпуляцій з класами CSS, метод .showModal() миттєво переводить вікно в активний стан. Діалог автоматично центрується у вікні перегляду (viewport). Більше не потрібно вручну вираховувати координати x та y - браузер бере на себе роль ідеального декоратора, розміщуючи вікно у так званому Top Layer, який завжди знаходиться над іншими елементами, незалежно від їхнього z-index.

2. Створення «інертного» фону

Контент основної сторінки під діалогом миттєво стає фоновим. Використання модального режиму активує псевдоелемент ::backdrop, який візуально затінює сторінку. Але головна магія - у функціональному блокуванні: користувач не може натиснути жодне посилання або кнопку поза межами вікна; браузер фіксує основне положення сторінки, запобігаючи «scroll leakage» (прокручуванню фону колесом миші), що дозволяє зосередити всю увагу на налаштуваннях.

3. Фокус та доступність

Як тільки вікно відкривається, фокус автоматично переміщується всередину діалогу. Це створює так звану «пастку фокусу» (focus trap): клавіша Tab буде цикулювати лише між елементами всередині модального вікна. Це стандарт

безпеки та зручності, який дозволяє користувачу легко керувати налаштуваннями за допомогою клавіатури, не «гублячись» у заблокованих частинах сайту.

Візуальний акцент: завдяки CSS-правилу ::backdrop, основна сторінка затемнюється та розмивається, що створює глибокий фокус на формі налаштувань (рис. 3).

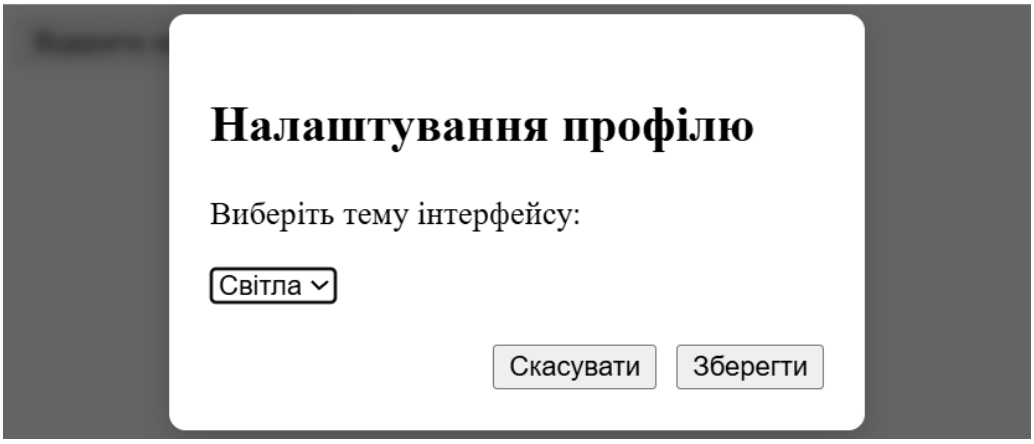


Рисунок 3. Момент відкриття (Виклик showModal())

Джерело рисунка: розробка автора.

Браузер автоматично переносить фокус на перший інтерактивний елемент у діалозі (наприклад, на список вибору теми). Користувач може перемикатися між полями лише всередині вікна. Якщо користувач передумав, він може просто натиснути клавішу Esc. Браузер самостійно закриє вікно, не потребуючи жодного додаткового коду на JavaScript.

Оскільки форма має method="dialog", результат натискання кнопок обробляється специфічним чином. Кнопка «Скасувати»: просто закриває вікно через виклик dialog.close(). Кнопка «Зберегти»: Оскільки вона має type="submit" та value="confirm", вікно закривається автоматично, а в консоль розробника виводиться підтвердження вибору. Після закриття діалогу фокус автоматично повертається на початкову кнопку «Відкрити налаштування», що дозволяє користувачеві продовжити роботу з того ж місця.

Отже, сучасна веб-розробка остаточно відійшла від концепції «маніпулювання сторінкою» на користь керування станом. При використанні React, Vue або Angular інтерфейс стає чистою проєкцією даних: UI = f(state).

Автоматизація синхронізації усуває цілий клас багів, пов’язаних із людським фактором. В імперативному підході розробник має пам'ятати про кожен , чи клас, який треба змінити при оновленні балансу користувача. У декларативному - просто змінити число в об'єкті стану, а реактивне ядро фреймворку (через Virtual DOM або Signal-системи) самостійно обчислює мінімально необхідні зміни в дереві елементів.

Програміст фокусується на бізнес-логіці («що має статися»), а не на технічній реалізації оновлення вузлів («як знайти елемент і змінити його колір»).

Хоча декларативність є пріоритетом для бізнес-логіки та маніпуляцій з масивами (методи filter, map, reduce), імперативний підхід залишається незамінним інструментом у специфічних нішах.

Коли йдеться про складну анімацію зі швидкістю, роботу з Canvas або обробку великих потоків даних у реальному часі, пряме керування ресурсами дозволяє уникнути надлишкових витрат ресурсів (пам'яті, процесорного часу, ваги файлів), які йдуть не на виконання конкретної задачі, а на підтримку роботи самого фреймворку.

Багато системних API браузера (наприклад, робота з фокусом, виділенням тексту або специфічними медіа-пристроями) за своєю природою є процедурними. Розуміння того, як створити декларативну обгортку над імперативним API, - це ознака архітектора високого рівня.

Сучасний UI - це багаторівнева система, де кожен шар має свій рівень пріоритетності та впливу на користувацький досвід:

- критичний шар (Modal/Dialog): перериває потік взаємодії для прийняття життєво важливих рішень. Це «блокуючий» шар, який гарантує фокус;

- контекстний шар (Popovers/Menus): надає доступ до вторинних функцій без втрати основного контексту;

- інформаційний шар (Tooltips/Hints): забезпечує атомарні підказки, що з’являються лише за запитом, не захаращуючи візуальний простір.

Така ієрархія створює передбачувану траєкторію уваги, де інтерфейс «спілкується» з користувачем, не викликаючи когнітивної перевтоми.

Проектування сьогодні виходить за межі естетики. Це створення відмовостійких систем зв'язку. Використання нативних тегів HTML5 - це не просто «правильний код», а забезпечення того, щоб машина (скрінрідери, пошукові роботи, ШІ-агенти) розуміла структуру даних.

Використання сучасних стандартів (наприклад, Popover API або Native Dialog) дозволяє зменшити обсяг JavaScript-коду, перекладаючи складні завдання на плечі браузера, що робить додатки швидшими та стабільнішими.

Імутабельні дані дозволяють легко реалізувати функції «Undo/Redo», відстежувати історію змін та синхронізувати стан між різними частинами програми чи навіть різними пристроями.

Поєднання нативних можливостей платформи з потужними абстракціями - це єдиний шлях до створення продуктів, які будуть актуальними не один сезон, а роками. Це шлях до інклюзивності, де програма однаково швидко працює і на флагманському смартфоні, і на бюджетному пристрої в умовах нестабільного інтерфейсу.

Сьогодні проектування інтерфейсу - це про ефективність каналу зв'язку між людиною та машиною. Використання таких інструментів, як реактивні компоненти, семантичні теги та стандартизовані API, перетворює розробку на процес створення відмовостійких систем, здатних працювати в реальному часі.

Сучасна архітектура користувацького інтерфейсу будується на ієрархії шарів, де кожен елемент має чітко визначену роль у керуванні увагою користувача.

Розуміння цієї ієрархії дозволяє створювати збалансовані інтерфейси, де критичні завдання вирішуються через діалоги, допоміжні функції - через поповери, а атомарні пояснення залишаються в підказках, забезпечуючи передбачувану та комфортну взаємодію.

Сучасний фахівець давно переріс роль ремісника, що просто транслює алгоритми в синтаксис; сьогодні він виступає архітектором складної цифрової взаємодії, де кожен рядок коду є частиною глобальної стратегії.

Використання нативних можливостей платформи HTML5 у поєднанні з принципами імутабельності та декларативності стає фундаментальним стандартом, а не просто технічним вибором. Такий підхід дозволяє створювати програмні продукти, де складність не стає перешкодою для продуктивності, а семантична чистота платформи забезпечує природну доступність та інклюзивність цифрового середовища за замовчуванням.

Інтеграція декларативного стилю мислення докорінно змінює процес управління станом системи, перетворюючи розробку на опис очікуваних результатів, а не нагромадження заплутаних інструкцій. Коли незмінність даних стає залізним правилом, система позбувається цілого класу помилок, пов’язаних із непередбачуваними побічними ефектами, що робить архітектуру прозорою та надзвичайно стійкою до деградації. Це створює надійний каркас, який легко піддається тестуванню та масштабуванню, дозволяючи продукту еволюціонувати разом із потребами бізнесу без втрати цілісності.

Зрештою, саме така синергія технологічної лаконічності та архітектурної дисципліни гарантує стабільно високу якість користувацького досвіду. Гнучкість системи до змін забезпечується не через збільшення кількості абстракцій, а через глибоке розуміння фундаментальних протоколів взаємодії. Це єдиний шлях до створення продуктів майбутнього, де навіть за умови надскладної внутрішньої логіки зовнішній інтерфейс залишається швидким, легким та адаптивним, забезпечуючи безшовну комунікацію між людиною та технологією.

DOI: 10.46299/ISG.2026.MONO.TECH.2.4.4

4.4 Моделі і технології інтелектуального моніторингу безпеки авіаційних соціотехнічних комплексів

ВСТУП

Актуальність теми. Сучасний етап розвитку цивільної авіації характеризується тотальною цифровізацією та інтеграцією розрізнених автоматизованих систем у єдиний інформаційно-технологічний простір. Авіаційні транспортні системи сьогодні є класичними авіаційними соціотехнічними комплексами (АСТК), де ефективність управління польотами та виробничими процесами залежить від синергії трьох компонентів: апаратного забезпечення (Hardware), програмного забезпечення (Software) та людського чинника (Lifeware) [95].

Проте стрімке впровадження новітніх інформаційних технологій (ІТ) породило специфічний клас ризиків. Інформаційні загрози еволюціонували від класичних деструктивних кібератак на програмний код до комплексних, гібридних інформаційних впливів на соціотехнічні контури АСТК [96]. Об'єктом таких впливів усе частіше стає операторський склад (пілоти, диспетчери, інженери), який приймає рішення на основі інформації, що надходить через автоматизовані робочі місця. Викривлення, затримка або умисна модифікація даних у процесі їхньої обробки та передачі можуть призвести до катастрофічних наслідків для безпеки польотів [97].

Більшість існуючих методів захисту зосереджені на суто технічних аспектах кібербезпеки (екранування, шифрування), ігноруючи динаміку поведінки людської ланки під дією інформаційних операцій. У зв'язку з цим виникає гостра науково-практична потреба у розробці нових інформаційних технологій та моделей, які дозволяють комплексно оцінювати стійкість АСТК до всього спектра інформаційних загроз, враховуючи специфіку соціотехнічної взаємодії.

Об’єкт дослідження: інформаційні процеси управління та забезпечення стійкості авіаційних соціотехнічних комплексів в умовах деструктивних інформаційних впливів.

Предмет дослідження: моделі, методи та інформаційні технології оцінювання стійкості інформаційних контурів АСТК до інформаційних загроз.

Мета дослідження: підвищення ефективності та безпеки функціонування авіаційних транспортних систем шляхом розробки та впровадження інформаційної технології комплексного оцінювання їхньої стійкості до інформаційних загроз на основі інтелектуального аналізу даних та математичного моделювання соціотехнічних процесів.

4.4.1 Системний аналіз авіаційних соціотехнічних комплексів як об'єктів інформаційної безпеки

4.4.1.1 Авіаційна соціотехнічна система як об’єкт деструктивного інформаційного впливу

Складні ергатичні, або соціотехнічні системи – це системи, обов'язковим елементом яких є людина-оператор із її знаннями, уміннями, психоемоційним станом, ціннісними орієнтаціями та ставленням до виконання службових обов'язків, яка безпосередньо взаємодіє з технічними пристроями у процесі досягнення цілей системи (рис. 1.1). Розрахунок параметрів таких систем здійснюється на основі фундаментальних наукових праць [95 - 98].

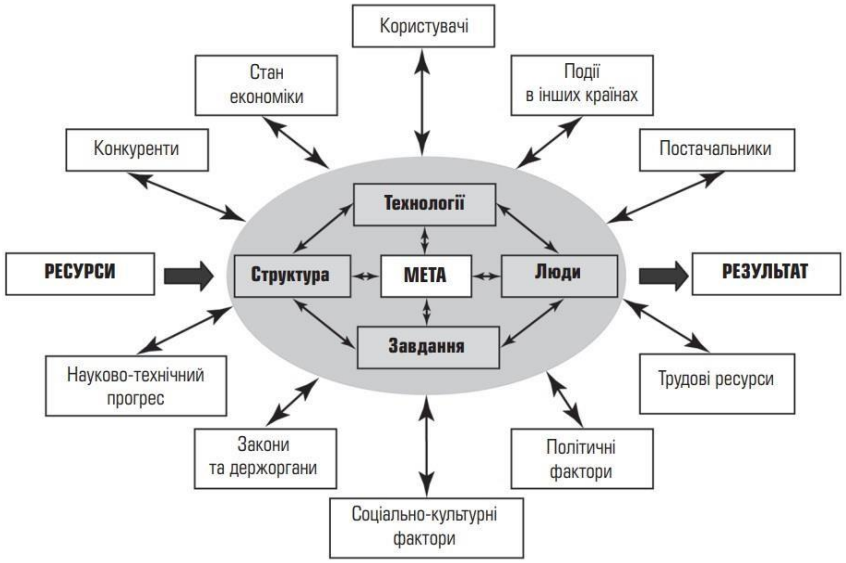


Рисунок 1.1. Загальна структура та чинники функціонування соціотехнічної системи [96]

Авіаційні транспортні системи сьогодні є класичними авіаційними соціотехнічними комплексами (АСТК), де ефективність управління польотами та виробничими процесами безпосередньо залежить від синергії трьох базових компонентів: апаратного забезпечення (Hardware), програмного забезпечення (Software) та людського чинника (Lifeware) [95].

На сучасному етапі розвитку глобальних комунікацій інформаційний простір став практично безмежним і всеохоплюючим. Ця властивість, з одного боку, забезпечує значні технологічні переваги, а з іншого – провокує виникнення нових вразливостей і деструктивних процесів, що порушують комплексну інформаційну безпеку. Водночас інформаційний простір функціонує в межах певних регуляторних та правових меж, обумовлених офіційними обмеженнями. Зазначені обмеження доцільно класифікувати на:

конвенціональні – такі, що зобов'язують суб'єктів дотримуватися комерційної, корпоративної чи персональної таємниці та забезпечують право людини на недоторканність приватного життя;

інституційні – пов'язані із захистом державної або військової таємниці на рівні відповідних інститутів та нормативно-правових актів.

Структура інформаційного простору визначається наявністю динамічних зв’язків між суб’єктами та об’єктами, на які ці суб’єкти здійснюють вплив. Суб’єкти та об’єкти з часом змінюють свої стани, утворюючи нові інформаційні контури й руйнуючи старі, що формує складну динаміку інформаційного простору. Такі зміни можуть відбуватися під дією деструктивних впливів (ДВ). З урахуванням присутності людини як ключового об’єкта захисту, деструктивні впливи доцільно поділити на:

інформаційно-кібернетичні впливи (ІКВ) – спрямовані на компрометацію, викривлення або знищення елементів технічної складової АСТК.

інформаційно-психологічні впливи (ІПВ) – метою яких є деструктивний вплив на соціальну (людську) складову АСТК.

Основна проблема полягає в тому, що структури в інформаційному просторі є фрагментарними, а зв'язки – локальними. Через це окремий суб'єкт інформаційного простору може навіть не підозрювати про існування іншого суб'єкта, який віддалений від нього архітектурно, але здатний здійснити на нього прихований вплив. Циркуляція інформації з обмеженим доступом (ІзОД), а також поширення спеціально створених маніпулятивних даних з метою реалізації деструктивного інформаційного тиску породжують системні проблеми, пов’язані із забезпеченням комплексної інформаційної безпеки соціотехнічних систем.

