

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОБЛЕМ МОДЕЛЮВАННЯ В ЕНЕРГЕТИЦІ
ІМ Г.Є. ПУХОВА

Кваліфікаційна наукова
праця на правах рукопису

ДОРОГИЙ ЯРОСЛАВ ЮРІЙОВИЧ

УДК 04.93'1

ДИСЕРТАЦІЯ

**МЕТОДИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ПРОЦЕСІВ
ПРОЕКТУВАННЯ КРИТИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ**

Спеціальність 05.13.05 – комп'ютерні системи та компоненти

Галузь знань – інформаційні технології

Подається на здобуття наукового ступеня
доктора технічних наук

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

Я. Ю. Дорогий

Науковий консультант

МОХОР Володимир Володимирович,
чл.-кор. НАН України,
доктор технічних наук, професор

АНОТАЦІЯ

Дорогий Я.Ю. **Методи підвищення ефективності процесів проектування критичної інформаційної інфраструктури.** – На правах рукопису.

Дисертація на здобуття наукового ступеня доктора технічних наук за спеціальністю 05.13.05 «Комп'ютерні системи та компоненти». – Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова Національної академії наук України. – Київ, 2021.

Темпи впровадження інформаційних технологій в усі сфери соціально-економічної та виробничої діяльності суспільства, що посилюються останнім часом свідчать про те, що для соціально-економічного розвитку держави все більшу роль відіграє інформація. Саме інформаційна сфера є визначальним фактором темпів розвитку постіндустріального суспільства на сучасному етапі.

В даний час одними з основних споживачів інформаційних технологій є ІТ-інфраструктури бізнес-підприємств, а особливо, ІТ-інфраструктури державних підприємств. Ці ІТ-інфраструктури відносяться до класу критичних систем, оскільки колізії в їх функціонуванні, як правило, призводять до значних фінансових, іміджевих та інших видів втрат. Сучасні тенденції розвитку ІТ в області автоматизації управління критичними об'єктами в першу чергу пов'язані з інтеграцією різномірних застосувань в єдину ІТ-інфраструктуру в умовах впровадження нових методів і технологій, які дозволяють перебудовувати вже існуючі інформаційні системи та ІТ-інфраструктури для підтримки інноваційних процесів. При цьому ІТ-інфраструктури повинні забезпечувати роботу та реінжиніринг критичних систем в режимі реального часу з мінімальними витратами і при цьому не знижувати рівень показників надійності, безпеки, масштабованості, ефективності, керованості і гарантованості якості інформаційної обробки критичних процесів і сервісів.

У зв'язку з цим активно розвивається напрямок, пов'язаний з ефективною організацією управління критичними ІТ-сервісами та процесами. Одним із

сучасних підходів, що дозволяє організувати ефективне управління ІТ-сервісами, є підхід ITSM (IT Service Management, управління послугами ІТ). Даний підхід базується на концепціях інформаційного управління, викладених в ITIL (IT Infrastructure Library, бібліотека інфраструктури інформаційних технологій) і описують процесний підхід до надання ІТ і забезпечення їх використання. В рамках цієї концепції виділяється структурний підрозділ – служба Service Desk, який реалізує процесний підхід до управління гарантованим рівнем підтримки ІТ-сервісів. Однак, науково-методичні аспекти в області інформаційного управління саме критичними ІТ-сервісами та процесами потребують подальшого опрацювання та розвитку. Проблеми ефективного управління бізнес-процесами, з урахуванням «концепції критичності», розглянуті в працях таких дослідників, як Я.В. Бон, Г. Кемерлінг, Д. Пондаман, П.Брукс, О.І. Шелухін, Т.Л. Бірюкова, В.К. Батоврін, Г.Н. Ісаєв, В.В. Репін. У свою чергу, аналіз роботи систем підтримки ІТ-сервісів і процесів для критичних систем (критичних інфраструктур) дозволяє віднести їх до класу складних організаційно-технічних систем критичного застосування або, іншими словами, до критичних інформаційних інфраструктур.

Разом з тим, автоматизація процесів критичної інформаційної інфраструктури поряд з незаперечними перевагами неминуче веде і до негативних явищ. Найбільш характерним негативним фактором виступає прогресуюча інформатизація криміналу і терористичних угруповань, що істотно загострює проблему протиборства в сфері інформаційних ресурсів.

Першочерговими об'єктами протиправних дій терористичних і кримінальних спільнот є інформаційні ресурси критичної інфраструктури, збиток від порушення інформаційної безпеки яких може призвести до техногенних катастроф з людськими жертвами. Прикладом тому є жахливі терористичні акти, здійснені протягом останніх 15 років в США, Іспанії, Франції та інших регіонах земної кулі.

До критичних інфраструктур, поряд з системами управління інфраструктурою зв'язку, фінансів, енергетики, транспорту, водопостачання та

надзвичайних служб, відносяться органи державного управління, в тому числі і системи управління силових структур.

Інформаційні елементи цих структур є критично важливими сегментами інформаційної сфери держави. Особливо актуальною ця проблема стає у зв'язку з необхідністю розробки і впровадження критичних інформаційних інфраструктур на підприємствах державної сфери. Під критичною інформаційною інфраструктурою розуміють автоматизовані системи управління та інформаційно-телекомунікаційні мережі, в яких циркулюють інформаційні потоки, що вимагають особливого захисту.

В останні роки спостерігається стійка тенденція наростання потоку інформації, що циркулює в системах управління різного призначення. На сучасному етапі виникає необхідність впровадження мережевих, автоматизованих методів передачі та обробки критичної інформації. Одним із шляхів вирішення цієї проблеми є розгортання сучасної ІТ-інфраструктури на базі сучасних мережевих технологій.

Технологія розробки ІТ-інфраструктур досить глибоко опрацьована і накопичений досвід в цій сфері може бути використаний при проектуванні ІТ-інфраструктур для критичних інфраструктур. Однак проектування і впровадження критичної інформаційної інфраструктури стримується через властиву їм специфіку, яку можна звести до трьох чинників, а саме: відсутності прийнятної математичної інструментарію для оцінки альтернативних проектних рішень, вибору ефективних елементів з існуючого набору пропонованих типових засобів інформаційної сфери критичної інформаційної інфраструктури з врахуванням особливих вимог, які до неї ставляться; умов експлуатації, що здійснюється при постійних потужних інформаційних впливах, обумовлених підвищеним інтересом до інформаційних ресурсів критичної інформаційної інфраструктури; забезпечення вибору оптимальної архітектури критичної інформаційної інфраструктури в залежності від задач проектування, а в подальшому, від реального навантаження на її елементи.

Успішне проектування критичної інформаційної інфраструктури стає можливим при вирішенні задач представлення та обґрунтування вибору архітектури критичної інформаційної інфраструктури, оцінки і вибору ефективних елементів та компонент з існуючого набору пропонованих типових засобів. Тому актуальним є завдання створення прийнятного математичного інструментарію для оцінки ефективності вибору проектних рішень та окремих елементів архітектури критичної інформаційної інфраструктури з існуючого набору пропонованих типових засобів інформаційної сфери КІІ, та їх вибору з точки зору забезпечення заданих критеріїв проектування критичної інформаційної інфраструктури.

До другого фактору слід віднести умови експлуатації, яка здійснюється при безперервних потужних інформаційних впливах, обумовлених підвищеним інтересом до інформаційних ресурсів критичної інформаційної інфраструктури з боку кримінальних і терористичних угруповань.

Тут виникає проблема забезпечення високої експлуатаційної надійності функціонування критичної інформаційної інфраструктури в умовах інформаційної протидії, яка передбачає забезпечення заданої оперативності та достовірності передачі та прийому інформації, що циркулює в критичній інформаційній інфраструктурі. Для вирішення цієї проблеми потрібне теоретичне і технічне обґрунтування адаптивних алгоритмів функціонування критичної інформаційної інфраструктури в умовах перешкод природної та організованої структури, які б дозволили відслідковувати наявність ресурсів та навантаження на елементи критичної інформаційної інфраструктури, та у разі необхідності, виконувати балансування навантаження або робити перерозподіл наявних ресурсів з метою забезпечення функціонування критичних сервісів, що визначає актуальність досліджень в напрямку вирішення проблем, обумовлених другим з чинників.

Нарешті третій чинник обумовлений забезпеченням вибору оптимальної архітектури критичної інформаційної інфраструктури, яка в залежності від задач проектування, а в подальшому, від реального навантаження на її елементи,

повинна комплектуватися певним набором функціональних блоків, при якій буде забезпечуватися гарантована ефективність її функціонування в особливі періоди кризових ситуацій, коли інтенсивність навантаження зростає. На даний час технологія такого вибору відсутня. Вирішення даної проблеми потребує теоретичного дослідження та є ще одним з актуальних завдань дисертаційного дослідження.

Зазначені проблеми повинні розглядатися системно на основі всебічного дослідження технології забезпечення і оцінки інформаційної безпеки критичної інформаційної інфраструктури, що базується на типових засобах. На основі системного підходу повинні бути розроблені теоретичні та технічні основи забезпечення експлуатаційної надійності функціонування критичної інформаційної інфраструктури і методика оптимізації структури її архітектури, з метою забезпечення інваріантності до умов функціонування.

Рішення наведених вище проблем може здійснюватися за двома напрямками. Перший – вирішення проблеми забезпечення функціональної стабільності циркулюючої інформації в критичній інформаційній інфраструктурі, на основі дослідження існуючих технологій. Вирішення цієї проблеми характеризується безліччю нетривіальних властивостей, тому питання оцінки елементів критичної інформаційної інфраструктури відносяться до числа складних як у науковому, так і в практичному плані проблем. Необхідно знайти такі підходи в оцінці ефективності її елементів, які системно враховували б всю безліч властивостей засобів, що входять до її складу. Одним з найбільш перспективних шляхів вирішення даної проблеми є узагальнення показників, що характеризують можливості засобів критичної інформаційної інфраструктури. Разом з тим, процес узагальнення показників має ряд особливостей, і, перш за все, наявність декількох ступенів свободи в класифікації можливостей критичної інформаційної інфраструктури. Те ж саме можна сказати і про інструмент дослідження – математичні моделі процесів її проектування та вдосконалення функціональних процесів. Існування вказаних проблем дозволило в даній дисертації запропонувати новий підхід до вирішення проблеми проектування

критичної інформаційної інфраструктури шляхом оцінки можливостей критичної інформаційної інфраструктури за допомогою різних критеріїв, які дозволяють спроектувати оптимальну архітектуру під кожну окрему задачу. Суть даного підходу полягає у встановленні науково обґрунтованих правил проектування з точки зору оцінки ступеня досягнення цілей набором мінімальної кількості формул. Це досягається комплексним використанням методів математичного й комп'ютерного моделювання, методів штучного інтелекту. Тобто новизна дослідження визначається необхідністю розробки методології проектування та методів вдосконалення критичної інформаційної інфраструктури з комплексною оцінкою їх ефективності з використанням комп'ютерних і математичних моделей, які мінімізують використовуваний математичний апарат.

Другий напрямок – вирішення проблеми забезпечення оперативності і достовірності передачі критичної інформації, а також оптимізації структури критичної інформаційної інфраструктури і розроблюваної єдиної системи проектування, на базі якої і розгортається критична інформаційна інфраструктура.

Для вирішення вказаних проблем необхідна розробка і дослідження нових інформаційних технологій, які враховують взаємозв'язок показників критичності систем та елементів і рівня гарантованості підтримуваних ІТ-сервісів і процесів.

Розробка математичних та комп'ютерних моделей процесів проектування та функціонування критичної інформаційної інфраструктури відноситься до числа складних проблем як в теоретичному, так і практичному плані, через те, що опис об'єкта здійснюється в динамічному режимі з урахуванням оцінки показників всіх її елементів.

З огляду на це, актуальними, в науковому та практичному аспектах, є:

- проблема пошуку нових підходів до проектування критичної інформаційної інфраструктури, які б системно враховували всю множину зазначених вище факторів, і можливості засобів, які входять до їх складу;

– проблема розробки і розвитку методів моделювання та підтримки прийняття рішень в критичній інформаційній інфраструктурі та розробка на їх основі єдиної системи проектування, що дозволяє проектувати, аналізувати, розвивати критичну інформаційну інфраструктуру та підтримувати гарантований рівень якості критичних ІТ-сервісів і процесів, синхронно зі змінами в ній.

На основі аналізу основних існуючих підходів до вирішення поставленої проблеми показано, що створення нових методів та моделей є актуальним завданням для подолання виявлених проблем і вимагає використання відповідних інформаційних технологій для їх вирішення. Таким чином, для досягнення поставленої мети в роботі отримано цілий ряд наукових результатів.

Розроблено метод обґрунтування проектних рішень щодо архітектури критичної інформаційної інфраструктури, в якому для порівняння альтернативних проектних рішень застосовуються множина з трьох показників ризику, що враховують досяжність цілей проектного рішення, можливість його імплементації та дотримання вимог щодо критичності, та ентропійний підхід для оцінювання їх взаємного впливу для задачі проектування критичної інформаційної інфраструктур. Метод дозволяє архітектору обирати найкращий варіант проектного рішення та оцінювати вплив окремих проектних рішень або елементів на інші проектні рішення щодо архітектури критичної інформаційної інфраструктури або на весь дизайн архітектури в цілому. Використання даного методу дозволило врахувати вплив окремих проектних рішень або елементів на інші проектні рішення щодо архітектури при проектуванні єдиної інформаційної системи Міністерства внутрішніх справ України.

Розроблено метод проектування критичних інформаційних інфраструктур на базі розширених відкритих гібридних автоматів, який поширює застосування відкритих гібридних автоматів для задачі проектування критичної інформаційної інфраструктури. Застосування методу дозволило при проектуванні центру обробки даних єдиної інформаційної системи МВС врахувати проблему взаємозалежності телекомунікаційної системи від систем охолодження та

електроживлення, і таким чином, якісно продумати резервування відповідних систем.

Розроблено метод структурної оптимізації нейронних мереж прямого поширення, який дозволяє отримувати оптимальну для вхідних даних структуру нейронної мережі, що значно підвищує можливості адаптаційного вибору моделей нейронних мереж для розв'язання задач функціонування критичних інформаційних інфраструктур. Метод перевірено на різних типах вхідних даних та показано його ефективність для розв'язанні окремих класів задач класифікації та розпізнавання.

Розроблено та програмно реалізовано метод верифікації даних в інформаційних ресурсах критичної інформаційної інфраструктури, який використовує процеси хешування і може застосовуватися для даних в знеособленому вигляді, що дозволило значно пришвидшити процес верифікації та скоротити кількість невизначених даних в національних електронних інформаційних ресурсах України при проведенні державного експерименту з верифікації даних.