Для комплексного дослідження АСТК використовують методологію системного аналізу і синтезу. Аналіз і синтез є фундаментальними загальнонауковими методами пізнання, які діють у протилежних, але нерозривно пов’язаних напрямах: аналіз спрямований від цілого до його частин, тоді як синтез – від окремих частин до формування цілісного об’єкта. Системний аналіз дозволяє досліджувати такі властивості та відношення в об’єктах, які важко виявити під час розгляду окремих ізольованих елементів. Натомість об’єкт досліджується як цілеспрямована система через призму взаємозв’язків між її цілями та засобами їх реалізації.

Невід'ємною складовою системного аналізу є моделювання – процес, який охоплює вивчення реальної соціотехнічної системи, побудову її адекватної моделі (як засобу опису, пояснення та прогнозування поведінки системи-оригіналу) та перенесення отриманих знань на реальний об'єкт (рис. 1.2).

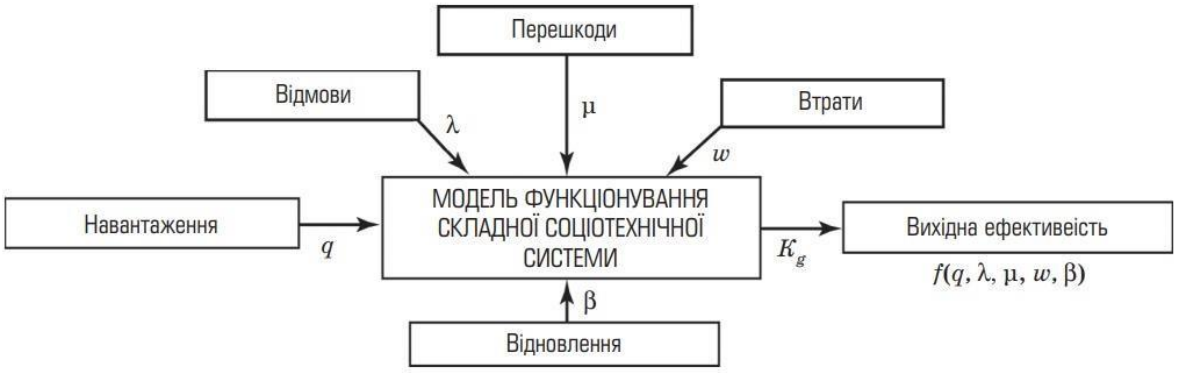


Рисунок 1.2. Процес моделювання складної соціотехнічної системи) [97]

Математична модель АСТК має створюватися на принципах функціонального об'єднання моделей окремих елементів і підсистем у єдиний комплекс програмно-реалізованих алгоритмів. Це забезпечує імітацію відповідних процесів для будь-яких вхідних умов і поточних станів системи, поєднуючи інформаційну та функціональну моделі авіаційної технічної системи (АТС). Параметри надійності та стійкості такої системи під час моделювання описуються такими показниками:

q – параметри потоку навантажень;

λ – параметри потоку відмов;

μ – параметри перешкод;

w – параметри втрат;

β – параметри відновлень;

K_g – коефіцієнт готовності [96]

Структурна складність СТС безпосередньо залежить від типів використовуваних елементів та специфіки зв'язків між ними (табл. 1.1).

Таблиця 1.1

Види структури СТС, елементи цих систем та зв'язки між ними

Вид структури	Елементи структури	Зв'язки між елементами структур
Функціональна	Функції, завдання, процедури	Інформаційні
Технічна	Пристрої, компоненти, комплекси	Лінії та канали звязку
Організаційна	Колективи людей і окремих виконавців	Інформаційні, супідрядності та взаємодії
Документальна	Документи	Взаємодії, вхідності та супідрядності
Алгоритмічна	Алгоритми	Інформаційні
Програмна	Програмні модулі й вироби	Керівні
Інформаційна	Форми існування та подання інформації в системі	Операційні перетворення інформації в системі

Для інформаційних об'єктів функціональної схеми будується інформаційна модель, що описує відношення між елементами у вигляді структур даних (їхнього складу та взаємозв'язків). У спрощеному вигляді побудова такої моделі передбачає три послідовні кроки:

Крок 1 – визначення типів сутностей;

Крок 2 – визначення типів зв'язків між сутностями;

Крок 3 – визначення ключових та неключових атрибутів сутностей, за якими розрізняються їхні екземпляри в межах кожного типу.

Сучасні СТС забезпечують широкий спектр форм подання інформації та методів її перетворення (зокрема, конвертацію з однієї форми в іншу залежно від характеру виконуваних завдань). Подання інформації можливе у вигляді формалізованих і неформалізованих текстів природною мовою, графічних зображень, реляційних відношень, аудіоповідомлень, відеопотоків тощо. Опис цих форм визначає характер взаємодії людини-оператора з інформаційною моделлю та структуру останньої.

Динамічна модель відбиває часові характеристики системи та послідовність взаємодії функцій у часі. Вона описує інформаційні процеси (динаміку функціонування), оперуючи такими поняттями, як стан системи, події, перехід із одного стану в інший, умови переходу та послідовність подій. Динамічну модель будують за чотири кроки:

Крок 1 – визначення технологічних операцій (дій), на які витрачається час;

Крок 2 – визначення черг, що виникають в очікуванні обслуговування;

Крок 3 – визначення когнітивних та технічних ресурсів, необхідних для виконання дій;

Крок 4 – задання статистичних параметрів моделі, її входів і виходів.

Зазначена сукупність моделей уможливлює комплексний опис як існуючої, так і проектованої соціотехнічної системи. У межах цієї моделі вектори атак на АСТК розділено на три базові класи:

технічні загрози: класичні кібератаки (DDoS, ін'єкції коду, маніпуляції з навігаційними даними – GPS-spoofing), спрямовані на порушення конфіденційності, цілісності та доступності даних в IT-мережах;

соціотехнічні загрози: використання методів соціальної інженерії (таргетований фішинг проти диспетчерського чи операторського складу, викрадення автентифікаційних даних легітимних користувачів через компрометацію їхніх особистих пристроїв).

Когнітивно-інформаційні загрози: цілеспрямоване впровадження дезінформації в канали комунікації або системи підтримки прийняття рішень (СППР) з метою змусити персонал прийняти хибне рішення (наприклад, неправдиве сповіщення загрози на борту або умисне викривлення метеоданих) [99].

Особливу небезпеку становлять комбіновані атаки, коли технічний збій (наприклад, штучне уповільнення роботи мережі) супроводжується інтенсивним когнітивним тиском на оператора. Це критично звужує час аналізу та стрімко збільшує ймовірність помилки під час керування польотами.

4.4.1.2 Проблема автоматизованого виявлення цілеспрямованих інформаційних атак на інформаційні ресурси АСТК і шляхи її вирішення

Виявлення цілеспрямованих атак з метою своєчасної протидії їм потребує оперативного аналізу інформаційного простору з використанням спеціалізованих систем моніторингу. Такі системи мають забезпечувати не тільки апаратний аналіз кіберинцидентів, а й кількісну оцінку динаміки їхніх проявів. У разі здійснення атаки інтенсивність потоку подій, яка являє собою часовий ряд кількості інформаційних інцидентів за певний проміжок часу (як правило, за добу), містить приховані закономірності як про сам факт атаки, так і про поточну фазу сценарію, за яким вона реалізується.

На сучасному етапі виявлення загроз інформаційній безпеці залишається переважно ручним процесом, де аналітики відстежують підозрілі події за допомогою розрізнених допоміжних інструментів. Здатність експертів розпізнавати аномальну активність та їхні повноваження щодо прийняття рішень ставлять людину на центральну роль у контурі безпеки. Проте надмірне покладання на обмежені людські можливості за умов дефіциту часу призводить до значної кількості пропущених загроз. Відтак, існує об'єктивна потреба у новій парадигмі виявлення, яка була б максимально автоматизованою, але дозволяла б аналітикам зберігати високу ситуативну обізнаність і контроль над процесом.

Управління технологічними процесами в авіації базується на використанні інформаційно-телекомунікаційних систем (ІТС), що об'єднують джерела інформації, канали її передавання, засоби оброблення, відображення та зберігання даних. У цих процесах людський фактор є критично важливим елементом. Попри постійний прогрес методів автоматизації, аналітики та особи, які приймають рішення, продовжують відігравати вирішальну роль у забезпеченні безпеки ІТС [98]. Класичний лінійний процес виявлення загроз починається зі спостереження за активністю в інформаційному просторі, яка згодом фільтрується набором інструментів захисту (рис. 1.3) [99].

Рисунок 1.3. Лінійний процес виявлення загроз [99]

Аналітик використовує результати роботи цих інструментів (оповіщення та зведення) для прийняття остаточного рішення про характер загрози та її можливий вплив на виконання місії. Однак специфіка сучасного кіберсередовища кидає серйозний виклик людському пізнанню: стрімке зростання обсягів мережевих даних, широке розмаїття джерел та раптові зміни архітектури мережі перевантажують оператора. На аналітика покладається надмірна відповідальність за прийняття якісних рішень. Як наслідок, традиційний підхід вимагає постійного підвищення кваліфікації персоналу. Проте основні когнітивні можливості людини (обсяг оперативної пам'яті, швидкість обробки інформації) не можуть масштабуватися відповідно до швидкості зростання мережевого трафіку, що перетворює людину на "вузьке місце" системи захисту. Крім того, адаптація моделей мислення аналітика відбувається значно повільніше, ніж поява нових модифікацій кібератак.

Останніми роками характер загроз зазнав докорінних змін, що підтверджується появою складних цілеспрямованих перманентних загроз – APT (Advanced Persistent Threats). Державні установи та провідні підприємства авіаційної галузі стають цілями високотехнологічних кампаній, які мають суттєві відмінності від класичних шкідливих програм. Головною особливістю APT є прагнення забезпечити приховане тривале закріплення в системі шляхом використання експлойтів "нульового дня" (zero-day) та мінімізації цифрового

сліду. Успішний захист від таких дій залежить від якості інформації, що надається аналітику, та його здатності оперативно її інтерпретувати.

Аналіз спеціалізованої наукової літератури [99]–[104] свідчить, що процес забезпечення безпеки інформації повинен мати комплексний характер і базуватися на глибокому логіко-евристичному аналізі можливих негативних наслідків. Такий підхід вимагає обов’язкової ідентифікації джерел загроз, уразливостей системи та визначення ступеня актуальності виявлених ризиків [100, 101]. Аналітики мають розкривати сутність спостережуваної поведінки мережі та прогнозувати подальші дії зловмисника [103, 104]. При цьому методичний інструментарію повинен залишатися гнучким, щоб забезпечувати можливість введення нових типів загроз та уразливостей без зміни базових алгоритмів.

Метою подальших розділів дослідження є обґрунтування та викладення структури адаптивного процесу виявлення загроз, у якому система моніторингу, автоматизований механізм аналізу та людина-аналітик функціонують у єдиному синергетичному контурі обміну інформацією. Ця архітектура дозволяє раціонально інвестувати обмежені когнітивні ресурси аналітика у вирішення найбільш критичних завдань.

Виявлення APT-атак апріорі вимагає залучення експертного потенціалу людини [105]. Протягом усього робочого процесу аналітик повинен спиратися на системи підтримки прийняття рішень (СППР), які фільтрують, агрегують та візуалізують дані. У випадках із високим рівнем достовірності діагностики системи автоматизації можуть самостійно ініціювати захисні дії. Проте в існуючих лінійних процесах (див. рис. 1.3) аналітик ізольований від етапів збору та первинної обробки доказів, що змушує його приймати рішення на основі жорстко нав'язаного ззовні інформаційного потоку. Брак прозорості та контролю заважає адаптації системи до нових атак.

На відміну від лінійних схем, запропонована синергетична концепція описує інтерактивний процес взаємодоповнення людини та автоматизації (рис. 1.4) [106].

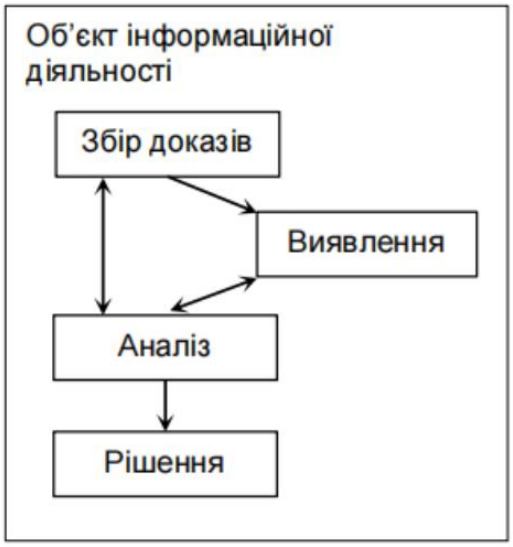


Рисунок 1.4. Синергетичний процес виявлення за концепцією "аналітика в циклі" [106]

Концепція базується на таких засадах:

диференціація рівнів залучення аналітика до процесу прийняття рішень;

динамічна адаптація механізмів виявлення відповідно до життєвого циклу загрози;

чітка типізація інтерфейсів взаємодії між компонентами системи, що дозволяє оперативно локалізувати загрози.

Для формування високого рівня довіри до рекомендацій СППР аналітик повинен мати доступ до логіки формування сповіщень. Він має взаємодіяти з базовими механізмами аналізу протягом усього циклу виявлення, а не лише на фінальній стадії [106]. Через таку взаємодію експерт здатний збагачувати систему унікальним контекстом, безперервно налаштовувати алгоритми під нові аномалії та задавати правила адаптації до змін у векторі атак.

Сучасні дослідження в галузі взаємодії людини з даними (HDI – Human-Data Interaction) пропонують людиноцентричний підхід до розробки динамічних потоків даних та автоматизованих міркувань [103]. Адаптація принципів HDI до сфери кібербезпеки дозволяє визначити аналітика як активний керуючий компонент системи. У межах нашої концепції виділено три фундаментальні аспекти HDI-взаємодії:

Зрозумілість (Comprehensibility): механізми збору та алгоритми аналітики повинні бути абсолютно прозорими й інтерпретованими для експерта.

Повноваження (Authority): аналітик володіє повним контролем над процесами збору даних та стратегіями управління ними.

Оборотність (Reversibility): можливість оператора змінювати методи обробки даних та переналаштовувати аналітику залежно від поточного контексту та цілей дослідження.

Увага людини є дефіцитним ресурсом, тому її слід спрямовувати на виконання найбільш критичних творчих завдань. Спираючись на класичну модель рівнів автоматизації за Sheridan & Verplank [107], виявлення атак розглядається як чотириетапний процес обробки інформації:

збір ознак (еквівалент збору інформації);

робота механізму виявлення (аналіз інформації);

прийняття рішення аналітиком (вибір рішення);

реагування (реалізація дій).

На кожному етапі участь людини може варіюватися від високої (низький рівень автоматизації, повний ручний контроль) до низької (автономна робота системи). Стосовно процесу винесення вердикту про наявність атаки виділено чотири рівні взаємодії:

Рівень 1 (Мінімальний): аналітик працює самостійно, шукаючи загрози у "сирих" неструктурованих логах.

Рівень 2 (Рекомендаційний): автоматизована система генерує сповіщення, а аналітик валідує їх та виявляє пропущені інциденти.

Рівень 3 (Контрольний): система працює автономно, роль аналітика зводиться до скасування хибних тривог (false positives) та точкового пошуку прихованих аномалій.

Рівень 4 (Максимальний): усі рутинні аспекти виявлення повністю автоматизовані, увага оператора цілком сфокусована на виборі найкращої стратегії ліквідації наслідків атаки.

Адаптація виявлення до життєвого циклу загрози базується на рівні її вивченості.

Фаза невідомої загрози (Zero-day): загроза націлена на невідому вразливість. Автоматичні сигнатурні методи тут безсилі. Виявлення вимагає глибокого експертного аналізу поведінкових аномалій, маркування активності та ручного збору доказів. Участь людини є максимальною.

Фаза накопичення знань: у міру фіксації прецедентів загрози інформація про неї накопичується, що дозволяє створювати перші евристичні правила й частково автоматизувати процес, знижуючи навантаження на аналітика. Це критично важливо, оскільки після публічного розкриття загрози інтенсивність атак із її використанням може зростати на 5 порядків [106].