Розроблено та програмно реалізовано керовану моделлю систему розподілу ресурсів критичної інформаційної інфраструктури, яка використовує методи оптимізації структури нейронних мереж та оптимізації навантаження критичної інформаційної інфраструктури, що дозволило в автоматичному режимі розподіляти ресурси критичної інформаційної інфраструктури з метою їх оптимального використання та задоволення потреб сервісів та компонент, що їх використовують.

Розроблено метод представлення та обґрунтування архітектури критичної інформаційної інфраструктури на основі розширених UML-діаграм, який дозволив спростити та пришвидшити пошук оптимальної архітектури єдиної інформаційної системи МВС, а також надав можливість зберегти всю історію проведених міркувань та обґрунтувань при її виборі.

Розроблено та програмно реалізовано метод оцінки та вибору оптимальної конфігурації компонент критичної інформаційної інфраструктури на базі

Марківського процесу прийняття рішень, який дозволив оцінити та вибрати оптимальну конфігурацію компонент при проектуванні центру обробки даних єдиної інформаційної системи МВС України, що дало можливість значно скоротити час на розробку відповідного технічного завдання.

Розроблено та програмно реалізовано метод розподілу ресурсів критичної інформаційної інфраструктури на базі генетичного алгоритму з чіткими параметрами, що надає можливість швидко визначити оптимальну схему розподілу ресурсів в центрах обробки даних з врахуванням критичності окремих процесів та сервісів. Даний метод використано для оптимізації розподілення ресурсів в центрі обробки даних єдиної інформаційної системи МВС.

Розроблено та програмно реалізовано метод оптимізації навантаження критичної інформаційної інфраструктури, який використовує метод рою часток, що надав можливість швидко визначити оптимальну схему розміщення віртуальних машин та критичних сервісів на них з врахуванням вимог щодо критичності окремих сервісів та процесів, і відповідно, оптимізувати розподіл ресурсів в центрі обробки даних єдиної інформаційної системи МВС.

Отримали подальший розвиток концепція проектування критичної інформаційної інфраструктури, типова виробнича архітектура ІТ-інфраструктури, життєвий цикл критичної інформаційної інфраструктури та метод верифікації параметризованих моделей архітектурних рішень критичної інформаційної інфраструктури, що дало можливість створити єдину систему проектування критичної інформаційної інфраструктури та набір інструментарію, який дозволяє значно прискорити та підвищити ефективність процесів проектування критичної інформаційної інфраструктури.

Ключові слова: єдина система проектування, критична інформаційна інфраструктура, керована моделлю система розподілу ресурсів, оптимізація структури, архітектура, нейронна мережа, оптимізація навантаження, розподіл ресурсів.

SUMMARY

Dorogyy Y.Y. Methods for improving critical information infrastructure design processes. – As the manuscript.

Thesis for a degree of Doctor of Engineering by specialty 05.13.05 «Computer systems and components». – Pukhov Institute for Modelling in Energy Engineering. – Kyiv, 2021.

The pace of introduction of information technologies in all spheres of social and economic and productive activity of society, which has been increasing recently, shows that information plays an increasing role for the socio-economic development of the state. It is the information sphere that is the determining factor in the pace of development of post-industrial society at the present stage.

Currently, one of the main consumers of information technology is the IT infrastructure of business enterprises, and in particular, the IT infrastructure of state-owned enterprises. These IT infrastructures are in the class of critical systems, as collisions in their operation tend to lead to significant financial, image and other losses. Current trends in IT development in the field of critical object management automation are primarily related to the integration of heterogeneous applications into a single IT infrastructure in the context of the introduction of new methods and technologies that allow to rebuild existing information systems and IT infrastructures to support innovative processes. At the same time, IT infrastructures should ensure the operation and reengineering of critical systems in real time with minimal cost, while not reducing the level of reliability, security, scalability, efficiency, controllability and quality assurance of information processing of critical processes and services.

In this regard, the direction associated with the effective organization of management of critical IT services and processes is actively developing. One of the modern approaches that enables effective management of IT services is the ITSM (IT Service Management) approach. This approach builds on the IT management concepts outlined in ITIL (IT Infrastructure Library) and describes the process approach to delivering IT and ensuring their use. Within this concept, there is a structural unit called

Service Desk, which implements a process approach to managing the guaranteed level of IT services support. However, the scientific and methodological aspects of information management of critical IT services and processes require further elaboration and development. Problems of effective management of business processes, taking into account the "concept of criticality", are considered in the works of such researchers as Ya.V. Bonn, G. Kemerling, D. Pondaman, P. Brooks, O.I. Shelukhin, T.L. Biryukova, VK Batovrin, GN Isaev, V.V. Repin. In turn, the analysis of the work of IT support systems and processes for critical systems (critical infrastructures) allows them to be assigned to the class of complex organizational and technical systems of critical application or, in other words, to critical information infrastructures.

At the same time, the automation of processes of critical information infrastructure, along with the undeniable advantages, inevitably leads to negative phenomena. The most characteristic negative factor is the progressive informatization of crime and terrorist groups, which significantly exacerbates the problem of confrontation in the field of information resources.

The primary objects of unlawful acts by terrorist and criminal communities are information resources of critical infrastructure, the damage of which can lead to technogenic catastrophes with human casualties. An example of this is the horrific acts of terrorism that have taken place over the last 15 years in the United States, Spain, France and other regions of the globe.

Critical infrastructures, along with systems for managing the infrastructure of communication, finance, energy, transport, water supply and emergency services, include public administration, including systems for managing power structures. The information elements of these structures are critical segments of the state's information sphere. This problem is especially urgent due to the need to develop and implement critical information infrastructures in public sector enterprises. Critical information infrastructure refers to automated control systems and information and telecommunication networks, which circulate information flows that require special protection.

In recent years, there has been a steady upward trend in the flow of information circulating in multi-purpose management systems. At the present stage, there is a need to introduce networked, automated methods of transmitting and processing critical information. One of the ways to solve this problem is to deploy modern IT infrastructure based on modern network technologies.

The technology of IT infrastructure development is quite deeply developed and the experience gained in this area can be used in the design of IT infrastructures for critical infrastructures. However, the design and implementation of critical information infrastructure is constrained by their inherent specifics, which can be reduced to three factors, namely: lack of acceptable mathematical tools for evaluating alternative design solutions, selection of effective elements from the existing set of typical critical information infrastructure information tools. requirements that apply to it; operating conditions carried out at constant powerful information influences caused by the increased interest to information resources of a critical information infrastructure; ensuring the choice of the optimal architecture of the critical information infrastructure depending on the design tasks, and in the future, on the actual load on its elements.

Successful design of critical information infrastructure becomes possible by solving the problems of presentation and justification of the choice of critical information infrastructure architecture, evaluation and selection of effective elements and components from the existing set of proposed standard tools. Therefore, the task of creating acceptable mathematical tools to assess the effectiveness of the choice of design solutions and individual elements of the architecture of critical information infrastructure from the existing set of proposed standard information tools for critical information infrastructure, and their choice in terms of providing specified criteria for critical information infrastructure.

The second factor includes the conditions of operation, which is carried out under continuous powerful information influences, due to the increased interest in information resources of critical information infrastructure by criminal and terrorist groups.

Here there is a problem of ensuring high operational reliability of the critical information infrastructure in the conditions of information counteraction, which involves ensuring the specified efficiency and reliability of transmission and reception of information circulating in the critical information infrastructure. To solve this problem requires theoretical and technical justification of adaptive algorithms for the operation of critical information infrastructure in the face of obstacles of natural and organized structure, which would track the availability of resources and load on critical information infrastructure, and if necessary, perform load balancing or redistribution of available resources in order to ensure the functioning of critical services, which determines the relevance of research to address the problems caused by the second factor.

Finally, the third factor is due to the choice of optimal architecture of critical information infrastructure, which, depending on the design tasks, and subsequently, the actual load on its elements, must be equipped with a set of functional units, which will ensure guaranteed efficiency in special times of crisis when the load intensity increases. Currently, the technology of such a choice is absent. The solution of this problem requires theoretical research and is another important task of dissertation research. These issues should be addressed systematically on the basis of a comprehensive study of the technology and security assessment of critical information infrastructure based on typical tools. On the basis of the systematic approach, theoretical and technical bases of maintenance of operational reliability of functioning of the critical information infrastructure and method of optimization of the structure of its information and telecommunication network should be developed, in order to ensure invariance to the operating conditions.

The above problems can be solved in two ways. The first is to solve the problem of ensuring the functional stability of circulating information in critical information infrastructure, based on the study of existing technologies. The solution to this problem is characterized by many non-trivial properties, so the issues of assessing the elements of critical information infrastructure are among the complex problems in both scientific and practical terms. It is necessary to find such approaches in the evaluation of the

effectiveness of its elements, which would systematically take into account the whole set of properties of the means included in its composition. One of the most promising ways to solve this problem is to summarize the indicators that characterize the capabilities of critical information infrastructure. At the same time, the process of generalization of indicators has several features, and above all, the availability of several degrees of freedom in the classification of capabilities of critical information infrastructure. The same can be said about the research tool - mathematical models of the processes of its design and improvement of functional processes. The existence of these problems has allowed in this dissertation to offer a new approach to solving the critical information infrastructure design problem by assessing the capabilities of critical information infrastructure using different criteria that allow to design the optimal architecture for each individual task. The essence of this approach is to establish scientifically sound design rules from the point of view of assessing the degree of achievement of goals by a set of minimum formulas. This is achieved through the integrated use of mathematical and computer modeling, artificial intelligence. That is, the novelty of the study is determined by the need to develop a design methodology and methods for improving critical information infrastructure with a comprehensive assessment of their performance using computer and mathematical models that minimize the mathematical apparatus used.

The second direction is to solve the problem of ensuring the promptness and reliability of critical information transmission, as well as optimizing the structure of the critical information infrastructure and developing a unified design system, on which the critical information infrastructure is deployed.

In order to solve these problems, it is necessary to develop and research new information technologies that take into account the interrelation between the criticality of systems and elements and the level of security of the supported IT services and processes.

The development of mathematical and computer models for the processes of designing and operating a critical information infrastructure is one of a number of complex problems, both theoretically and practically, due to the fact that the

description of the object is dynamically taking into account the evaluation of all its elements.

In this regard, relevant in scientific and practical aspects are:

- the problem of finding new approaches to the design of critical information infrastructures that would systematically take into account the whole set of factors mentioned above and the capabilities of the tools that make up them;
- the problem of designing and developing methods for modeling and supporting decision-making in critical information infrastructure and developing on their basis a single system of design that allows to design, analyze, develop critical information infrastructure and maintain the guaranteed quality level of critical IT services and processes, synchronously with changes in it.

Based on the analysis of the main existing approaches to solving the problem, it is shown that the creation of new methods and models is an urgent task for overcoming the identified problems and requires the use of appropriate information technologies to solve them. Thus, a number of scientific results were obtained in order to achieve this goal.

A method of substantiation of design decisions on the architecture of critical information infrastructure has been developed, in which a set of three risk indicators is used to compare alternative design solutions, taking into account the achievability of project solution objectives, possibility of its implementation and criticality requirements, and entropy approach to assess their mutual impact. tasks of designing critical information infrastructures. The method allows the architect to choose the best design solution and assess the impact of individual design decisions or elements on other design decisions regarding the architecture of critical information infrastructure or on the entire architecture design as a whole. The use of this method allowed to take into account the impact of individual design decisions or elements on other design decisions regarding architecture when designing a unified information system of the Ministry of Internal Affairs of Ukraine.

A method for designing critical information infrastructures based on extended open hybrid automata has been developed, which extends the use of open hybrid

automata for the task of designing critical information infrastructure. The application of the method allowed to take into account the problem of interdependence of the telecommunication system from the cooling and power supply systems when designing the data processing center of the unified information system of the Ministry of Internal Affairs, and thus to qualitatively consider the redundancy of the respective systems.

A method of structural optimization of neural networks of direct propagation has been developed, which allows to obtain the optimal neural network structure for input data, which significantly increases the possibilities of adaptive choice of neural network models to solve critical information infrastructures. The method is tested on different types of input data and its effectiveness for solving certain classes of classification and recognition problems is shown.

Developed and software implemented a method of data verification in information resources of critical information infrastructure, which uses hashing processes and can be used for impersonal data, which significantly accelerates the verification process and reduces the amount of uncertain data in national electronic information resources of Ukraine during the state verification experiment data.

A managed model of critical information infrastructure resource allocation system has been developed and programmatically implemented, which uses methods of optimizing the structure of neural networks and optimizing the load of critical information infrastructure, which allowed to automatically allocate critical information infrastructure resources to optimize their use and meet the needs of services and components they are used.

A method of presenting and substantiating the architecture of critical information infrastructure based on extended UML-diagrams was developed, which simplified and accelerated the search for the optimal architecture of the unified information system of the Ministry of Internal Affairs, and provided an opportunity to preserve the history of considerations.

Developed and programmatically implemented a method for assessing and selecting the optimal configuration of critical information infrastructure components based on the Markov decision-making process, which allowed to assess and select the

optimal configuration of components in designing a data center of a unified information system of the Ministry of Internal Affairs of Ukraine. task.

A method of resource allocation of critical information infrastructure based on a genetic algorithm with clear parameters has been developed and implemented, which provides an opportunity to quickly determine the optimal resource allocation scheme in data centers taking into account the criticality of individual processes and services. This method was used to optimize the allocation of resources in the data center of the unified information system of the Ministry of Internal Affairs.

Developed and implemented a method for optimizing the load of critical information infrastructure, which uses the method of particle swarming, which allowed to quickly determine the optimal layout of virtual machines and critical services on them, taking into account the criticality requirements of individual services and processes, and accordingly optimize resource allocation in the center data processing of the unified information system of the Ministry of Internal Affairs.

The concept of critical information infrastructure design, typical production infrastructure of IT infrastructure, life cycle of critical information infrastructure and method of verification of parametrized models of architectural solutions of critical information infrastructure were further developed, which allowed to create a single system of critical information infrastructure design and toolkit and increase the efficiency of critical information infrastructure design processes.

Keywords: unified design system, critical information infrastructure, model-driven resource allocation system, structure optimization, architecture, neural network, load optimization, resource allocation.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Монографії, в яких опубліковані основні наукові результати дисертації:

- [1] Я. Дорогий, та В. Цуркан, “Методи представлення та обґрунтування архітектури критичної ІТ-інфраструктури”, in *Engineering sciences: development prospects in countries of Europe at the beginning of the third millennium: Collective monograph*, vol. 1, Riga: Izdevnieciba “Baltija Publishing,” 2018, pp. 197–237.
- [2] Я. Дорогий, “Моделі і методи представлення архітектурних рішень критичної ІТ-інфраструктури”, in *Modern engineering research: topical problems, challenges and modernity: Collective monograph*, Riga: Izdevnieciba “Baltija Publishing,” 2020, pp. 127–156, doi: 10.30525/978-9934-588-47-1.7.