Фаза відомої (типової) загрози: знання про атаку повністю формалізовані. Процес обробки даних та детекції передається автоматичним засобам (IDS/IPS). Людина лише затверджує фінальний протокол реагування, не залучаючись до рутинних обчислень.

Для реалізації описаного підходу запропоновано трикомпонентну синергетичну архітектуру взаємодії (рис. 1.5).

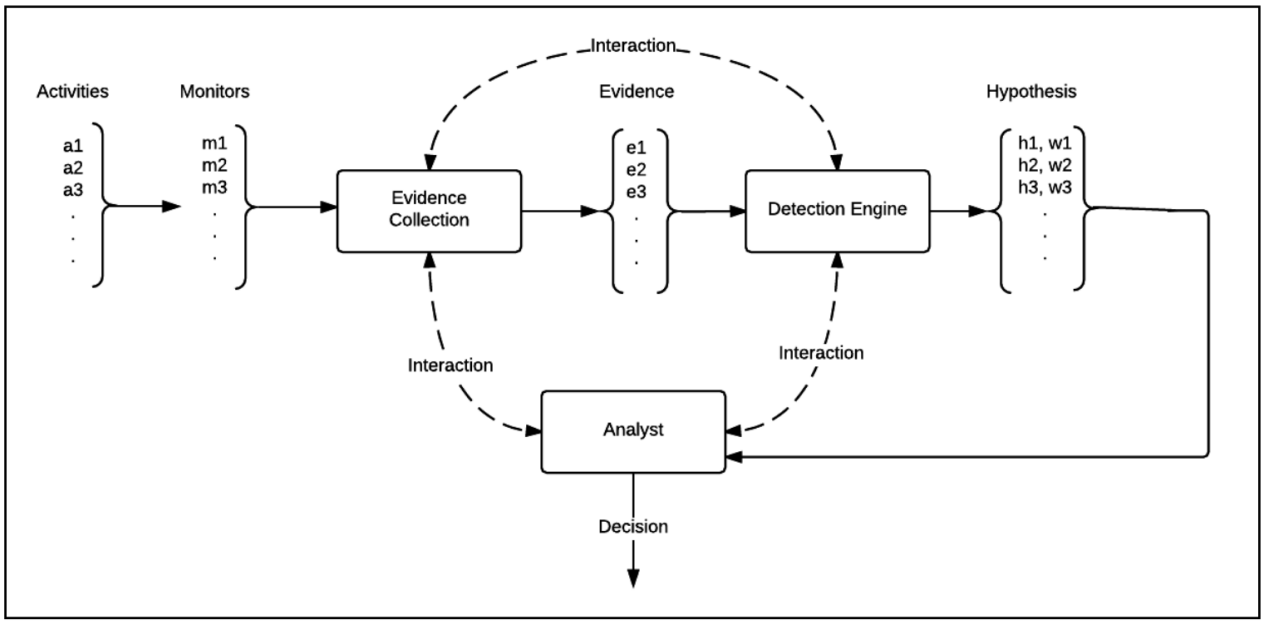


Рисунок 1.5. Трьохкомпонентна синергетична архітектура системи виявлення

Остаточне рішення приймається Людиною-аналітиком (**A**) за активної підтримки Механізму збору фактичних даних (**C**) та Механізму виявлення/висновків (**D**).

Функціональні обов'язки компонентів розподілені таким чином:

Компонент C (Collection): керує програмно-апаратними засобами моніторингу на різних рівнях архітектури мережі (глибокий аналіз пакетів – DPI на локальних вузлах або моніторинг системних процесів на хостах). Він трансформує сирий трафік у формалізовані факти-підтвердження. **C** володіє внутрішніми метриками про вартість збору, частоту оновлення даних, дисперсію та достовірність джерел, що дозволяє йому оптимізувати навантаження на канали зв'язку та сигналізувати аналітику про деградацію якості моніторингу.

Компонент D (Detection/Inference): високопродуктивне ядро системи, що обробляє масиви спостережень і розраховує ймовірності наявності конкретних загроз. Завдяки інтеграції методів керованого машинного навчання (Human-in-the-loop ML) [107], компонент **D** постійно донавчається на основі вердиктів людини, адаптуючись до динамічних змін кіберсередовища. Він транслює свою внутрішню математичну логіку у зрозумілий для аналітика формат, розкриваючи причинно-наслідкові зв'язки сформованих гіпотез.

Компонент **A** (Analyst): приймає остаточне рішення, спираючись на ймовірнісні оцінки від **D** та фактологічну базу від **C**, володіючи інструментами інтерактивного впливу на контури збору та аналізу даних.

Висновки до розділу 1

У першому розділі виконано всебічний аналіз сучасного стану проблеми забезпечення інформаційної безпеки авіаційних соціотехнічних комплексів в умовах цілеспрямованих деструктивних впливів, що дозволило отримати такі наукові та практичні результати:

1. Обґрунтовано, що сучасний авіаційний соціотехнічний комплекс є складним об'єктом деструктивного інформаційного впливу, де критичною вразливістю є контур взаємодії між автоматизованими системами та людським

оператором. Показано, що сучасні інформаційні атаки зміщують фокус із суто технічного руйнування інфраструктури на когнітивне маніпулювання даними з метою провокації операторського складу на прийняття помилкових рішень.

2. Встановлено, що традиційні автоматизовані системи виявлення кібератак та аномалій мають суттєві обмеження при захисті АСТК, оскільки вони функціонують відірвано від операційного контексту та психофізіологічного стану людської ланки, а також характеризуються високим рівнем хибних спрацьовувань та низькою інтерпретованістю результатів аналізу («ефект чорної скриньки»).

3. Запропоновано концептуальні засади синергетичного автоматизованого моніторингу на основі людино-центричного підходу (Human-in-the-Loop) та принципів взаємодії людини з даними (HDI). Визначено, що інтеграція когнітивних можливостей людини-аналітика та обчислювальної потужності автоматизованих алгоритмів на етапах збору, фільтрації та інтерпретації даних є ключовим фактором підвищення операційної стійкості авіаційних систем.

4.4.2 Математичне та концептуальне моделювання процесів забезпечення безпеки в авіаційних соціотехнічних системах

4.4.2.1 Концептуальна модель інтелектуального моніторингу комплексної безпеки авіаційних соціотехнічних систем

Авіаційна галузь є однією з найбільш технологічно насичених сфер діяльності, де безпека та ефективність функціонування залежать від взаємодії технічних систем, людського фактора, організаційних процесів та інформаційного середовища. Авіаційні соціотехнічні комплекси (АСТК) належать до класу складних емерджентних систем, що характеризуються високою розмірністю, багаторівневістю та надчутливістю до зовнішніх і внутрішніх деструктивних впливів.

Соціальна складова (людський фактор, або ланка Lifeware) у цій структурі є найбільш вразливим елементом. Інформаційні потоки, що циркулюють в

авіаційних СТС, безпосередньо формують інформаційне поле, на основі якого оператори (пілоти, диспетчери управління повітряним рухом, інженери) приймають критично важливі рішення. Таким чином, деструктивний інформаційний вплив (ДІВ) на інформаційне середовище авіапідприємства призводить до спотворення сприйняття реальної повітряної чи технологічної обстановки персоналом, що в умовах дефіциту часу загрожує аварійними ситуаціями або авіаційними катастрофами.

Концептуальна модель інтелектуального моніторингу АСТК є теоретичною основою для побудови інтегрованих моніторингових рішень, здатних забезпечувати проактивне виявлення загроз, оцінювання ризиків та підтримку прийняття рішень у реальному часі.

Модель ґрунтується на системному підході, який розглядає АСТК як багаторівневу структуру, де технічні, людські, організаційні та інформаційні компоненти взаємодіють у єдиному операційному середовищі.

Основна ідея моделі полягає у створенні безперервного циклу інтелектуального аналізу, що забезпечує перехід від реактивного до проактивного управління безпекою. Такий підхід дає змог не лише фіксувати події, а й розуміти їхні причини, прогнозувати можливі сценарії розвитку та формувати рекомендації для операторів.

Сформуємо загальну математичну модель, що описує динаміку зміни рівня безпеки системи [109] в контексті інтелектуального моніторингу.

Архітектура концептуальної моделі включає п’ять ключових підсистем, кожна з яких виконує специфічні функції та взаємодіє з іншими елементами системи:

1. Підсистема збору та інтеграції даних
2. Підсистема інтелектуальної обробки та аналізу
3. Підсистема оцінки ситуації та ризиків
4. Підсистема прогнозування сценаріїв
5. Підсистема підтримки прийняття рішень

Кожна підсистема має власні методи, алгоритми та інформаційні потоки, але всі вони інтегровані у єдину аналітичну платформу, що забезпечує цілісність моніторингу.

З огляду на викладене рівень безпеки системи можна характеризувати за такою матрицею:

$$B=\begin{pmatrix} K_1 & F_1 & V_1 & T_1 & S_1 \\ K_2 & F_2 & V_2 & T_2 & S_2 \\ K_3 & F_3 & V_3 & T_3 & S_3 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ K_n & F_n & V_n & T_n & S_n \end{pmatrix}, \quad (2.1)$$

де:

K_i – показник рівня безпеки за і-м критерієм, $i=\overline{1, n}$;

F_i – тенденція зміни і-го критерію (зростає, зменшується, нейтральний), $i=\overline{1, n}$;

V_i – швидкість зміни і-го критерію (наприклад: низька, нижче середнього, середня, вище середнього, висока), $i=\overline{1, n}$;

T_i – час для і-го критерію, який дає змогу правильно інтерпретувати значення параметра V_i, $i=\overline{1, n}$;

S_i – ступінь критичності негативних наслідків при реалізації ризиків, який погіршує значення і-го критерію, $i=\overline{1, n}$.

Матрицю вигляду B в подальшому будемо називати матрицею безпеки (МБ).

Критерії в матриці безпеки можна згрупувати за відповідними напрямами забезпечення безпеки.

Таким чином, кожний кортеж $(K_i, F_i, V_i, T_i, S_i)$ характеризує стан безпеки за i-м критерієм.

Часткові матриці, що складаються з рядків і визначають певний напрям забезпечення безпеки, в свою чергу, описують стан у відповідній галузі.

Показники рівня безпеки K_i тісно пов’язані з наслідками від можливої реалізації наявних у системі загроз та заходами, які спрямовані на запобігання, локалізацію й усунення таких наслідків [109].

Рівень безпеки АСТК не є сталим – це динамічний показник. Матриця безпеки будується на основі лінгвістичних змінних.

Первинні загрози. Це безпосередні чинники впливу (кібератаки, відмова обладнання, помилка пілота).

Вторинні загрози. Це наслідки взаємодії первинних загроз. Наприклад, технічна затримка в мережі (первинна) спричиняє стрес у диспетчера (вторинна), що веде до "соціотехнічної відмови".

Імовірність виникнення вторинних загроз є умовною і залежить від стану системи та стану зовнішнього середовища.

Зокрема, деякі стани системи можуть спровокувати виникнення загроз, поява яких в інших умовах була б неможливою.

Введемо такі позначення:

$\overline{UG_i}$ *і* $\widetilde{UG_j}$ *(i, j = 1, 2, 3, ...)* – сукупність первинних і вторинних загроз, що виникають з імовірностями $\overline{PUG_i}$ і $\widetilde{PUG_j}$, відповідно, здійснюючи вплив $\overline{n_{km}}$ і $\tilde{n}_{km}$ на елемент (*k, m*) матриці безпеки *B* (*k = 1, 2, 3, ... ; m = 1, 2, 3, 4, 5*).

Вплив кожної з первинних або вторинних загроз можна описати матрицею впливу, що має вигляд [109]:

$$N_i = \begin{pmatrix} n_{11} & n_{12} & n_{13} & n_{14} & n_{15} \\ n_{21} & n_{22} & n_{23} & n_{24} & n_{25} \\ \dots & \dots & \dots & \dots & \dots \\ n_{n1} & n_{n2} & n_{n3} & n_{n4} & n_{n5} \end{pmatrix}. \qquad (2.2)$$

Фактично матриця впливу являє собою матрицю ваги впливу *i*-го негативного фактору на елементи МБ.

Необхідно відмітити, що вплив $\overline{n}_{km}$ і $\tilde{n}_{km}$ на деякі елементи матриці безпеки *B* може бути як негативним, так і позитивним.

Елементи матриці, що негативно впливають, є негативними щодо елементів МБ, елементи, що позитивно впливають – є позитивними щодо елементів МБ, елементи, що ніяк не впливають, є нейтральними.

Кортеж $\overline{R}i = \left(\overline{N_i}\,;\, \overline{PUG_i}\right)$ є ризиком реалізації *i*-ї первинної загрози.

Цей кортеж відображає появу наслідків з імовірністю $\overline{PUG_i}$, які змінюють стан системи через відповідні матриці впливу $\overline{N_i}$.

Імовірності виникнення первинних загроз $\overline{PUG_i}$ є незалежними величинами. Однак, сукупність превентивних заходів захисту дає змогу послабити вплив первинних загроз на ступінь комплексної безпеки системи.

Цей факт може бути описаний за допомогою матриці превентивних заходів забезпечення стійкості:

$$Z_j = \begin{pmatrix} z_{11} & z_{12} & z_{13} & z_{14} & z_{15} \\ z_{21} & z_{22} & z_{23} & z_{24} & z_{25} \\ \dots & \dots & \dots & \dots & \dots \\ z_{n1} & z_{n2} & z_{n3} & z_{n4} & z_{n5} \end{pmatrix}, \quad (2.3)$$

де $j = \overline{1,\mathrm{M}}$, M – загальна кількість превентивних заходів.

Елементи матриці Z_i є демпферними коефіцієнтами (коефіцієнтами запасу стійкості).

Тоді під залишковим впливом будемо мати на увазі матрицю алишкового впливу *N,* елементи якої знаходяться з виразу:

$$\hat{n}_{mn} = n_{mn} \otimes \max_{k=1\dots\mathrm{M}.} z_{mn}^{k}, \quad (2.4)$$

де z_{mn}^{k} – елемент (*m, n*) матриці коефіцієнтів запасу стійкості Z_k. Символом "$\otimes$" позначена у такий спосіб певна для двох матриць операція.

У випадку числових значень елементів матриць, це може бути, наприклад, операція простого поелементного множення або додавання. У випадку лінгвістичних значень, ця операція (в нечіткій математиці –композиція) визначається за допомогою принципу розширення звичайних (чітких) математичних функцій на нечіткі числа, запропонованого Л.Заде [109] .

Під залишковим ризиком розуміється кортеж

$$\overline{R}i = \left(\overline{N_i}\,;\, \overline{PUG_i}\right) \quad (2.5)$$

Якщо, незважаючи на превентивні заходи забезпечення стійкості, реалізація визначеної множини первинних загроз призвела до появи наслідків, то необхідно розпочати заходи для їх локалізації та усунення [110].

Насамперед, необхідно оцінити відхилення поточного стану системи B від безпечного стану B_s.

Уведемо поняття різниці між двома матрицями, визначивши результат застосування операції "⋕" до двох елементів матриць аналогічно тому, як це було зроблено для операції "⊗" : у випадку числових значень елементів матриць – це операція поелементного вирахування, у випадку лінгвістичних значень, операція визначається за допомогою принципу розширення Л. Заде.

Тоді матриця $Q = B_s \# \widehat{B}$ є матрицею втрат безпеки на цьому етапі.

Матриця втрат безпеки Q являє собою вхідні дані для блоку ліквідації наслідків (БЛН).

Реалізація заходів цього блоку може бути формалізована за допомогою матриці ліквідації наслідків:

$$L = \begin{pmatrix} l_{11} & l_{12} & l_{13} & l_{14} & l_{15} \\ l_{21} & l_{22} & l_{23} & l_{24} & l_{25} \\ \dots & \dots & \dots & \dots & \dots \\ l_{n1} & l_{n2} & l_{n3} & l_{n4} & l_{n5} \end{pmatrix}. \quad (2.6)$$

Результат застосування БЛН може бути записаний у такому вигляді:

$$\widehat{Q} = Q \otimes L = \begin{pmatrix} \hat{q}_{11} & \hat{q}_{12} & \hat{q}_{13} & \hat{q}_{14} & \hat{q}_{15} \\ \hat{q}_{21} & \hat{q}_{22} & \hat{q}_{23} & \hat{q}_{24} & \hat{q}_{25} \\ \dots & \dots & \dots & \dots & \dots \\ \hat{q}_{n1} & \hat{q}_{n2} & \hat{q}_{n3} & \hat{q}_{n4} & \hat{q}_{n5} \end{pmatrix}. \quad (2.7)$$

Матрицю $\widehat{Q}$ назвемо матрицею залишкових втрат безпеки.