Наукові праці, в яких опубліковані основні наукові результати дисертації:

- [3] Y. Y. Dorogyu, “Management of Critical It-Infrastructures”, *Inf. Telecommun. Sci.*, vol. 0, no. 1, pp. 10–15, Aug. 2014, doi: 10.20535/2411-2976.12014.10-15.
- [4] Я. Дорогий, “Життєвий цикл критичної ІТ-інфраструктури”, *Електроніка та зв’язок*, т. 20, №4, с. 100–105, 2015.
- [5] Я. Дорогий, “Порівняльний аналіз методів представлення та обґрунтування архітектури критичної ІТ-інфраструктури”, *Вісник НТУ «ХПІ». Серія Механіко-технологічні системи та комплекси*, т. 33, №1255, с. 42–54, 2017.
- [6] Я. Дорогий, “Мета-модель компенсаційно-декомпенсаційного підходу до проектування архітектури критичної ІТ-інфраструктури”, *Вісник університету України Інформатика, обчислювальна техніка та кібернетика*, т. 2, №17, с. 14–22, 2016.
- [7] Я. Дорогий, “Розробка підходу до проектування, моделювання та дослідження критичної ІТ-інфраструктури”, *Технологічний аудит та резерви виробництва*, т. 5, №2(37), с. 34–41, 2017.

- [8] Y. Dorogyu, “Development of a model for optimal configuration components selection for architecture of critical IT infrastructure at its designing”, *Технологічний аудит та резерви виробництва*, №6/2(38), с. 19–27, 2017.
- [9] Я. Дорогий, “Алгоритм структурної оптимізації нейронної мережі”, *Вісник НТУУ «КПІ», «Інформатика, управління та обчислювальна техніка»*, №61, с. 169–173, 2014.
- [10] Я. Дорогий, “ТС-сумісна архітектура критичної ІТ-інфраструктури”, *Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка збірник наукових праць*, т. 65, с. 123–126, 2017.
- [11] Я. Дорогий, “Розподіл ресурсів критичної ІТ-інфраструктури з використанням хмарних технологій”, *Електроніка та зв’язок*, т. 1, №90, с. 42–49, 2016.
- [12] Я. Дорогий, “Технологія пошуку сумнівних записів при створенні єдиного реєстру ідентифікації фізичних осіб України”, *Inf. Technol. Secur.*, т. 7, №2, с. 114–125, 2019.
- [13] Y. Dorogyu, O. Doroga-Ivaniuk, V. Tsurkan, and S. Telenyk, “Comparative analysis of the architecture designing platform for critical it infrastructure”, *Inf. Technol. Secur.*, vol. 5, no. 2, pp. 90–118, 2017.
- [14] Я. Дорогий, І. Бондаренко, Т. Шемседінов, та С. Стиренко, “Аналіз проблем побудови критичної ІТ-інфраструктури міністерства”, *Inf. Technol. Secur.*, т. 1, №10, с. 96–107, 2018.
- [15] Y. Dorogyu, S. Telenyk, V. Tsurkan, and D. Halushko, “Structural optimization of neural network in qualitative evaluation method of IT-infrastructure functioning”, *Inf. Telecommun. Sci.*, vol. 6, no. 2, pp. 36–43, 2015.
- [16] Я. Дорогий, О. Дорога-Іванюк, В. Цуркан, та Д. Ференс, “Застосування алгоритму структурної оптимізації нейронної мережі в задачах класифікації даних”, *Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка збірник наукових праць*, №62, с. 100–104, 2015.

- [17] О. Дорога-Іванюк, Я. Дорогий, та Д. Ференс, “Реалізація алгоритму структурної оптимізації нейронної мережі”, *Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка збірник наукових праць*, №63, с. 101–106, 2015.
- [18] G. Nowakowski, Y. Dorogyu, and O. Doroga-Ivaniuk, “Neural network structure optimization algorithm,” *J. Autom. Mob. Robot. Intell. Syst.*, vol. 12, no. 1, pp. 5–13, 2018, doi: 10.14313/JAMRIS_1-2018/1. **(Журнал індексується в Elsevier Scopus)**
- [19] Я. Дорогий, Є. Максименко, О. Крук, та В. Цуркан, “Оцінювання ризику безпеки інформації на основі спектрального підходу”, *Inf. Technol. Secur.*, т. 3, №2(5), с. 138–146, 2015.
- [20] Я. Дорогий, В. Мохор, В. Цуркан, та О. Крук, “Функціональне моделювання системи керування ризиком безпеки інформації”, *Захист інформації*, т. 18, №1, с. 74–80, 2016.
- [21] Я. Дорогий, О. Бакалинський, О. Богданов, С. Михайлов, В. Цуркан, та В. Мохор, “Використання ентропійного підходу для оцінювання ризиків безпеки інформації”, *Inf. Technol. Secur.*, т. 4, №2(7), с. 255–261, 2016.
- [22] Я. Дорогий, Д. Ференс, та О. Дорога-Іванюк, “Модель розподілу ресурсів критичної ІТ-інфраструктури з чіткими параметрами на основі генетичного алгоритму”, *Inf. Technol. Secur.*, т. 2, №11, с. 124–144, 2018.
- [23] Я. Дорогий, Д. Ференс, та О. Дорога-Іванюк, “Модель розподілу ресурсів критичної ІТ-інфраструктури з чіткими параметрами на основі методу рою часток”, *Електронне моделювання*, т. 41, №2, с. 1–15, 2019.
- [24] А. Йовенко, та Я. Дорогий, “Вибір методу інтеграції розподілених гетерогенних модулів для розробки корпоративних систем”, *Вісник університету України Інформатика, обчислювальна техніка та кібернетика*, т. 1, №18, с. 14–22, 2016.

- [25] Я. Дорогий, В. Мохор, та В. Цуркан, “Концептуальна модель описання архітектури системи управління інформаційною безпекою”, *Безпека інформації*, т. 25, №3, с. 162–166, 2019.
- [26] Я. Дорогий, та В. Цуркан, “Огляд методів верифікації параметризованих моделей”, *Збірник наукових праць Національного університету кораблебудування імені адмірала Макарова*, №1(479), с. 82–91, 2020, doi: [https://dpo.org/10.15589/znp2020.1\(479\).10](https://dpo.org/10.15589/znp2020.1(479).10).
- [27] Я. Ю. Дорогий, В. В. Мохор, В. В. Цуркан, та Ю. М. Штифурак, “Структури архітектури систем управління безпекою інформації”, *Інформатика та математичні методи в моделюванні*, т. 9, № 4, с. 209–221, 2019, doi: 10.15276/imms.v9.no4.209.
- [28] G. Nowakowski, Y. Dorogyu, and O. Doroga-Ivaniuk, “The realisation of neural network structural optimization algorithm,” in *Annals of Computer Science and Information Systems (ACSIS)*, vol. 11, 2017, pp. 1365–1371. **(Журнал індексується в Elsevier Scopus)**
- [29] Д. Ференс and Я. Дорогий, “Оптимізація розміщення віртуальних машин у хмарних інфраструктурах за допомогою гнучкого алгоритму на базі генетичного програмування”, *Інфокомунікаційні системи та технології*, т. 1, №1, 2017.
- [30] К. Левченко and Я. Дорогий, “Порівняння часу виконання базових операцій фреймворків машинного навчання”, *Інфокомунікаційні системи та технології*, т. 2, №2, с. 27–31, 2018.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