Якщо $\widehat{Q} \neq B_s$, то такий стан системи може ініціювати появу вторинних загроз з імовірностями $\overline{PUG_i}$.

Таким чином, крім первинних загроз, залежно від поточного стану системи і її оточення, можуть виникнути вторинні загрози, ймовірність появи яких дорівнює $\overline{PUG_i}$.

Кортеж $\overline{R}i = (\overline{N_i}; \overline{PUG_i})$ є ризиком реалізації *i*-ої вторинної загрози.

Зауважимо, що ймовірності виникнення вторинних загроз не є безумовними, як для первинних загроз. Вони залежать від поточного стану системи. З первинними загрозами починають боротися ще до їхнього настання, тобто фактично намагаються звести до мінімуму їхні наслідки, не маючи можливості вплинути на сам факт їх виникнення. У випадку з вторинними загрозами, намагаються взагалі не допустити їх, тобто нейтралізувати їх причини. Це принципова розбіжність у блоках заходів, вплив яких формалізовано множиною матриць Z_j і матрицею L.

Для побудови ефективної системи безпеки АСТК визначаються такі її основні показники:

Рівень стійкості соціотехнічного контуру ($\{\boldsymbol{R_{st}}\}$): здатність людської ланки правильно інтерпретувати інформацію та приймати адекватні рішення в умовах завад або цілеспрямованої дезінформації.

Час виявлення деструктивного впливу ($\{\boldsymbol{T_{det}}\}$): інтервал часу від моменту проникнення загрози в інформаційне середовище до моменту її ідентифікації автоматизованою системою чи оператором.

Когнітивне навантаження на аналітика ($\{\boldsymbol{W_{cog}}\}$): показник, що визначає ступінь залучення та перевантаження людського оператора під час обробки інформаційних повідомлень та сигналів тривоги.

Запропонована концептуальна модель інтелектуального моніторингу комплексної безпеки передбачає безперервний збір даних про стан технічних компонентів (мережевий трафік, цілісність баз даних, телеметрія) та одночасний аналіз поведінкових і когнітивних маркерів людського фактора. На відміну від класичних систем виявлення вторгнень, дана модель інтегрує інформаційні потоки обох контурів, що дозволяє виявляти не просто технічні аномалії, а комплексні соціотехнічні атаки, спрямовані на маніпулювання свідомістю та рішеннями операторів АСТК.

4.4.2.2 Математична модель аналізу уразливостей соціотехнічних систем до впливів соціальної інженерії на основі нечітких направлених соціальних графів

Забезпечення стійкості АСТК вимагає детального аналізу уразливостей користувачів стосовно форм маніпулювання їхньою свідомістю через методи соціальної інженерії (фішинг, претекстинг, вішинг тощо). Для формалізації цієї складної і багатофакторної проблеми запропоновано використання математичного апарату нечітких направлених соціальних графів.

Нечіткий направлений соціальний граф задається у вигляді:

$$G = (V, E) \quad (2.8)$$

де V - множина суб'єктів, Е - множина дуг між парами суб'єктів.

Однак, використання такого підходу на практиці обмежується складністю чітких міркувань стосовно поведінки як соціального інженера, так і користувача соціотехнічної системи [111]. Це обумовлено, по-перше, різноманітністю існування і реалізації способів впливу соціальної інженерії. По-друге, неоднозначністю поведінки користувача соціотехнічної системи залежно від маніпулятивної форми впливання на нього з боку соціального інженера.

Для врахування і, як наслідок подолання цих обмежень використано підхід, що запропоновано в [111]. За основу взято твердження про нечіткість мислення людини. Це означає, що його елементи належать до нечітких класів об'єктів. При цьому перехід від належності до неналежності класу безперервний. Ступінь такого переходу визначається функцією належності. Тому для врахування цих особливостей використовується логіка з нечіткою істинністю, нечіткими відношенням, нечіткими правилами виведення. Застосування нечіткої логіки обумовлюється здатністю до обирання важливої інформації залежно від проблемної ситуації.

З огляду на це аналізування уразливостей соціотехнічних систем до впливів соціальної інженерії відображається нечітким направленим соціальним

графом [112], [113]. Його використання дозволяє врахувати нечіткість поведінки як соціального інженера, так і користувача соціотехнічної системи.

Розглянемо множину впорядкованих пар елементів множини V, що визначаються декартовим добутком

$$V \times V = \{(v_i, v_j) \mid v_i \in V \cap v_j \in V\},$$

$$i = 1, n; \; j = 1, n. \qquad (2.9)$$

Тоді модель аналізування уразливостей соціотехнічних систем до впливів соціальної інженерії задається нечітким направленим соціальним графом G [112], [113], [114]. Під ним розуміється підмножина множини декартового добутку (2.2)

$$\forall (v_i, v_j) \in V \times V : \mu_G(v_i, v_j) \in M$$

$$G = \{((v_i, v_j) \mid \mu_G(v_i, v_j))\}, \; G \subseteq V \times V \qquad (2.10)$$

де M - множина належностей (v_i, v_j) множини $V \times V$, M = [94,95];

$\mu_G(v_i, v_j)$ – функція належностей (v_i, v_j) множини $V \times V$.

Кожна впорядкована пара тлумачиться дугою і нею відображається направленість від і до j суб'єкта (наприклад, (v_1, v_2) – від суб'єкта v_1 (відправника) до суб'єкта v_2 (отримувача) як маніпулятивної форми). Така направленість може бути або відсутньою, або асиметричною.

Крім цього направленість між суб'єктами враховується шляхом визначення числа вхідних та вихідних дуг між ними, а саме [115]:

$$d_I(v_i), (v_j, v_i) \quad v_j \in V$$

$$d_O(v_i), (v_j, v_i) \quad v_j \in V \qquad (2.11)$$

Кількості вхідних і вихідних дуг використовуються для визначення різновиду суб'єкта при аналізуванні вразливостей соціотехнічних систем до впливів соціальної інженерії:

ізольований, $d_I(v_i) = 0$, $d_O(v_i) = 0$. Наприклад, соціальний інженер не застосовує маніпулятивну форму v_i;

відправник, $d_I(v_i) = d_O(v_i) > 0$. Наприклад, соціальний інженер (суб'єкт v_i) впливає через маніпулятивні форми на користувача соціотехнічної системи;

отримувач, $d_I(v_i) > d_O(v_i) = 0$. Наприклад, на користувача (суб'єкт v_i) соціотехнічної системи через його вразливості впливає соціальний інженер.

Важливість суб'єктів виявляється завдяки використанню характеристик центральності та пріоритету [111], [115].

Серед них найбільш активні суб'єкти характеризуються центральністю, наприклад: маніпулятивна форма, уразливість користувача соціотехнічної системи. Вона визначається ступенем центральності і близькістю. Ступінь центральності знаходиться за такою рівністю

$$C_D(v_i) = \frac{\sum_{j=1}^{n} \mu_G(v_i, v_j)}{n\text{-}1} \quad (2.12).$$

Тоді як близькість між двома суб'єктами встановлюється з огляду на відстань між ними

$$C_D(v_i) = \frac{J_i / n\text{-}1}{\sum_{j=1}^{n} d(v_i, v_j) / J_i}, \quad (2.13)$$

де J_i – кількість суб'єктів у межах впливу суб'єкта v., наприклад, користувачів соціотехнічної системи, $d(v_i, v_j)$ – відстань між суб'єктами vi та vj, наприклад, між соціальним інженером і користувачем соціотехнічної системи.

У даному випадку зосереджуються на виборі суб'єктів. Однак, поза їх увагою залишається обумовленість таких активностей як відправник або як отримувач. Дана особливість враховується характеристикою пріоритетності. Тому нею відображаються, наприклад, або вразливості користувача, або користувач соціотехнічної системи. Це означає, що кожен з них тлумачиться як суб'єкт отримувач. За аналогією з центральністю для визначення пріоритету суб'єктів використовується ступінь центральності і близькість.

Ступенем пріоритету характеризується незалежність кожного з суб'єктів. Це означає, що пріоритетні суб'єкти порівно з іншими отримують більше можливостей, ніж іншими. Його ступінь визначається

$$P_D(v_i) = \frac{d_I(v_i)}{n\text{-}1}$$

де $d_I(v_i)$ – кількість суб'єктів, що пов'язані з суб'єктом v_i, наприклад, кількість маніпулятивних форм соціального інженера, що реалізуються через уразливість користувача соціотехнічної системи.

Близькістю пріоритету характеризується кількість суб'єктів, що безпосередньо (уразливість – користувач соціотехнічної системи) або опосередковано (маніпулятивна форма – уразливість – користувач соціотехнічної системи) пов'язані з суб'єктом v_j .

$$P_p(v_i)=\frac{I_i/_{n-1}}{\sum_{j=1}^{n} d(v_j,v_i)/I_i} \qquad (2.14)$$

де I_i – кількість суб'єктів у межах впливу суб'єкта v_i .

Використання (2.7) дає змогу аналізувати вразливості соціотехнічних систем на рівні суб'єкта, пари суб'єктів (діада), трійка суб'єктів (тріада) [111]. Для цього кожен з рівнів відображається підграфом G_s

$$G_s \subseteq G.$$

Діада задається підграфом, що має дві вершини та дугу між ними. При цьому впорядкована пара вершин може знаходиться у одному з двох станів і відображає, наприклад:

вплив соціального інженера (суб'єкт v_1) на користувача соціотехнічної системи (суб'єкт v_3)

$$G_s=\{((v_1, v_3)|\mu_G(v_1, v_3))\};$$

відсутність впливу соціального інженера (суб'єкт v_1) на користувача соціотехнічної системи (суб'єкт v3)

$$G_s=\{((v_1, v_3)|0)\}.$$

Тріада представляється підграфом, що має три вершини та дуги між ними. Може перебувати в одному в декількох станах і відображає, наприклад:

безпосередній або опосередкований вплив соціального інженера (суб'єкт v_1) без або з урахуванням обману як маніпулятивної форми (суб'єкт v_1) на користувача соціальної мережі (суб'єкт v_3)

$$G_s=\{((v_1, v_2)|\mu_G(v_1, v_2),((v_1, v_3)|\mu_G(v_1, v_3)), (v_2, v_3)|\mu_G(v_2, v_3))\}$$

опосередкований вплив соціального інженера (суб'єкт v_1) з урахуванням обману як маніпулятивної форми (суб'єкт v_2) на користувача соціальної мережі (суб'єкт v_3)

$$G_s=\{((v_1, v_2)|\mu_G(v_1, v_2),((v_1, v_3)|0), (v_2, v_3)|\mu_G(v_2, v_3))\};$$

безпосередній вплив соціального інженера (суб'єкт v_1) на користувача соціальної мережі (суб'єкт v_3)

$$G_s=\{((v_1, v_2)|\mu_G(v_1, v_2),((v_1, v_3)|\mu_G(v_1, v_3)), (v_2, v_3)|0))\}.$$

Отже, задання моделі аналізування уразливостей соціотехнних систем нечітким направленим соціальним графом надало змогу встановити особливості впливання соціальної інженерії. Для цього визначено множину суб'єктів (соціальний інженер, маніпулятивна форма, уразливість, користувач) та впорядковані пари її елементів з функціями їх належності. Виокремлено різновиди суб'єктів з боку соціального інженера, користувача, маніпулятивної форми, уразливості за числами вхідних і вихідних дуг. Тоді як важливість кожного з них отримано за допомогою характеристик центральності та пріоритету. Крім цього, встановлено рівні суб'єкта, діади, тріади для аналізування уразливостей соціотехнічних систем до впливів соціальної інженерії. Це уможливить визначати способи таких впливів з урахуванням особливостей їх реалізації через уразливості користувачів і, як наслідок, протидіяти їм.

Таке представлення дозволяє детально врахувати особливості впливу соціальної інженерії на персонал авіапідприємства. Важливість та критичність кожної вершини та дуги у графі визначається через розрахунок характеристик центральності та пріоритету:

Виокремлення цих рівнів дає змогу розробляти точкові, адаптивні способи протидії ДІВ з урахуванням специфіки їх реалізації через уразливості користувачів.

4.4.2.3 Математичні моделі оцінювання потужності деструктивного інформаційного впливу в АСТК

Для кількісного та якісного прогнозування наслідків ДІВ на безпеку авіаційних процесів розроблено дві взаємодоповнюючі математичні моделі. Вони дають змогу оцінити потужність інформаційного впливу з урахуванням специфіки його джерела, каналів поширення та механізмів реалізації.

4.4.2.3.1 Перша математична модель (Оцінка потенціалу та проникності ДІВ)

Модель базується на аналізі параметрів джерела впливу та динаміки його сприйняття соціотехнічною системою. Потужність впливу W_{DIV}^{I} розглядається як функція від трьох інтегральних векторів:

$$W_{DIV}^{I} = f(I_{src}, K_{ch}, S_{rec})$$

де:

I_{src} – вектор інтенсивності та релевантності деструктивної інформації, що генерується джерелом (включає рівень маскування під легітимні повідомлення);

K_{ch} – коефіцієнт проникності каналу передачі даних (враховує наявність технічних фільтрів, засобів захисту та швидкість поширення інформації);

S_{rec} – коефіцієнт сприйнятливості (абсорбції) інформації реципієнтом (людською ланкою), що залежить від поточного рівня когнітивного навантаження та психоемоційного стану оператора.

Для ідентифікації ймовірних джерел впливу та об’єктно-суб’єктних моделей взаємодії введемо такі позначення:

V' – різні джерела впливу, які використовують відповідні механізми застосування інформаційного впливу;

D_m – об’єкти впливу, які доцільно розрізняти за різними критеріями, наприклад, соціальні об’єкти, технічні об’єкти тощо;

Y_m – кількість об’єктів, які змінили свій стан під впливом інформаційного

впливу P_j.

Враховуючи принцип адитивності, можна запропонувати таке співвідношення для оцінювання ефективності інформаційного впливу:

$$Y_m(D_m) = \frac{Y_m(P_j)}{|D_m|} \tag{2.15}$$

де $|D_m|$ – потужність множини D_m .

З урахуванням того, що одне джерело впливу може одночасно застосовувати декілька механізмів реалізації інформаційного впливу відносно одного об'єкта впливу, використовуючи такі дії, як: дискредитація, агітація, недобросовісна конкуренція тощо. Аналітичний вираз для оцінювання потужності інформаційного впливу можна представити як узагальнену математичну модель, що враховує вищеозначені особливості впливу

$$\mathrm{V_i^j(D_m)} = \sum_{\mathrm{i,j,m=1}}^{\mathrm{n}} \frac{Y_m(P_j)}{|D_m|} k_t \tag{2.16}$$

де V_j показує приналежність джерела впливу до і-го класу, яке використовує *j-й* механізм реалізації інформаційних операцій;

$Y\ (P_j)$ – кількість об'єктів *m-го* класу, які змінили свій стан під дією *j*-го механізму впливу;

k_t – коефіцієнт, який враховує частоту звернення до даного джерела впливу *i*; змінюється від 0 до 1;

n – відповідна кількість класів та механізмів впливу.

Очевидно, що ефективність інформаційного впливу визначається кількістю об'єктів, що змінили свій стан у тому напрямку, який необхідно для об'єкта впливу. Аналіз виразу (2.9) дає можливість зробити попередній аналіз ризиків відносно ефективності джерел та відповідних механізмів проведення спеціальних інформаційних операцій, які використовує об'єкт впливу і проведення попереднього ранжування відносно потужності ймовірних інформаційних спеціальних операцій. Але для прийняття рішення щодо побудови ефективного комплексного захисту від проведення спеціальних

інформаційних операцій потрібно визначення найменш захищених місць і елементів, причин і ймовірних наслідків проведення спеціальних інформаційних операцій.

Аналіз цієї моделі дозволяє обчислити граничні значення, за яких деструктивна інформація долає бар'єри захисту та починає безпосередньо впливати на контур управління польотами.

4.4.2.3.2 Друга математична модель (Оцінка емерджентних наслідків та операційного ризику)

Друга модель орієнтована на оцінювання кумулятивного ефекту від ДІВ та розраховує ступінь деградації функцій АСТК. Модель використовує апарат марковських процесів або байєсівських мереж довіри (залежно від обраної деталізації) для моделювання переходів системи між станами «Нормальне функціонування», «Прихована компрометація», «Аварійний стан через хибні дії оператора».

Вихідним показником моделі є індекс операційного ризику $Risk_{ops}$, який враховує ймовірність прийняття оператором помилкового рішення під дією ДІВ та потенційні збитки (затримки рейсів, пошкодження майна, загроза життю).

У практичній площині результати моделювання дозволяють нам вирішити такі задачі:

1. Визначити найбільш ефективні або небезпечні джерела проведення інформаційних операцій, а також використовуваних механізмів з боку об’єкта впливу;
2. Визначити причинно-наслідковий зв'язок ефективності проведення інформаційних операцій з боку об’єкта впливу;
3. Запропонувати комплексний захист у вигляді проведення спеціальних контроперацій.

Розроблені моделі і вирішення наведених задач дозволяє запропонувати методику для прийняття управлінських рішень щодо керування інформаційною

безпекою на об’єкті захисту в умовах інформаційної війни. Доцільно розглядати процес управління інформаційною безпекою як неперервний і багаторівневий, тому суть методики має враховувати принципи неперервності і багаторівневості. Для ефективної інформаційної протидії необхідно побудувати відповідну ефективну організацію, яка б структурно і функціонально відповідала і вирішувала вище наведені задачі/

Узагальнення результатів обох моделей дає змогу не лише констатувати факт атаки, а й формулювати обґрунтовані управлінські рішення для служб авіаційної безпеки та IT-підрозділів авіапідприємств, спрямовані на оперативне проведення контрзаходів та зниження потужності деструктивного впливу до безпечного рівня.

4.4.2.4 Математична модель оцінювання нелінійної стійкості соціотехнічного контуру АСТК

Деструктивні інформаційні впливи та маніпуляції в авіаційних системах мають виражений нелінійний характер. Психофізіологічна реакція операторів (пілотів, диспетчерів) на дезінформацію або стрес, а також поширення похибок в автоматизованих контурах відбуваються за нелінійними законами, що характеризуються ефектами накопичення «критичної маси» та раптового зриву управління.

Математична модель оцінювання нелінійної стійкості базується на системі диференціальних рівнянь, що описують динаміку зміни стану стійкості контуру під дією зовнішнього тиску:

$$\frac{dx}{dt} = \varphi(x) - \psi(W_{DIV}, W_{cog})$$

де:

x– поточний рівень стійкості соціотехнічного контуру;

$\varphi(x)$ – функція внутрішнього самовідновлення та адаптації системи (за рахунок досвіду оператора та вбудованих інструментів перевірки помилок);

$\psi(W_{DIV}, W_{cog})$ – нелінійна функція деградації контуру під спільною дією потужності атаки та когнітивного перевантаження.

Першим кроком при дослідженні наведеної вище системи диференціальних рівнянь є визначення стаціонарних точок, тобто розв'язок системи рівнянь:

$$f_i\{X,a\}=0 \tag{2.17}$$

Другим кроком досліджень є визначення характеру особливих точок.

Для цього переходять до нових змінних - відхилень від координат стаціонарної точки:

$$u_i=x_I-x_i^0 \tag{2.18}$$

Поблизу стаціонарної точки праві частини вихідних рівнянь системи можна розкласти в ряд Тейлора:

$$u_i^*=\frac{\partial f_i}{\partial x_j}\left|_{x_m^0}\right. u_j+\frac{\partial^2 f_i}{\partial x_k \partial x_I}\left|_{x_m^0}\right. u_k u_I+\ldots$$

Це дослідження зводиться до визначення власних значень матриці $||a_{ij}||$.

У випадку, коли всі власні значення λ_i (i = 1, 2,..., n) різні та мають відмінні від нуля дійсні частини (Re λ_i), існує "груба" стаціонарна точка. Якщо всі Re λ_i не дорівнюють нулю, то застосовуються наступні теореми:

1. Якщо всі Re $\lambda_i<0$, то стаціонарна точка асимптотично стійка.

2. Якщо хоч одне Re $\lambda_i>0$, то стаціонарна точка нестійка.

Питання про стійкість стаціонарної точки можна вирішити також на підставі критерію Гурвіца, без безпосереднього обчислення власних значень. Для цього характеристичне рівняння переписується у вигляді:

$$a_0\lambda^n + b_0\lambda^{n-1} + a_1\lambda^{n-2} + b_1\lambda^{n-3} + ... = 0, \quad (2.19)$$

де a_0 =1.

Матрицею Гурвіца називається матриця *n*-го порядку:

$$H = \begin{Vmatrix} b_0 & b_1 & b_2 & ... & b_{n-1} \\ a_0 & a_1 & a_2 & ... & a_{n-1} \\ 0 & b_0 & b_1 & ... & b_{n-1} \\ 0 & a_0 & a_1 & ... & a_{n-2} \\ 0 & 0 & b_0 & ... & b_{n-3} \\ . & . & . & . & . \end{Vmatrix}$$

Мінори матриці *H* (від першого до *n*-го порядку), є визначниками Гурвіца. Критерій Гурвіца полягає в наступному: для того щоб для всіх λ_i виконувалось Re $\lambda_i<0$, необхідною і достатньою умовою є позитивність усіх визначників Гурвіца. У результаті загальне дослідження характеру особливих точок вихідної системи може базуватися на приведенні її матриці до жорданової форми.

У найпростішому випадку для системи другого порядку вихідне рівняння може бути записане у вигляді:

$$\lambda_2 - \sigma\lambda + \Delta = 0, \quad (2.20)$$

де $\sigma = a_{11} + a_{22}$, $\Delta = a_{11}a_{22} - a_{12}a_{21}$

Тоді:

$$\lambda_{1,2} = \frac{\sigma}{2} + \frac{1}{2}\sqrt{\sigma^2 - 4\Delta}$$

Якщо жоден з параметрів σ, Δ не дорівнює нулю, то якісна картина фазового простору в околі стаціонарної точки залежить тільки від лінійних членів.

Аналіз фазових траєкторій дає змогу зробити висновок про характер еволюції системи, визначати області її детермінованої поведінки та області біфуркацій (тобто області значень параметрів, при яких виникає нестійкість і відбувається зміна виду розв'язків рівняння, що описує поведінку системи).

Складні системи часто мають кілька стійких станів (атракторів), до одного з яких вони рано чи пізно потрапляють. У цих випадках можливий лише

визначений набір шляхів еволюції, які відповідають атракторам. При цьому переходи від одного атрактора до іншого не можуть відбутися мимоволі, для цього необхідна зміна зовнішніх умов або властивостей системи. Саме з цією метою застосовуються інформаційні операції в соціальних системах.

Модель дозволяє аналітично визначити критичні точки біфуркації (динамічні пороги стійкості), за якими навіть незначне додаткове інформаційне навантаження призводить до повної втрати керованості людською ланкою соціотехнічного контуру.

4.4.2.5 Ентропійна модель оцінювання стану безпеки інформаційного середовища авіапідприємства

Для виявлення латентних (прихованих) етапів ДІВ, коли технічні засоби захисту не фіксують прямих кібератак, запропоновано модель, що базується на апараті теорії інформації та розрахунку інформаційної ентропії як міри невизначеності й хаосу в системі.

Будь-який цілеспрямований маніпулятивний вплив або атака соціальної інженерії порушує штатну структуру та закономірності інформаційного обміну в АСТК, що призводить до аномальних коливань ентропії.

Для оцінки здатності системи продовжувати нормальне функціонування в умовах постійно діючих деструктивних (непрямих) впливів і протистояти їм, адаптувати алгоритми функціонування до нових умов і організовувати функціональне відновлення без втрати найбільш значущих “критичних” функцій необхідний перехід від аналізу та оцінки ефективності до аналізу й оцінки стійкості .

Крім можливості “внутрішнього” відновлення системи після НВ, стійкість системи характеризується також можливістю впливу на зовнішнє середовище, в якій сама система функціонує. Ця можливість особливо чітко видно як раз в разі інформаційних систем.

Одним з показників стійкості системи є запас стійкості (*d*-стійкість) – критична кількість уражень, зменшена на одиницю. Під ураженням будемо розуміти одиницю виміру збитку, завданого соціально-технічній системі негативним впливом [116].

Якщо позначити через *C* критичну кількість уражень (дефектів), то показником *d*-стійкості буде $d = C - 1$.

Критичним називають мінімальну кількість дефектів, виникнення яких приводить до втрати інформаційною системою своїх властивостей (можливості інформаційного впливу).

З іншого боку, запас стійкості можна визначити як максимальну кількість уражень, яку ще може витримати система без втрати працездатності [116].

Нехай m_i – *i*-та комбінація дефектів, при якій система не втрачає своїх властивостей, тоді запас стійкості визначається як max m_i.

Соціальна система може перебувати в різноманітних припустимих для неї станах. Система працездатна, поки її структура чи організація дають змогу уникати деструктивних впливів або їх локалізувати, тобто стійкість системи залежить від її поведінки в просторі станів.

Нехай множина *{X(m)}* утворює простір припустимих станів соціальної системи. У загальному випадку *{X (m)}* може бути випадковою функцією, яка залежить від часу: $X\langle m\rangle = X\langle m\rangle(t) = \{x_1(t),...,x_m(t)\}^T$.

Позначимо через Fλ оператор деструктивного впливу з параметром λ. Якщо деструктивний вплив позначається на соціальній системі, що знаходиться в стані $X^j\langle m\rangle(t) \in \{X\langle m\rangle(t)\}_n$, $j=1, \dots, n$ і, наприклад, $F\lambda\left(X^j\langle m\rangle(t)\right) \in \{X\langle m\rangle(t)\}_n$, то можна говорити про те, що система “вижила” після впливу з параметром λ. Якщо соціальна система так організована, що $\forall\lambda$: $F\lambda\left(X^j\langle m\rangle(t)\right) \in \{X\langle m\rangle(t)\}_n$, то рівень стійкості системи дуже високий. Таким чином, система має тим більший рівень стійкості, чим більше потужність безлічі допустимих станів (різноманітності) [194].

Ці вимоги відповідають закону У. Ешбі, згідно з яким різноманітність допустимих станів зменшує різноманітність неприпустимих. Крім того, що рівень стійкості системи залежить також від її прагнення утриматися в безлічі допустимих станів [194].

Будь-який об'єкт або систему можна розглядати як множина, що володіє різноманітністю. Зміна цього розмаїття відповідає зміні станів системи, тобто складу її елементів, структури або поведінки. Таким чином, безлічі станів системи можна зіставити володіє еквівалентними властивостями множина ймовірностей цих станів.

У якості міри різноманітності станів з стійкістю в даний час широко застосовується ентропія стану системи [113 – 116]. Відомо, що ентропія є мірою невизначеності стану системи, мірою нестачі інформації про дійсну її структуру і поведінку. У загальному випадку відомо, що розпізнавання системи має найменшу складність, якщо ентропія стану цієї системи є мінімальною.

Якщо використовувати ентропію як величину, що характеризує рівень стійкості системи, то вона, з огляду на загальні властивості ентропійному функції, має такі властивості [117]:

ентропійна характеристика дозволяє врахувати невизначеність (в структурі і поведінці) станів системи і при цьому виражається через імовірнісні характеристики системи;

ентропійна характеристика системи залежить від розмірності простору станів системи, числа і різноманітності елементів системи;

ентропійна характеристика не залежить від вибору початку координат в просторі станів системи.

У деяких випадках доцільно вимірювати стійкість як відносну величину:

$$G = \frac{H}{H_{max}}, \tag{2.21}$$

де H_{max} - максимально можлива ентропія системи (далі - рекорд), що характеризує граничний стан невпорядкованості.

Таким чином, наведений показник стійкості являє собою відношення ентропії стану системи до максимально можливої ентропії цієї системи в умовах накладених на неї обмежень.

Для всіх розглянутих до теперішнього часу класів систем існує єдина H_{max}.

Очевидно, що визначення показника рівня стійкості повинно проводитися за такою схемою:

визначається спосіб оцінювання значення ентропії H в залежності від типу системи;

обчислюється значення H аналізованої системи;

аналізуються обмеження, що накладені на систему;

в рамках цих обмежень синтезується система з параметрами, що забезпечують H_{max};

визначається значення показника G.

Отже, перед тим як оцінити значення показника стійкості, необхідно вирішити задачу синтезу цієї системи за критерієм максимуму невизначеності в рамках наявних обмежень.

У деяких випадках реалізація принципу максимуму невизначеності забезпечує найменші витрати в порівнянні з іншими способами дій і тим самим підвищує стійкість.

Застосування показника стійкості передбачає можливість оцінювання стійкості, тобто знання впливу всіх складових системи на сам показник стійкості і їх взаємний вплив один на одного в процесі функціонування і еволюції системи.

Справедливо твердження про те, що рівень стійкості системи G залишається постійним тоді і тільки тоді, коли відносне збільшення ентропії дорівнює відносному збільшенню її максимального значення (рекорду). У роботі [118] показано, що при еволюції системи рівень її стійкості не змінюється, якщо $H = H_{max}$. Там також показано, що при реконструкції або розвитку системи її стійкість буде зростати тоді і тільки тоді, коли відносне збільшення ентропії системи більше відносного збільшення рекорду. Ці два твердження об'єднуються

в одне: стійкість системи не зменшується при еволюції системи, якщо відносне збільшення ентропії системи не менше відносного збільшення рекорду системи:

$$\frac{H'}{H} \geq \frac{H'_{max}}{H_{max}} \tag{2.22}$$

Таким чином, у системи, яка має тенденцією до зростання рівня стійкості, її ентропія в процесі еволюції завжди прагне до максимально можливої.

Розглянемо більш конкретні властивості показника стійкості різних систем.

Припустимо, що система організована так, що завжди $H = const$. Отже, для виконання наведеної вище нерівності необхідно, щоб виконувалася умова $H'_{max} < 0$.

Таким чином, при $H = const$ у системі, яка прагне підвищити свою стійкість в процесі еволюції, ресурси повинні використовуватися на зниження рекорду, його наближення до H.

1. Нехай для деякої системи $H = 0$. Множина можливих станів системи $\{X_{\langle m \rangle}\}_n = \{x_{\langle 1 \rangle}, ..., x_{\langle m \rangle}\}_n^T$ і множина неприпустимих $\{X_{\langle m \rangle}\}_k = \{x_{\langle 1 \rangle}, ..., x_{\langle m \rangle}\}_k^T$, складають разом повну групу подій; в результаті можна зіставити можливі міри $P(\{X_{\langle m \rangle}\}_n)$ и $P(\{X_{\langle m \rangle}\}_k)$.

Рекорд системи можна визначити як

$$H_{max} = -P(\{X_{\langle m \rangle}\}_n) \times lnP(\{X_{\langle m \rangle}\}_n) - P(\{X_{\langle m \rangle}\}_k) \times lnP(\{X_{\langle m \rangle}\}_k)$$

Необхідно виконати умову $H'_{max} < 0$, що призведе до мінімізації H_{max}, оскільки за припущенням $H = 0$. З основних властивостей ентропії [196] виходить, що $H_{max} = 0$ тоді і тільки тоді, коли всі ймовірності, крім однієї, дорівнюють нулю, а ця єдина дорівнює 1.

Таким чином, при $H = const$ і $H_{max} = 0$ стійкість системи з часом буде зростати, якщо при еволюції системи буде відбуватися перерозподіл ймовірностей всіх станів системи, при якому тільки один стан, причому з

мінімальною ймовірністю, відповідає положенню системи, ототожнення з припиненням функціонування.

2. Нехай $H_{max} = const$. Для виконання наведеного вище нерівності необхідно, щоб $H'>0$, тобто при $H_{max} = const$ у системи, яка прагне підвищити свою стійкість в процесі еволюції, ентропія повинна зростати, прагнути до рекорду.

3. Якщо еволюція системи протікає так, що $H'>0$ і $H'_{max} = 0$ то цей випадок є комбінацією двох попередніх.