- [31] S. Telenyk, O. Rolick, M. Bukasov, Y. Dorogiy, D. Halushko, and A. Pysarenko, “Qualitative evaluation method of IT-infrastructure elements functioning”, in *Proc. 2014 IEEE International Black Sea Conference on Communications and Networking, BlackSeaCom 2014*, 2014, pp. 165–169, doi:

- [32] Y. Dorogyu and K. Valchuk, “Comparison between serialization formats”, in *Collection of Conference Papers of International Scientific-Practical Conference “Problems and Prospects of Intergration of Science and Technology,”* 2015.
- [33] Я. Дорогий, І. Козлюк, В. Цуркан, та В. Мохор, “Критична інфраструктура: вразливості, загрози, ризики,” на *II міжн. наук.-практ. конф. «Інформаційні технології та взаємодії»*, 2015.
- [34] Я. Дорогий, та О. Дорога-Іванюк, “Критерій оптимальності побудови критичної ІТ-інфраструктури”, на *2-й міжн. наук.-практ. конф. Актуальні проблеми розвитку науки і техніки, 20 грудня*, 2015.
- [35] Я. Дорогий, Є. Максименко, В. Цуркан, та В. Мохор, “Аналітика кібербезпеки: поняття, структура, задачі”, на *Межд. научн. конф. ИТБ-2015, 21 октября*, 2015.
- [36] О. Дорога-Іванюк, та Я. Дорогий, “Умови створення критичних процесів або сервісів критичної ІТ-інфраструктури”, на *V міжн. наук.-техн. конф. «Проблеми інформатизації»*, 10-11 грудня, 2015.
- [37] Y. Dorogyu, O. Doroga-Ivaniuk, B. Mart, and V. Tsurkan, “Life cycle of IT-infrastructure,” in *Proceedings of The Congress on Information Technology, Computational and Experimental Physics 2015 (CITCEP 2015)*, 2015.
- [38] Y. Dorogyu, O. Doroga-Ivaniuk, S. Telenik, and D. Halushko, “Qualitative evaluation method of IT-infrastructure functioning based on structural optimization of neural network,” in *Proc. 2015 2nd International Scientific-Practical Conference Problems of Infocommunications Science and Technology, PIC S&T 2015 - Conference Proceedings*, 2015, pp. 1–4, doi: 10.1109/INFOCOMMST.2015.7357252. (Матеріали конференції індексуються в Elsevier Scopus)

- [39] Д. Ференс, та Я. Дорогий, “Структурна оптимізація штучних нейронних мереж”, на 2-й міжн. наук.-практ. конф. *Summer Infocom 2016*, 2016.
- [40] Я. Дорогий, О. Крук, В. Цуркан, та В. Мохор, “Процесна модель системи керування ризиком безпеки інформації”, на V Міжнар. на-ук.-техн. конф. *“Захист інформації і безпека інформаційних систем,”* 2016.
- [41] Я. Дорогий, О. Крук, В. Цуркан, та В. Мохор, “Дерево вузлів функціональної моделі системи керування ризиком безпеки інформації”, на Міжнар. наук.-практ. конф. *“Безпека інформації в інформаційно-телекомунікаційних системах”*, 25-26 травня, 2016.
- [42] Я. Дорогий, О. Крук, В. Мохор, та В. Цуркан, “Модель потоків даних системи керування ризиком безпеки інформації”, на V-й міжн. наук.-практ. конф. *“Кібербезпека інформаційних технологій /// Фізико-технологічні проблеми передавання, оброблення та зберігання інформації в інфокомунікаційних системах*, 2016.
- [43] Я. Дорогий, О. Крук, В. Цуркан, та В. Мохор, “Функціональна модель системи керування ризиком безпеки інформації”, на II-й міжн. наук.-практ. конф. *“Актуальні проблеми моделювання ризиків і загроз виникнення надзвичайних ситуацій на об'єктах критичної інфраструктури,”* 2016.
- [44] Y. Y. Dorogyu, O. O. Doroha-Ivaniuk, U. Dzelendzyak, and K. Tomczyk, “Usage of structural optimization algorithm of neural nets in problems of data classification”, in *Proceedings of the 2017 IEEE 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2017*, 2017, vol. 2, pp. 983–987, doi: 10.1109/IDAACS.2017.8095234. **(Матеріали конференції індексуються в Elsevier Scopus)**
- [45] Я. Дорогий, “Проектування критичних ІТ-інфраструктур з використанням розмічених транзиційних систем”, на V заочн. научн. конф. *«Фундаментальные и прикладные исследования в современной науке»*, 31 октября, 2017.

- [46] Я. Дорогий, “Моделювання критичних ІТ-інфраструктур з використанням розмічених транзиційних систем”, на *VI Міжн. наук.-практ. конференція (I Міжнародний симпозиум «Практичне застосування нелінійних динамічних систем в інфокомунікаціях», PREDT-2017, 9-11 листопада, 2017.*
- [47] Я. Дорогий, О. Дорога-Іванюк, and В. М. В. Цуркан, “Cyber resilience: the concept of information security in cyberspace”, на *III міжнар. наук.-практ. конф. «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації», 2017.*
- [48] Y. Dorogyu, “Meta model of compensatory-decompensational approach for architecture of critical it infrastructure designing”, in *Proc. 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, TCSET 2018 - Proceedings, 2018, vol. 2018-April, pp. 223–228, doi: 10.1109/TCSET.2018.8336191. (Матеріали конференції індексуються в Elsevier Scopus)*
- [49] Я. Дорогий, В. Мохор, та В. Ц. Ю. Штифурак, “Моделі оцінювання ризику кібербезпеки”, на *VII Міжн. наук.-практ. конф. «Фізико-технологічні проблеми передавання, оброблення та зберігання інформації в інфокомунікаційних системах (PREDT-2018)», 8-10 листопада, 2018.*
- [50] А. Горносталь, та Я. Дорогий, “Мурашиний алгоритм кластеризації”, на *XI Міжнар. наук.-практ. конф. "ІНТЕРНЕТ-ОСВІТА-2018, 2018.*
- [51] Я. Дорогий, та К. Левченко, “Порівняння фреймворків машинного навчання”, на *VI-й міжн. наук.-практ. конф. з інформаційних систем та технологій Summer InfoCom 2018, 2018.*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	29
ВСТУП	31
РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМ, ТЕНДЕНЦІЙ І ПЕРСПЕКТИВ РОЗВИТКУ ЕФЕКТИВНОГО ПРОЕКТУВАННЯ КРИТИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ	46
1.1 Критична інформаційна інфраструктура як об’єкт дослідження	46
1.2 Аналіз підходів проектування архітектури підприємства з критичною інфраструктурою	50
1.3 Аналіз існуючих методів та підходів представлення та обґрунтування архітектури критичної інформаційної інфраструктури	69
1.4 Висновки до першого розділу	75
РОЗДІЛ 2 МОДЕЛІ І МЕТОДИ ПРЕДСТАВЛЕННЯ АРХІТЕКТУРНИХ РІШЕНЬ КРИТИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ.....	77
2.1 Основні означення	78
2.2 Представлення архітектури критичної інформаційної інфраструктури на базі розмічених транзиційних систем	86
2.3 Верифікація моделей на базі РТС	96
2.4 Критерій оптимальності проектування та функціонування КІП.....	99
2.5 Висновки до другого розділу	103
РОЗДІЛ 3 МОДЕЛІ І МЕТОДИ ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ КРИТИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ.....	105
3.1 Метод обґрунтування рішень на базі ентропійного підходу.....	105
3.2 Оцінка та вибір оптимальної конфігурації компонент критичної інформаційної інфраструктури за допомогою Марківського процесу прийняття рішень	113

3.3 Оцінка впливу забезпечуючих систем на архітектуру критичної інформаційної інфраструктури	118
3.4 Висновки до третього розділу	124
РОЗДІЛ 4 МОДЕЛІ І МЕТОДИ ВДОСКОНАЛЕННЯ КРИТИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ.....	126
4.1 Аналіз стану і оцінювання показників елементів проектних рішень критичної інформаційної інфраструктури з використанням методу структурної оптимізації нейронних мереж при навчанні.....	126
4.2 Методи та моделі управління ресурсами критичної інформаційної інфраструктури	137
4.3 Метод верифікації даних в інформаційних ресурсах критичної інформаційної інфраструктури	166
4.4 Висновки до четвертого розділу	170
РОЗДІЛ 5 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ЗАПРОПОНОВАНИХ В РОБОТІ МОДЕЛЕЙ ТА МЕТОДІВ.....	172
5.1 Єдина інформаційна система МВС України.....	172
5.2 Дослідження моделі оцінки впливу забезпечуючих систем на архітектуру критичної інформаційної інфраструктури	186
5.3 Дослідження методу оцінки та вибору оптимальної конфігурації компонент критичної інформаційної інфраструктури за допомогою Марківського процесу прийняття рішень.....	194
5.4 Дослідження методу структурної оптимізації нейронних мереж...	205
5.5 Дослідження методу розподілу ресурсів критичної інформаційної інфраструктури на основі генетичного алгоритму з чіткими параметрами ..	215
5.6 Дослідження методу розподілу ресурсів критичної інформаційної інфраструктури на базі методу рою часток з чіткими параметрами	219

5.7 Дослідження методу верифікації даних в інформаційних ресурсах критичної інформаційної інфраструктури (на прикладі ЄІС МВС)	221
5.8 Висновки до п'ятого розділу	234
РОЗДІЛ 6 ЄДИНА СИСТЕМА ПРОЕКТУВАННЯ КРИТИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ.....	236
6.1 Концепція проектування критичної інформаційної інфраструктури	236
6.2 Структура і принципи функціонування ЄСПК.....	265
6.3 Технологічні особливості реалізації розроблюваної ЄСПК.....	267
6.4 Розробка ЄСПК	272
6.5 Висновки до шостого розділу	310
ВИСНОВКИ.....	311
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	316
ДОДАТОК А ІЛЮСТРАЦІЙНІ МАТЕРІАЛИ	339
ДОДАТОК Б ФУНКЦІОНАЛ СИСТЕМИ РОЗГОРТАННЯ КРИТИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ.....	345
Б.1 Розробка функціоналу для створення образу контейнера.....	345
Б.2 Розробка функціоналу для створення Docker Stackfile	347
ДОДАТОК В СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА	360
ДОДАТОК Г ДОВІДКИ ПРО ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОГО ДОСЛІДЖЕННЯ.....	367

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

DoDAF	The Department of Defense Architecture Framework
FEAF	The Federal Enterprise Architecture Framework
GEAF	The Gartner Enterprise Architecture Framework
ITIL	IT Infrastructure Library, бібліотека інфраструктури інформаційних технологій
ITSM	IT Service Management, управління послугами IT
MODAF	The British Ministry of Defense Architecture Framework
NAF	The NATO Architecture Framework
TEAF	The Treasury Enterprise Architecture Framework
TMN	Telecommunication Management Network, Мережа управління телекомунікаціями
TOGAF	The Open Group Architecture Framework
RM-ODP	Open Distributed Processing – Reference Model
SOA	Service-oriented architecture, сервіс-орієнтована архітектура
IT	Інформаційна технологія
ITI	IT-інфраструктура
ITM	Інформаційно-телекомунікаційні мережі
KI	Критична інфраструктура
KII	Критична інформаційна інфраструктура
KC	Критична система
OTC	Оператор телекомунікаційної системи
PC	Розподілена система
PTC	Розмічена транзитивна система
SUI	Система управління інфраструктурою
TKM	Телекомунікаційна мережа

ТС	Транзиційна система
ФОАП	Фреймворк опису архітектури підприємства
ФП ЄІС МВС	Функціональна підсистема єдиної інформаційної системи МВС
ЦП ЄІС МВС	Центральна підсистема єдиної інформаційної системи МВС
ЄІС МВС	Єдина інформаційна система МВС
ЄСПК	Єдина система проектування критичної інформаційної інфраструктури

ВСТУП

Актуальність. Темпи впровадження інформаційних технологій в усі сфери соціально-економічної та виробничої діяльності суспільства, що посилюються останнім часом свідчать про те, що для соціально-економічного розвитку держави все більшу роль відіграє інформація [52–55]. Саме інформаційна сфера є визначальним фактором темпів розвитку постіндустріального суспільства на сучасному етапі [56–58].

В даний час одними з основних споживачів інформаційних технологій (ІТ) є ІТ-інфраструктури (ІТІ) бізнес-підприємств, а особливо, ІТ-інфраструктури державних підприємств. Ці ІТІ відносяться до класу критичних систем, оскільки колізії в їх функціонуванні, як правило, призводять до значних фінансових, іміджевих та інших видів втрат. Сучасні тенденції розвитку ІТ в області автоматизації управління критичними об'єктами в першу чергу пов'язані з інтеграцією різномірних застосувань в єдину ІТ-інфраструктуру в умовах впровадження нових методів і технологій, які дозволяють перебудовувати вже існуючі інформаційні системи та ІТ-інфраструктури для підтримки інноваційних процесів. При цьому ІТІ повинні забезпечувати роботу та реінжиніринг критичних систем (КС) в режимі реального часу з мінімальними витратами і при цьому не знижувати рівень показників надійності, безпеки, масштабованості, ефективності, керованості і гарантованості якості інформаційної обробки критичних процесів і сервісів.

У зв'язку з цим активно розвивається напрямок, пов'язаний з ефективною організацією управління критичними ІТ-сервісами та процесами. Одним із сучасних підходів [59–60], що дозволяє організувати ефективне управління ІТ-сервісами, є підхід ITSM (IT Service Management, управління послугами ІТ). Даний підхід базується на концепціях інформаційного управління [61], викладених в ITIL (IT Infrastructure Library, бібліотека інфраструктури інформаційних технологій) і описують процесний підхід до надання ІТ і забезпечення їх використання. В рамках цієї концепції виділяється структурний

підрозділ – служба Service Desk, який реалізує процесний підхід до управління гарантованим рівнем підтримки ІТ-сервісів. Однак, науково-методичні аспекти в області інформаційного управління саме критичними ІТ-сервісами та процесами потребують подальшого опрацювання та розвитку. Проблеми ефективного управління бізнес-процесами, з урахуванням «концепції критичності», розглянуті в працях таких дослідників, як Я.В. Бон, Г. Кемерлінг, Д. Пондаман [59], П.Брукс [60], О.І. Шелухін [62], Т.Л. Бірюкова [63], В.К. Батоврін [64], Г.Н. Ісаєв [65–66], В.В. Репін [67]. У свою чергу, аналіз роботи систем підтримки ІТ-сервісів і процесів для критичних систем (критичних інфраструктур (КІ)) дозволяє віднести їх до класу складних організаційно-технічних систем критичного застосування або, іншими словами, до критичних інформаційних інфраструктур (далі – критична інформаційна інфраструктура, КІІ).