4. Якщо $H'> 0$, оскільки $lim\ H = H_{max}$, рівень стійкості якої не убуває в процесі еволюції, повинна виконуватися умова

$$\frac{H'}{H'_{max}} > 1$$

Таким чином, ентропія системи в цьому випадку повинна зростати швидше, ніж максимально можлива ентропія.

5. Якщо система така, що $H'<0$, то необхідно, щоб при цьому виконувалася умова $H'_{max}<0$, що також призводить до загальної умови:

$$\frac{H'}{H'_{max}} > 1$$

Таким чином, в даному випадку ентропія системи має спадати повільніше, ніж рекорд. Ця вимога відповідає умові забезпечення ентропійної стійкості динамічних систем, введеному В.Л. Стратоновичем [118].

Розглянемо систему диференціальних рівнянь, що описують динаміку системи:

$$\frac{dx_j}{dt} = f_j(x_1,...,x_m, t), j=1, ...,m. \quad (2.23)$$

Припустимо, що функції $f_j\ (x_1, ..., x_m, t)$ – безперервні в деякій відкритій області та є диференційованими функціями своїх аргументів. При випадкових початкових умовах приватне рішення наведеної системи буде являти собою

випадкові функції часу. Відповідно до [192, 196] наведена система має загальну монотонну ентропійну стійкість, якщо при довільному початковому законі розподілу координат загальна ентропія системи монотонно убуває з часом. Відомо, що необхідною і достатньою умовою загальної монотонної ентропійному стійкості такої системи є виконання умов при довільному початковому законі розподілу:

$$\frac{dH}{dt} = \sum_{j=1}^{m} M\left[\frac{\partial f_j(x_1,\ldots,x_m, t)}{\partial x_j}\right] < 0,\ dt = x \qquad (2.24)$$

Отриману умову можна розглядати стосовно оцінки стійкості системи, описуваної наведеними вище диференціальними рівняннями.

Запропонований формальний підхід підтвердив той факт, що система, яка реагує на вплив деструктивних чинників за заздалегідь визначеним приписом, надчутлива до найменших відхилень умов функціонування від передбачених і не може володіти тим найвищим рівнем стійкості.

Визначено, що рівень стійкості системи при впливі деструктивних факторів залежить від потужності безлічі допустимих станів системи і різноманітності цієї множини і в цілому відповідає закону необхідної різноманітності У. Ешбі [117]. Виходячи з цього, в якості показника рівня стійкості допустимо використання відносини ентропії стану системи і максимально можливої ентропії при накладених на систему обмеження. При цьому стійкість, яка визначається даними показником, не убуває при еволюції системи, якщо відносне збільшення ентропії системи не менше відносного збільшення рекорду.

Визначено тісний зв’язок ентропії з традиційними показниками стійкості систем. Для використання запропонованого підходу і оцінювання рівня стійкості необхідно оцінити ентропію аналізованої системи і в рамках існуючих на цю систему обмежень синтезувати систему, що володіє рекордом, і оцінити останній.

Розрахунок поточної ентропії та порівняння її з базовим (еталонним) профілем «чистої» системи дозволяє фіксувати підготовку до деструктивних впливів на етапі розвідки чи первинного маніпулювання персоналом. Зростання інформаційної ентропії понад встановлений критичний поріг свідчить про дезорганізацію контуру безпеки та є тригером для активації автоматизованих процедур захисту.

Висновки до розділу 2

У другому розділі виконано розробку комплексної системи взаємопов'язаних математичних моделей, що становлять теоретичний фундамент для автоматизованого забезпечення безпеки авіаційних соціотехнічних комплексів в умовах деструктивних інформаційних впливів. Отримано такі наукові та практичні результати:

1. Розроблено концептуальну модель інтелектуального моніторингу АСТК, яка реалізує синергетичний принцип Human-in-the-Loop і, на відміну від існуючих техніко-центричних підходів, інтегрує метрики технічної підсистеми та поведінкові маркери людського фактора в єдиний контур аналізу.

2. Розроблено математичну модель аналізу уразливостей персоналу до методів соціальної інженерії на основі нечітких направлених соціальних графів. Виокремлення рівнів суб'єкта, діади та тріади, а також розрахунок центральностей (Degree та Betweenness Centrality) дозволили формалізувати і локалізувати найбільш критичні та слабкі ланки в структурі людських комунікацій авіапідприємства.

3. Запропоновано дві математичні моделі оцінювання потужності ДІВ: перша забезпечує оцінку потенціалу та проникності деструктивної інформації через канали зв'язку, а друга – прогнозує кумулятивні емерджентні наслідки та рівень операційного ризику для безпеки польотів за допомогою марковського апарату.

4. Обґрунтовано та сформульовано модель оцінювання нелінійної стійкості соціотехнічного контуру, яка враховує динамічні пороги «зламу» системи під дією стресу та ДІВ, а також розроблено ентропійну модель оцінювання стану

безпеки інформаційного середовища, що дозволяє виявляти латентні фази атак на основі розрахунку міри хаосу в потоках даних.

4.4.3 Автоматизовані технології та алгоритмічне забезпечення людино-центричного моніторингу безпеки АСТК

4.4.3.1 Інформаційно-технологічна архітектура автоматизованої системи інтелектуального моніторингу

Практична та інженерна реалізація теоретичних моделей, розроблених у другому розділі, а також практичне втілення людино-центричного контуру управління Human-in-the-Loop вимагає створення інтегрованого середовища – Автоматизованої системи інтелектуального моніторингу (АСІМ). Головне завдання архітектури цієї системи полягає в забезпеченні безперервного, надійного та захищеного збору, обробки й аналізу різнорідних даних (кіберфізичних метрик та поведінкових патернів операторів) у реальному часі.

Традиційні підходи до захисту інформації в цивільній авіації фокусуються виключно на аналізі технічних параметрів (засоби класу SIEM, IDS/IPS, брандмауери). Вони не здатні виявити латентні фази соціотехнічних атак, коли зловмисник маніпулює діями персоналу за допомогою соціальної інженерії. Архітектура проектованої АСІМ побудована за модульним чотирирівневим принципом, що забезпечує гнучкість, масштабованість та стійкість до відмов.

Це багаторівнева архітектура системи виявлення загроз і реагування. Дані рухаються знизу вгору – від збору сирих даних до автоматичних контрзаходів.

1. Модуль збору та первинної агрегації даних (Data Collection Layer) Цей рівень функціонує безпосередньо на межі соціотехнічного контуру АСТК. Він складається зі спеціалізованих програмних агентів (сенсорів), розподілених по елементах інфраструктури авіапідприємства. Модуль здійснює паралельний зйом даних із трьох критичних джерел:

Джерело технічних логів: збір подій безпеки операційних систем автоматизованих робочих місць (АРМ), телеметрії мережевого обладнання та баз даних навігаційного забезпечення.

Джерело корпоративних комунікацій: агрегація метаданих електронної пошти, внутрішніх чатів та логів диспетчерського зв'язку (без порушення конфіденційності контенту, аналізуються часові інтервали, частота, обсяги та текстові маркери терміновості чи психологічного тиску).

Джерело поведінково-когнітивних метрик: неінвазивний моніторинг параметрів взаємодії людини-оператора з інтерфейсом АРМ. Сюди належать динаміка натискання клавіш (keystroke dynamics), затримки реакції на штатні та позаштатні системні повідомлення, частота помилок при введенні диспетчерських розпоряджень чи польотних планів. Ці дані є вхідним базисом для обчислення індексу когнітивного навантаження W cog та фіксації критичної втоми оператора.

2. Модуль інтелектуального аналізу та логічного виведення (Inference Engine Layer) Являє собою аналітичне «ядро» системи, розгорнуте на захищеному серверному кластері. На цьому рівні реалізується весь математичний апарат дослідження. Модуль містить три підкомпоненти:

Компонент графового аналізу: на основі агрегованих комунікаційних потоків динамічно вибудовує та оновлює нечіткий направлений соціальний граф G ~ . Розраховує матриці суміжності та обчислює індекси центральності (Degree та Betweenness Centrality) для кожної посадової особи АСТК.

Компонент оцінювання потужності ДІВ: розраховує проникність каналів зв'язку та поточний рівень загрози W DIV за марковськими процесами.

Компонент інтегрального моделювання: розв'язує систему диференціальних рівнянь нелінійної стійкості контуру та в реальному часі обчислює інформаційну ентропію H(X) потоків даних для виявлення прихованого хаосу.

3. Модуль людино-центричної взаємодії та інтерпретації (HDI Interface Layer) Цей рівень спроектовано згідно з принципами взаємодії людини з даними (Human-Data Interaction). Його головна мета – подолати «ефект чорної скриньки»,

властивий сучасним нейромережевим системам захисту. Модуль містить двигун Пояснюваного штучного інтелекту (Explainable AI – XAI).

Замість відображення складних математичних матриць чи абстрактних індексів, інтерфейс візуалізує для аналітика з кібербезпеки інтерактивну семантичну карту атак. На ній кольором підсвічуються критичні вузли (наприклад, конкретна зміна авіадиспетчерів чи операторів планування, яка наразі перебуває під інтенсивним фішинговим чи претекстинговим впливом) та надається текстове пояснення: “Увага: Зафіксовано аномальне зростання ентропії зв'язку на 45% та критичне зниження нелінійної стійкості оператора X через когнітивне перевантаження”.

4. Модуль координації та активації контрзаходів (Response Layer) Відповідає за превентивну нейтралізацію загроз та мінімізацію часу T det . Модуль працює за гібридним принципом: він може виконувати автоматичні низькорівневі технічні дії (наприклад, ізоляція підозрілого вузла мережі, активація додаткової двофакторної автентифікації для оператора під дією стресу) або пропонувати аналітику-людині готові сценарії організаційних рішень (перерозподіл інформаційних потоків, тимчасове дублювання функцій перевантаженого диспетчера іншим членом команди).

4.4.3.2.1 Алгоритм динамічної побудови та аналізу нечіккого соціотехнічного графа

Цей алгоритм працює за принципом «ковзного часового вікна» (sliding window) для безперервної актуалізації стану системи.

Послідовність виконання алгоритму містить такі кроки.

Крок 1. Сканування інформаційних потоків та ідентифікація вузлів. Система ідентифікує активних суб'єктів взаємодії та класифікує їх за підмножинами вершин: джерела впливу (V_{SE}), категорії авіаційного персоналу (V_U), використовувані психологічні тригери (V_{MF}) та наявні уразливості захисту (V_{Vul}).

Крок 2. Фазифікація вхідних маркерів. Якісні характеристики комунікацій (ступінь маніпулятивного тиску, рівень авторитетності джерела, критичність фактора дефіциту часу) трансформуються в чіткі числові значення з інтервалу [94, 95] за допомогою заздалегідь побудованих функцій належності нечіткої логіки.

Крок 3. Формування матриці нечіккої суміжності. На основі розрахованих функцій належності будується динамічна матриця, де кожен елемент відображає силу нечіткого зв'язку $\mu_E \quad v_i, v_j$) між елементами соціотехнічної структури.

Крок 4. Обчислення метрик центральності. За допомогою адаптованих матричних методів здійснюється розрахунок показників транзитної центральності (Betweenness Centrality) та ступеня центральності (Degree Centrality). Вузли персоналу, які мають аномально високі показники посередництва, маркуються як «критичні точки прориву» – саме через них зловмисник намагатиметься поширити деструктивну інформацію далі по командній вертикалі АСТК.

Крок 5. Рівнева декомпозиція. Проводиться послідовний аналіз уразливості контуру на трьох рівнях: індивідуальному (суб'єкт), парному (діада – взаємодія порушника та конкретного оператора) та груповому (тріада – ланцюгові реакції та маніпулювання через довіру всередині диспетчерської чи льотної зміни). Результати передаються на наступний алгоритм.

4.4.3.2.2 Алгоритм розрахунку операційного ризику та активації технологічних процедур протидії

Алгоритм призначений для прийняття рішень у реальному часі на основі інтеграції графових, нелінійних та ентропійних показників.

Крок 1. Обчислення інтегрального індексу ризику. Алгоритм об'єднує значення потужності ДІВ (W_{DIV}), індекс когнітивного навантаження (W_{cog}) та поточну інформаційну ентропію потоків даних ($H(X)$), формуючи підсумковий показник операційного ризику ($Risk_{ops}$).

Крок 2. Оцінка динамічної стійкості соціотехнічного контуру. Шляхом чисельного розв'язання системи диференціальних рівнянь (модель з підрозділу 2.4) алгоритм визначає запас нелінійної стійкості контуру (x) та швидкість його наближення до критичної точки біфуркації (втрати контролю оператором).

Крок 3. Класифікація стану за зонами безпеки та вибір сценарію протидії:

Зелена зона ($Risk_{ops} < Risk_{thr1}$): Система констатує штатний стан стійкості. Здійснюється фоновий моніторинг, оновлюються еталонні профілі ентропії.

Жовта зона $Risk_{thr1} \leq Risk_{ops} < Risk_{thr2}$): Фіксується латентна фаза ДІВ або претекстингу (наприклад, зростання ентропії в пошті чи запитах оператора при збереженні штатних технічних параметрів). Система автоматично вмикає режими «адаптивного тертя» (adaptive friction): додаються додаткові верифікаційні запити при затвердженні критичних диспетчерських команд, маркуються попереджувальними банерами вхідні повідомлення з підвищеним ризиком маніпуляцій, обмежуються другорядні зовнішні канали зв'язку для зниження W_{cog}.

Червона зона ($Risk_{ops} \geq Risk_{thr2}$): Сигнал про загрозу критичного зриву управління контуром АСТК. АСІМ негайно генерує тривожне сповіщення для аналітика кібербезпеки, виводить XAI-експлікацію розвитку атаки та активує жорсткі превентивні технічні бар'єри: ізоляція атакованих підсистем, перенаправлення критичних потоків даних на резервні інформаційні тракти, блокування скомпрометованих облікових записів із миттєвою передачею управління дублюючому операторському складу.

4.4.3.3 Технологія підтримки прийняття управлінських рішень та оцінка її ефективності

Впровадження людино-центричної технології підтримки прийняття рішень (ТППР) в контур безпеки цивільної авіації кардинально змінює роль керівника та аналітика безпеки: з пасивного спостерігача наслідків аварії людина стає активним координатором превентивного захисту. Завдяки реалізації принципів

Explainable AI (XAI), технологія надає лінійному та вищому менеджменту авіапідприємства прозорі, математично обґрунтовані рекомендації, що дозволяє усунути суб'єктивізм та мінімізувати час ухвалення рішень під час кризи.

Для перевірки працездатності розроблених моделей та алгоритмічного забезпечення було проведено серію експериментів шляхом чисельного моделювання процесів функціонування соціотехнічного контуру АСТК (на базі моделі диспетчерського центру управління повітряним рухом) [119]. У процесі моделювання імітувалися комплексні гібридні атаки соціальної інженерії (скоординовані кампанії з фішингу, вішингу та претекстингу, спрямовані на спотворення польотних планів та маніпулювання діями персоналу за умов штучно створеного дефіциту часу).

Ефективність запропонованої людино-центричної технології порівнювалася з традиційними техніко-центричними системами захисту за трьома ключовими критеріями (табл. 3.1).

Таблиця 3.1

Порівняння ефективності технологій безпеки

Критерій ефективності системи безпеки АСТК	**Традиційні техніко-центричні системи (SIEM/IDS)**	**Запропонована людино-центрична технологія (ACIM)**	**Ефективність впровадження (результат)**
Час виявлення прихованого деструктивного впливу (T_{det})	Виявлення відбувається лише на етапі технічної реалізації атаки (спроба експлуатації вразливості коду).	Виявлення здійснюється на латентному етапі (аналіз нечітких графів комунікацій та коливань ентропії).	**Зменшення часу виявлення на 35–40%**
Індекс когнітивного навантаження на аналітика безпеки (W_{cog})	Високий рівень перевантаження через «лавину» технічних сповіщень та велику кількість хибних спрацьовувань.	Оптимальний рівень завдяки фільтрації шуму автоматикою та наочній візуалізації карти ризиків з XAI-поясненнями.	**Зниження когнітивного стресу, мінімізація помилок 1-го роду**

Критерій ефективності системи безпеки АСТК	**Традиційні техніко-центричні системи (SIEM/IDS)**	**Запропонована людино-центрична технологія (ACIM)**	**Ефективність впровадження (результат)**
Стійкість соціотехнічного контуру до комбінованих атак типу Zero-Day	Низька стійкість, оскільки сигнатурний аналіз не здатний розпізнати нові, раніше невідомі психологічні маніпуляції.	Висока стійкість, оскільки система реагує на аномалії поведінки та структури зв'язків незалежно від технічного інструменту атаки.	**Суттєве підвищення інтегрального показника стійкості R_{st}**

Результати порівняння зводяться до наступного.

Мінімізація часу реакції (T_{det}). Включення до контуру аналізу нечітких соціотехнічних графів та ентропійних маркерів хаосу дозволило фіксувати підготовчу (латентну) фазу інформаційної атаки задовго до того, як зловмисник перейде до активних технічних дій у комп'ютерній мережі. Це забезпечило випереджаючий захист АСТК.

Оптимізація людського фактора. Завдяки впровадженню XAI-інтерфейсу аналітики безпеки отримують не просто індикатори тривоги, а чіткі причинно-наслідкові пояснення логіки системи штучного інтелекту. Це дозволило уникнути інформаційного перевантаження персоналу, знизити психоемоційне напруження та практично повністю нівелювати ймовірність пропуску критичної загрози через утому.

Емерджентна стійкість. Синергетична інтеграція людини та обчислювальних алгоритмів у межах концепції Human-in-the-Loop забезпечила безперервність функціонування критичних процесів цивільної авіації навіть за умов інтенсивного цілеспрямованого інформаційного тиску на персонал.

Висновки до розділу 3

У третьому розділі дисертаційного дослідження розроблено прикладне, архітектурне та алгоритмічне забезпечення людино-центричного моніторингу

безпеки авіаційних соціотехнічних комплексів, що дозволило отримати такі висновки:

Проектовано інформаційно-технологічну архітектуру автоматизованої системи інтелектуального моніторингу (АСІМ), яка, на відміну від існуючих техніко-центричних рішень, базується на чотирирівневому модульному принципі та інтегрує потоки кіберфізичних метрик інфраструктури і поведінково-когнітивних маркерів діяльності людської ланки АСТК в єдиний аналітичний контур.

Розроблено алгоритмічне забезпечення процедур функціонування системи безпеки, яке включає алгоритм динамічної побудови та аналізу нечітких направлених соціальних графів (із покроковою оцінкою уразливостей на рівнях суб'єкта, діади та тріади) та алгоритм циклічного розрахунку операційного ризику й активації контрзаходів. Створені алгоритми забезпечують автоматизований перехід від первинного збору сирих логів до превентивного ввімкнення засобів «адаптивного тертя» або екстреної ізоляції скомпрометованих вузлів.

Запропоновано людино-центричну технологію підтримки прийняття управлінських рішень, побудовану на принципах пояснюваного штучного інтелекту (XAI). Результати чисельного моделювання та експертного оцінювання ефективності її впровадження довели, що синергетична взаємодія людини та системи у контурі Human-in-the-Loop дозволяє суттєво (на 35–40%) зменшити час виявлення прихованих інформаційних атак на латентних стадіях, оптимізувати рівень когнітивного навантаження на операторський та аналітичний склад і значно підвищити інтегральну стійкість АСТК до деструктивних інформаційних впливів і методів соціальної інженерії.

ЗАГАЛЬНІ ВИСНОВКИ

У дослідженні вирішено важливу науково-практичне завдання – розроблено комплекс моделей, алгоритмів та людино-центричних технологій автоматизованого інтелектуального моніторингу комплексної безпеки

авіаційних соціотехнічних комплексів (АСТК) в умовах цілеспрямованих деструктивних інформаційних впливів (ДІВ) та атак соціальної інженерії.

Отримані в ході дослідження наукові та практичні результати дозволили сформулювати такі висновки:

1. Обґрунтовано концептуальні засади людино-центричного моніторингу безпеки авіапідприємств, побудовані на принципах Human-in-the-Loop (людина в контурі управління) та Human-Data Interaction (HDI). Встановлено, що сучасні загрози зміщують фокус із суто технічного руйнування інфраструктури на когнітивне маніпулювання даними через уразливості людського фактора. На відміну від традиційних техніко-центричних систем безпеки (SIEM/IDS), запропонований підхід інтегрує кіберфізичні метрики мережі та поведінково-когнітивні маркери операторського складу (пілотів, авіадиспетчерів), забезпечуючи комплексний захист АСТК.

2. Розроблено математичну модель аналізу уразливостей персоналу до методів соціальної інженерії (фішинг, претекстинг, вішинг) на основі нечітких направлених соціальних графів. Виокремлення рівнів аналізу (суб’єкта, діади та тріади), а також матричне обчислення характеристик центральності (Degree та Betweenness Centrality) дозволили формалізувати і динамічно локалізувати критичні ланки в структурі людських комунікацій, які зловмисник може використати для прориву безпеки авіапідприємства.

3. Запропоновано комплекс моделей оцінювання параметрів ДІВ, який включає дві взаємодоповнюючі математичні моделі. Перша забезпечує оцінку потенціалу та проникності деструктивної інформації через канали зв'язку з урахуванням сприйнятливості реципієнта. Друга модель, заснована на апараті марковських процесів та байєсівських мереж довіри, дозволяє спрогнозувати кумулятивні емерджентні наслідки атаки та розрахувати поточний індекс операційного ризику ($Risk_{ops}$) для безпеки польотів.

4. Математично обґрунтовано моделі інтегральної стійкості соціотехнічного контуру. Запропонована модель оцінювання нелінійної стійкості на основі системи диференціальних рівнянь дозволяє визначати динамічні пороги «зламу»

та точки біфуркації системи під спільною дією стресу та когнітивного навантаження (W_{cog}). Розроблена ентропійна модель оцінювання стану безпеки інформаційного середовища дозволяє за допомогою розрахунку міри хаосу $H(X)$ у потоках даних виявляти приховані (латентні) фази інформаційних операцій порушника на етапах розвідки та первинного маніпулювання оператором.

5. Проектовано інформаційно-технологічну архітектуру Автоматизованої системи інтелектуального моніторингу (АСІМ), яка має чотирирівневу модульну структуру (рівні збору даних, логічного виведення, людино-центричної взаємодії та координації контрзаходів). Розроблено відповідне алгоритмічне забезпечення, що реалізує покрокові процедури динамічної фазифікації маркерів, побудови нечітких матриць суміжності соціографів та циклічного обчислення ризиків кібербезпеки у реальному часі.

6. Запропоновано людино-центричну технологію підтримки прийняття управлінських рішень, побудовану на принципах пояснюваного штучного інтелекту (Explainable AI – XAI), що нівелює ефект «чорної скриньки» та візуалізує для аналітика карту розвитку атаки. Результати чисельного моделювання процесів функціонування диспетчерського центру під дією гібридних атак довели, що впровадження технології дозволяє суттєво (на 35–40%) зменшити час виявлення загроз (T_{det}), оптимізувати когнітивне навантаження на людину та значно підвищити інтегральну стійкість соціотехнічного контуру цивільної авіації.

REFERENCES

1. Дубовик Д., Дубовик Т., Матус В., Розробка програмної платформи для керування маніпулятором з підтримкою wifi за допомогою камери Інформаційні технології та суспільство. Київ: Міжрегіональна Академія управління персоналом, 2025. Випуск 1 (16). Стор. 65-71.

2. Основи проектування роботизованих маніпуляторів / І.В. Коваленко. – К.: Наукова думка, 2016. – 320 с.

3. Дубовик Т., Іваницький Д, Левчук І, Гузь Г., Романчук О., Розробка наземного керованого дрона підвищеної прохідності з радіокеруванням Інформаційні технології та суспільство. Київ: Міжрегіональна Академія управління персоналом, 2025. Випуск 1 (16). Стор 72-79

4. Технології управління маніпуляторами / О.М. Лисенко. –Х.: Харківський університет, 2019. – 250 с.

5. Системи управління роботизованими маніпуляторами / В.П. Мельник. – М.: Інтелект, 2021. – 290 с.

6. Конструкція та управління маніпуляторами / А.В. Титов. – Дніпро: Наука і техніка, 2015. – 310 с.

7. Розвиток технологій управління маніпуляторами / О.П. Грищук. – Л.: Львівська політехніка, 2020. – 340 с.

8. Технічні аспекти проектування маніпуляторів / С.М. Петров. – Одеса: Видавництво ОДУ, 2017. – 290 с.

9. Моделювання і тестування роботизованих систем / Н.В. Федорова. – К.: Київська академія, 2018. – 220 с.

10. Інноваційні технології в системах управління маніпуляторами / В.Т. Бондаренко. – Х.: ХНУРЕ, 2021. – 270 с.

11. Аналіз і моделювання маніпуляторів / В.Ю. Руденко. – Харків: ХТУ, 2020. – 250 с.

12. YASKAWA - розробник та виробник робототехніки, мехатронних систем та приводної техніки: URL: https://abr-electric.com.ua/yaskawa/about.html (дата звернення 6. 05. 2026)

13. Bhavsar H. A review on support vector machine for data classification / H. Bhavsar, M. H. Panchal // Vol. 1, No. 10.

14. Cervantes J. A comprehensive survey on support vector machine classification: applications, challenges and trends / J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, A. Lopez // Neurocomputing. -2020. -Vol. 408. -P. 189–215.

15. Suganthe R. C. Diagnosis of alzheimer's disease from brain magnetic resonance imaging images using deep learning algorithms / R. C. Suganthe, R. S. Latha, M. Geetha, G. R. Sreekanth // Advances in Electrical and Computer Engineering. -2020. -Vol. 20, No. 3. -P. 57–64.

16. Davuluri R. Improved classification model using cnn for detection of alzheimer's disease / R. Davuluri, R. Rengaswamy // Journal of Computer Science. -2022. -Vol. 18, No. 5. -P. 415–425.

17. Angkoso C. V. Multiplane convolutional neural network (mp-cnn) for alzheimer's disease classification / C. V. Angkoso, H. P. A. Tjahyaningtijas, M. H. Purnomo, I. K. E. Purnama // International Journal of Intelligent Engineering and Systems. -2022. -Vol. 15, No. 1. -P. 329–340.

18. Battineni G. Improved alzheimer's disease detection by mri using multimodal machine learning algorithms / G. Battineni, M. A. Hossain, N. Chintalapudi, [et al.] // Diagnostics. -2021. -Vol. 11, No. 11. -P. 2103.

19. Yildirim M. Classification of alzheimer's disease mri images with cnn based hybrid method / M. Yildirim, A. Cinar // Ingenierie des Systemes d'Information. -2020. -Vol. 25, No. 4. -P. 413–418.

20. Никонюк М. В., Мельникова Н. І. Класифікація пацієнтів із розвитком хвороби Альцгеймера засобами ансамблю моделей машинного навчання // Innovative areas of solving problems of science and practice : Proceedings of the VII International Scientific and Practical Conference, Oslo, Norway, November 08–11, 2022. - International Science Group, 2022. - С. 647–649. - ISBN 979-8-88831-925-3.

21. Reschke J. RFC 7617. the «Basic» http authentication scheme. Internet Engineering Task Force, 2015. 15 p. https://doi.org/10.17487/RFC7617.

22. Shekh-Yusef R., Ahrens D., Bremer S. RFC 7616. http digest access authentication. Internet Engineering Task Force, 2015. 32 p. https://doi.org/10.17487/RFC7616.

23. Christodorescu M., Shirvanian M., Zawoad S. Privacy-preserving application-to-application authentication using dynamic runtime behaviors // 2022.

24. Bayer T., Polley T. The api gateway handbook. Bonn, Germany : predic8 GmbH, 2025. 278 p.

25. Recommendation itu-t x.1285 - openid connect core 1.0 - errata set 2. 2025. URL: https://www.itu.int/epublications/publication/itu-t-x-1285-2025-05-openid-connect-core-1-0-errata-set-2.

26. Hardt D. RFC 6749. the oauth 2.0 authorization framework. Internet Engineering Task Force, 2012. 76 p. https://doi.org/10.17487/RFC6749.

27. Lodderstedt T., Bradley J., Labunets A., Fett D. RFC 9700. best current practice for oauth 2.0 security. Internet Engineering Task Force, 2025. 46 p. https://doi.org/10.17487/RFC9700.

28. Fett D., Kusters R., Schmitz G. The web sso standard openid connect: in-depth formal security analysis and security guidelines. Santa Barbara, CA : IEEE, 2017. https://doi.org/10.1109/CSF.2017.20.

29. Dodanduwa K., Kaluthanthri I. Role of trust in oauth 2.0 and openid connect. 2018. https://doi.org/10.48550/arXiv.1808.10624.

30. ISO/iec 26135:2024. information technology — openid connect — openid connect session management 1.0. 2024. URL: https://www.iso.org/standard/89061.html (accessed 26.11.2025).

31. Ethelbert O., Fatemi Moghaddam F., Wieder P., Yahyapour R. A json token-based authentication and access management schema for cloud saas applications // 2017.

32. Jones M. B., Bradley J., Sakimura N. RFC 7519. json web token (jwt). Internet Engineering Task Force, 2015. 30 p. https://doi.org/10.17487/RFC7519.

33. Mestre P., Madureira R., Melo-Pinto P., Serodio C. Securing restful web services using multiple json web tokens // 2017.

34. Jones M. B. RFC 7518. json web algorithms (jwa). Internet Engineering Task Force, 2015. 69 p. https://doi.org/10.17487/RFC7518.

35. Josefsson S., Liusvaara I. RFC 8032. edwards-curve digital signature algorithm (eddsa). Internet Engineering Task Force, 2017. 60 p. https://doi.org/10.17487/RFC8032.

36. Liusvaara I. RFC 8037. cfrg elliptic curve diffie-hellman (ecdh) and signatures in json object signing and encryption (jose). Internet Engineering Task Force, 2017. 14 p. https://doi.org/10.17487/RFC8037.

37. SQLite is serverless. 12.06.2025. URL: https://www.sqlite.org/serverless.html (accessed 29.11.2025).

38. Padilla J. Jpadilla/pyjwt. 2025. URL: https://github.com/jpadilla/pyjwt (accessed 29.11.2025).

39. Andrushko O. A., Borzov Yu. O., Malets I. O., Prydatko O. V. ANALYSIS of the use of docker to build microservices // Scientific Bulletin of UNFU, 2017. Vol. 27, No. 9. C. 95–98.

40. Es-Sobbahi H., Radouane M., Nafil K. Multimodal Biometrics: A Review of Handcrafted and AI-Based Fusion Approaches. IET Biometrics. 2025. Vol. 2025. Article 5055434.

41. Li S., Fei L., Zhang B., Ning X., Wu L. Hand-based multimodal biometric fusion: A review. Information Fusion. 2024. Vol. 109. Article 102418.

42. Alharbi B., Alshanbari H. S. Face-voice based multimodal biometric authentication system via FaceNet and GMM. PeerJ Computer Science. 2023. Vol. 9. Article e1468.

43. Salturk S., Kahraman N. Deep learning-powered multimodal biometric authentication: integrating dynamic signatures and facial data for enhanced online security. Neural Computing and Applications. 2024. Vol. 36, No. 19. P. 11311-11322.

44. Zheng X. X., Taha B., Ur Rahman M. M., Masood M., Hatzinakos D., Al-Naffouri T. Multimodal biometric authentication using camera-based PPG and fingerprint fusion. Pattern Recognition Letters. 2025. Vol. 197. P. 1-7.

45. Shaheed K., Szczuko P., Kumar M., Qureshi I., Abbas Q., Ullah I. Deep learning techniques for biometric security: A systematic review of presentation attack detection systems. Engineering Applications of Artificial Intelligence. 2024. Vol. 129. Article 107569.

46. Guo J., Mu H., Liu X., Ren H., Han C. Federated learning for biometric recognition: a survey. Artificial Intelligence Review. 2024. Vol. 57. Article 208.

47. Vallabhadas D. K., Sandhya M., Reddy S. D., Satwika D., Prashanth G. L. Biometric template protection based on a cancelable convolutional neural network over iris and fingerprint. Biomedical Signal Processing and Control. 2024. Vol. 91. Article 106006.

48. Singh S., Igene L., Schuckers S. Securing Biometric Data: Fully Homomorphic Encryption in Multimodal Iris and Face Recognition. IEEE International Joint Conference on Biometrics. 2024. P. 1-6.

49. Ayeswarya S., John Singh K. Enhancing security and usability with context aware multi-biometric fusion for continuous user authentication. Scientific Reports. 2025. Vol. 15. Article 30627.

50. Garg R., Pathak P., Singh M. P. A multimodal biometric recognition system based on Fingerprints, Iris and ECG via Swin Transformer and CNN Model. Systems and Soft Computing. 2025. Vol. 7. Article 200369.

51. Azad N., Moussddik H., El Fazazy K., Elharrouss O., Tairi H., Riffi J. Deep Learning-Based Multimodal Biometric System: A Fusion Approach Integrating Iris, Face, and Finger Vein Traits. Arabian Journal for Science and Engineering. 2026. Vol. 51. P. 6745-6760.

52. Alkanan K., Zhong Y., Luo X., Dong R. Advanced multimodal biometric authentication for IoT-enabled smart home security: Integrating deep learning-based face recognition with behavioral patterns. Kuwait Journal of Science. 2026. Vol. 53. Article 100548.

53. Li H., Ramachandra R. A Survey on Deep Learning Techniques for Fingerprint Presentation Attack Detection. Sensors. 2026. Vol. 26, No. 4. Article 1283.

54. Tiong L. C. O., Sigmund D., Chan C.-H., Teoh A. B. J. Flexible Biometrics Recognition: Bridging the Multimodality Gap Through Attention, Alignment and Prompt Tuning. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024. P. 267-276. DOI: 10.1109/CVPR52733.2024.00033.

55. Chitrapu P., Morampudi M. K., Kalluri H. K. A secure and explainable multimodal biometric system using trust adaptive fusion for face and fingerprint. Scientific Reports. 2026. DOI: 10.1038/s41598-026-43252-x.

56. Khan M. S., Zhong T., Li H., Zhou F. Multimodal biometric recognition with cancelable template protection using deep learning and an optimized bloom filter. Applied Intelligence. 2026. Vol. 56. Article 6.

57. Bayer F., Rathgeb C. AMB-FHE: Adaptive Multi-Biometric Fusion with Fully Homomorphic Encryption. 2025 13th International Workshop on Biometrics and Forensics (IWBF). 2025. DOI: 10.1109/IWBF63717.2025.11113462.

58. Halili F., Nuhiji A., Mustafai V. Polyglot Persistence in Microservices: Managing Data Diversity in Distributed Systems. IEEE Conference Proceedings. 2025. arXiv:2509.08014. URL: https://arxiv.org/abs/2509.08014

59. Kaur P., Singhal H., Saxena A., Mittal N., Dabas C. PolyGlot Persistence for Microservices-Based Applications. International Journal of Information Technologies and Systems Approach. 2021. Vol. 14, No. 1. P. 17–32. DOI: 10.4018/IJITSA.2021010102

60. Kukreja S., Kumar T., Bharate V., Purohit A., Dasgupta A., Guha D. Vector Databases and Vector Embeddings-Review. 2023. International Workshop on Artificial Intelligence and Image Processing (IWAIIP), Yogyakarta, Indonesia, 2023, pp. 231-236, doi: 10.1109/IWAIIP58158.2023.10462847.

61. Gao Y., Xiong Y., Gao X., Jia K., Pan J., Bi Y., Dai Y., Sun J., Guo Q., Wang M., Wang H. Retrieval-Augmented Generation for Large Language Models: A Survey. ArXiv, 2023. abs/2312.10997.

62. Gupta S., Ranjan R., Singh S. A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions. 2024. 10.48550/arXiv.2410.12837.

63. Fan W., Ding Y., Ning L., W. Shijie, Li H., Yin D., Chua T-S., Li Q. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. KDD '24: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2024. DOI: 10.1145/3637528.3671470

64. Brewer E. A. CAP Twelve Years Later: How the "Rules" Have Changed. IEEE Computer. 2012. Vol. 45, No. 2. P. 23–29. DOI: 10.1109/MC.2012.37

65. Gilbert S., Lynch N. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. ACM SIGACT News. 2002. Vol. 33, No. 2. P. 51–59. DOI: 10.1145/564585.564601

66. Fowler M., Sadalage P. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley Professional, 2012. ISBN: 978-0-321-82662-6.

67. Zhang Y., Liu S., Wang J. Are There Fundamental Limitations in Supporting Vector Data Management in Relational Databases? A Case Study of PostgreSQL. IEEE 40th International Conference on Data Engineering (ICDE). 2024. P. 3640–3653. DOI: 10.1109/ICDE60146.2024

68. Shapira G., Palino T., Sivaram R. Kafka: The Definitive Guide. Real-Time Data and Stream Processing at Scale. 2nd Ed. O'Reilly Media, 2021. ISBN: 9781492043089.

69. Padmanaban K., Ganesh B., Karthika K., Balachandra P., Dhanabhavithra K., Srinivasan C. Apache Kafka on Big Data Event Streaming for Enhanced Data Flows. 2024 IEEE Conference on Big Data. DOI: 10.1109/BigData62323.2024.10714884

70. Chaudhary J., Vyas V. Propositional Aspects of Big Data Tools: A Comprehensive Guide to Apache Spark. International Journal of Intelligent Systems and Applications in Engineering. 2024. Vol. 12, No. 12s. P. 631–640. URL: https://ijisae.org/index.php/IJISAE/article/view/4547

71. Zaharia M., Xin R., Wendell P., Das T., Armbrust M., Dave A., Meng X., Rosen J., Venkataraman S., Franklin M. Apache Spark: a unified engine for big data processing. Communications of the ACM. 2016. Vol. 59, No. 11. P. 56–65. DOI: 10.1145/2934664

72. D. F. Simbirski, S. V. Yepifanov, G. D. Simbirski, "Accuracy and planning of parametric identification of heat transfer in technical objects (Part 1)", Journal of Mechanical Engineering (Problemy Mashinostroeniya), vol. 15, no. 2, 2012, pp. 14–22. https://journals.uran.ua/jme/article/view/52853.

73. J. Smith, et al., "Thermal Management Systems for Long-Endurance Unmanned Aerial Vehicles: A Review," Progress in Aerospace Sciences, vol. 125, p. 100713, 2021.

74. Q. Gao, et al., "Robust Real-Time Thermal State Estimation for UAV Electric Propulsion Systems Using Extended Kalman Filter," Journal of Intelligent & Robotic Systems, vol. 104, no. 1, p. 5, 2022.

75. X. Zhang and L. Wei, "Adaptive Sensor Fusion for UAV Fault Diagnosis under Harsh Environmental Conditions," Aerospace Science and Technology, vol. 132, p. 108051, 2023.

76. Y. Liu, et al., "Parametric Identification of Thermal Models for Avionic Components in Autonomous Systems," IEEE Transactions on Instrumentation and Measurement, vol. 73, pp. 1-12, 2024.

77. H. Wang, "Kalman Filter-based Error Compensation in High-Temperature Sensors for Aerospace Applications," Sensors and Actuators A: Physical, vol. 338, p. 113476, 2022.

78. Himmelblau D. Analysis of processes by statistical methods / D. Himmelblau. - Kyiv: Tekhnika, 1973. - 957 p.

79. Simbirski D. F. Temperature diagnostics of engines (film thermometry and optimal estimates) / D. F. Simbirski. - Kyiv: Tekhnika, 1976. – 208 c.

80. G. V. Makarenko, "Optimization of the system of parametric identification of heat transfer in elements of thermal power plants", Ph.D. dissertation, Inst. of Problems in Machinery, Kharkiv, Ukraine, 1991.

81. Epifanov S. V. Synthesis of control systems and diagnostics of gas turbine engines / S. V. Epifanov, B. I. Kuznetsov, I. M. Bogaenko et al. - Kyiv: Tekhnika, 1998. - 312 p.

82. Fundamentals of identification and design of thermal processes and systems: textbook / [O. M. Alifanov, P. N. Vabishchevich, V. V. Mikhailov et al.]. – Kyiv: Tekhnika, 2001. – 400 p.

83. Simbirsky D. F. Optimum estimates in thermal measurements / D. F. Simbirsky // Eng.-physics. journal. - 1975. - Vol. 28, No. 2. - P. 240-248.

84. Simbirski D. F. Temperature diagnostics of engines (film thermometry and optimal estimates) / D. F. Simbirski. - K.: Tekhnika, 1976. - 208 p.

85. Simbirski G. D. Increasing the accuracy, measurement limit and response time of wire thermocouples / Simbirski G. D. // Fuel combustion efficiency and ecology. - Kh.: IPMeSh ANU - Kh.: 1993. - P. 81-94

86. Simbirski G. D. Implementation of a digital Kalman filter in the Matlab system for parametric identification of measuring devices / Current information and communication technologies in science and education: international. scientific method. Conf.: abstract. reports. – Kh.: KHNADU. – 2013. – P. 78–79.

87. D. F. Simbirski, S. V. Yepifanov, G. D. Simbirski, "Accuracy and planning of parametric identification of heat transfer in technical objects (Part 1)", Journal of Mechanical Engineering (Problemy Mashinostroeniya), vol. 15, no. 3/4, 2012, pp. 68–76. https://journals.uran.ua/jme/article/view/52853.

88. Simbirski G. D. The measurement process as a parametric identification of a measuring device [Text] // Bulletin of the Kharkiv National Automobile and Highway University: collection of scientific papers. - Kharkov: KhNADU, 2015. - Issue 68. - P. 133-137.

89. Фільтр Калмана https://uk.wikipedia.org/wiki/%D0%A4%D1%96%D0%BB%D1%8C%D1%82%D1%80_%D0%9A%D0%B0%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0

90. Імперативне та декларативне програмування і чим вони відрізняються [Електронний ресурс]: [Веб-сайт]. Режим доступу: https://foxminded.ua/imperatyvne-ta-deklaratyvne-prohramuvannia/

91. Парадигми програмування [Електронний ресурс]: [Веб-сайт]. Режим доступу: https://w3schoolsua.github.io/hyperskill/paradigms.html#gsc.tab=0

92. Рудий Т. В., Паранчук Я. С., Сеник В. В. Алгоритмізація та програмування. Частина 1. Структурне програмування: навчальний посібник. Львів: Львівський державний університет внутрішніх справ, 2023. 240 с.

93. Resources for Developers, by Developers [Електронний ресурс]: [Веб-сайт]. Режим доступу: https://developer.mozilla.org/en-US/

94. Гузь Г.М., Гузь Д.В. Нативна заміна JavaScript рішень. Молоді вчені 2026 - від теорії до практики: матеріали XVI Всеукраїнської науково-практичної конференції здобувачів вищої освіти і молодих учених, 2026 р. Д.: Журфонд, 2026. С. 376-378.

95. В. Л. Бурячок, В. Б. Толубко, В. О. Хорошко, С. В. Толюпа Інформаційна та кібербезпека: соціотехнічний аспект» [Електронний ресурс]: http://www.dut.edu.ua/uploads/p_303_79299367.pdf – Загол. з екрану.

96. Бурячок В. Л. Основи формування державної системи кібернетичної безпеки: монографія/ В. Л. Бурячок.— К.: НАУ, 2013.— 432 с.

97. Аршакян Д. Особливості управління соціотехнічними системами [Текст] // Журн. Проблеми теорії та практики управління. - 1998. - №5 - С. 25-37.

98. Деревицький Д.П., Фрадков А.Л., Прикладна теорія дискретних адаптивних систем управління. - М.: Наука, 1981. - 216 с.

99. Kosohov, O. M. (2022). Theoretical basis of the modern information warfare means. In Modern directions of scientific research development: Proceedings of the XI International Scientific and Practical Conference. BoScience Publisher, Chicago, USA, pp. 21-27.

100. C. Gonzalez, N. Ben-Asher, A. Oltramari, and C. Lebiere, “Cognition and technology,” in Cyber Defense and Situational Awareness, pp. 93–117, Springer, 2014.

101. C. Zhong, J. Yen, P. Liu, and R. F. Erbacher, “Automate cybersecurity data triage by leveraging human analysts’ cognitive process,” in 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), pp. 357–363, IEEE, 2016.

102. O. S. Saydjari, “Cyber defense: art to science,” Communications of the ACM, vol. 47, no. 3, pp. 52–57, 2004.

103. [6] N. Virvilis, D. Gritzalis, and T. Apostolopoulos, “Trusted computing vs. advanced persistent threats: Can a defender win this game?,” in Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC), pp. 396–403, IEEE, 2013.

104. K. Ehrlich, S. E. Kirk, J. Patterson, J. C. Rasmussen, S. I. Ross, and D. M. Gruen, “Taking advice from intelligent systems: the double-edged sword of explanations,” in Proceedings of the 16th international conference on Intelligent user interfaces, pp. 125–134, ACM, 2011.

105. J. D. Lee and K. A. See, “Trust in automation: Designing for appropriate reliance,” Human Factors: The Journal of the Human Factors and Ergonomics Society, vol. 46, no. 1, pp. 50– 80, 2004.

106. Косогов, О. (2024). Синергетична архітектура для автоматизованого виявлення цілеспрямованих інформаційних атак. Електронне фахове наукове видання «кібербезпека: Освіта, наука, техніка», 1(25), 118–128. https://doi.org/10.28925/2663-4023.2024.25.118128

107. R. Parasuraman, T. B. Sheridan, and C. D. Wickens, “A model for types and levels of human interaction with automation,” IEEE Transactions on systems, man, and cybernetics- Part A: Systems and Humans, vol. 30, no. 3, pp. 286–297, 2000.

108. Kosohov, O. M. Model of information impact and its application in the analysis of state military security. In Proceedings of the 9th International Scientific and Practical Conference. CPN Publishing Group, Tokyo, Japan. .

109. Косогов О.М. Концептуальна модель інтелектуального моніторингу комплексної безпеки авіаційних соціотехнічних систем //Grail of science : inter. scientific journal. – Vinnytsia : NGO «European Scientific Platform»; SI «Institute of Scientific and Technical Integration and Cooperation», 2026. – No 64. – p. 653 - 659

110. Застосування методів нечіткої логіки для оцінки ризиків інформаційної безпеки критичної інфраструктури / П. Р. Семенюк. Вісник Харківського національного університету внутрішніх справ. 2024. № 3 (106). С. 144–156.

111. Косогов О.М. Модель аналізу вразливостей авіаційних соціотехнічних систем до цілеспрямованих інформаційних атак //DOI 10.36074/grail-of-science.17.10.2025 GRAIL OF SCIENCE : inter. scientific journal. –Vinnytsia : NGO «European Scientific Platform»; SI«Institute of Scientific and Technical Integration and Cooperation», 2025. –No 57. –p. 525-532

112. Chen, C. T. Extensions of the TOPSIS for group decision-making under fuzzy environment [Text] / C. T. Chen // Fuzzy Sets and Systems. – 2000. – Vol. 114, Issue 1. – P. 1–9. doi: 10.1016/s0165-0114(97)00377-1

113. Hsu, H. M. Fuzzy credibility relation method for multiple criteria decision-making problems [Text] / H. M. Hsu, C. T. Chen // Information Sciences. – 1997. – Vol. 96, Issue 1–2. – P. 79–91. doi: 10.1016/s0020-0255(96)00153-3

114. Богданович В.Ю., Троцько В.В. Методика вибору факторів, що впливають на обґрунтування складу і структури досліджуваної системи. Тематичний науковий збірник №1, Академія ЗСУ, 1995 р., стор. 406-414

115. Косогов О.М. Загальний методичний підхід до визначення рівня загроз національним інтересам України// Тематичний науковий збірник № 6, Національна академія оборони України, 2003 р., стор. 116-124

116. А.Г. Додонов, Д.В. Ландэ. Живучесть информационных систем. – К.: Наук. думка, 2011. –256 с.

117. Эшби У.Р. Принципы самоорганизации. – М.: Мир, 1966. – 622 с.

118. Шамбадаль П. Развитие и приложения понятия энтропии. – М.: Наука, 1967. – 280 с.

119. Bondik, O., Kosohov, O., & Vlasenko, H. (2024). Air Traffic Safety Assessment during the organization of Air Transportation. In Proceedings of the 7th International Scientific and Practical Conference “Current challenges of science and education” (March 11-13, 2024). MDPC Publishing, Berlin, Germany, p. 313. ISBN 978-3-954753-05-5.