Разом з тим, автоматизація процесів КІІ поряд з незаперечними перевагами неминуче веде і до негативних явищ. Найбільш характерним негативним фактором виступає прогресуюча інформатизація криміналу і терористичних угруповань, що істотно загострює проблему протиборства в сфері інформаційних ресурсів [68–75].

Першочерговими об'єктами протиправних дій терористичних і кримінальних спільнот є інформаційні ресурси КІ, збиток від порушення інформаційної безпеки яких може призвести до техногенних катастроф з людськими жертвами [76–79]. Прикладом тому є жахливі терористичні акти, здійснені протягом останніх 15 років в США, Іспанії, Франції та інших регіонах земної кулі [78, 80–81].

До критичних інфраструктур [75], поряд з системами управління інфраструктурою зв'язку, фінансів, енергетики, транспорту, водопостачання та надзвичайних служб, відносяться органи державного управління, в тому числі і системи управління силових структур.

Інформаційні елементи цих структур є критично важливими сегментами інформаційної сфери держави. Особливо актуальною ця проблема стає у зв'язку

з необхідністю розробки і впровадження критичних інформаційних інфраструктур на підприємствах державної сфери [74, 75, 82–85]. Під КІІ розуміють сукупність об'єктів критичної інформаційної інфраструктури, тобто комунікаційні або технологічні системи об'єкта критичної інфраструктури, кібератака на яку безпосередньо вплине на стале функціонування такого об'єкта критичної інфраструктури [86].

В останні роки спостерігається стійка тенденція наростання потоку інформації, що циркулює в системах управління різного призначення. На сучасному етапі виникає необхідність впровадження мережевих, автоматизованих методів передачі та обробки критичної інформації [87–89]. Одним із шляхів вирішення цієї проблеми є розгортання сучасної ІТ-інфраструктури на базі сучасних мережевих технологій.

Технологія розробки ІТ-інфраструктур досить глибоко опрацьована і накопичений досвід в цій сфері може бути використаний при проектуванні ІТ-інфраструктур для критичних інфраструктур [90–91]. Однак проектування і впровадження КІІ стримується через властиву їм специфіку [74, 75, 81, 92–94], яку можна звести до трьох чинників. В якості першого чинника варто зазначити відсутність прийнятного математичного інструментарію для оцінки альтернативних проектних рішень, вибору ефективних елементів з існуючого набору пропонованих типових засобів інформаційної сфери КІІ з врахуванням особливих вимог, які до неї ставляться. Успішне проектування КІІ стає можливим при вирішенні задач представлення та обґрунтування вибору архітектури критичної інформаційної інфраструктури, оцінки і вибору ефективних елементів та компонент з існуючого набору пропонованих типових засобів. Тому актуальним є завдання створення прийнятного математичного інструментарію для оцінки ефективності вибору проектних рішень та окремих елементів архітектури критичної інформаційної інфраструктури з існуючого набору пропонованих типових засобів інформаційної сфери КІІ, та їх вибору з точки зору забезпечення заданих критеріїв проектування критичної інформаційної інфраструктури.

До другого фактору слід віднести умови експлуатації, яка здійснюється при безперервних потужних інформаційних впливах, обумовлених підвищеним інтересом до інформаційних ресурсів КІІ з боку кримінальних і терористичних угруповань [94–96].

Тут виникає проблема забезпечення високої експлуатаційної надійності функціонування КІІ в умовах інформаційної протидії, яка передбачає забезпечення заданої оперативності та достовірності передачі та прийому інформації, що циркулює в КІІ. Для вирішення цієї проблеми потрібне теоретичне і технічне обґрунтування адаптивних алгоритмів функціонування КІІ в умовах перешкод природної та організованої структури, які б дозволили відслідковувати наявність ресурсів та навантаження на елементи критичної інформаційної інфраструктури, та у разі необхідності, виконувати балансування навантаження або робити перерозподіл наявних ресурсів з метою забезпечення функціонування критичних сервісів, що визначає актуальність досліджень в напрямку вирішення проблем, обумовлених другим з чинників.

Нарешті третій чинник обумовлений забезпеченням вибору оптимальної архітектури КІІ, яка в залежності від задач проектування, а в подальшому, від реального навантаження на її елементи, повинна комплектуватися певним набором функціональних блоків, при якій буде забезпечуватися гарантована ефективність її функціонування в особливі періоди кризових ситуацій, коли інтенсивність навантаження зростає. На даний час технологія такого вибору відсутня. Вирішення даної проблеми потребує теоретичного дослідження та є ще одним з актуальних завдань дисертаційного дослідження.

Зазначені проблеми повинні розглядатися системно на основі всебічного дослідження технології забезпечення і оцінки інформаційної безпеки КІІ [95, 96], що базується на типових засобах. На основі системного підходу повинні бути розроблені теоретичні та технічні основи забезпечення експлуатаційної надійності функціонування КІІ і методика оптимізації архітектури КІІ, з метою забезпечення інваріантності до умов функціонування КІІ.

Рішення наведених вище проблем може здійснюватися за двома напрямками. Перший – вирішення проблеми забезпечення функціональної стабільності циркулюючої інформації в КІІ, на основі дослідження існуючих технологій. Вирішення цієї проблеми характеризується безліччю нетривіальних властивостей, тому питання оцінки елементів КІІ відносяться до числа складних як у науковому, так і в практичному плані проблем. Необхідно знайти такі підходи в оцінці ефективності її елементів, які системно враховували б всю безліч властивостей засобів, що входять до її складу. Одним з найбільш перспективних шляхів вирішення даної проблеми є узагальнення показників, що характеризують можливості засобів КІІ [97–101]. Разом з тим, процес узагальнення показників має ряд особливостей, і, перш за все, наявність декількох ступенів свободи в класифікації можливостей КІІ [102, 103]. Те ж саме можна сказати і про інструмент дослідження КІІ – математичні моделі процесів її проектування та вдосконалення функціональних процесів. Існування вказаних проблем дозволило в даній дисертації запропонувати новий підхід до вирішення проблеми проектування КІІ шляхом оцінки можливостей КІІ за допомогою різних критеріїв, які дозволяють спроектувати оптимальну архітектуру під кожен окрему задачу. Суть даного підходу полягає у встановленні науково обґрунтованих правил проектування з точки зору оцінки ступеня досягнення цілей набором мінімальної кількості формул. Це досягається комплексним використанням методів комп'ютерного моделювання, методів штучного інтелекту. Тобто новизна дослідження визначається необхідністю розробки методології проектування та методів вдосконалення КІІ з комплексною оцінкою їх ефективності з використанням математичних та комп'ютерних моделей, які мінімізують використовуваний математичний апарат.

Другий напрямок – вирішення проблеми забезпечення оперативності і достовірності передачі критичної інформації, а також оптимізації структури КІІ і розроблюваної системи проектування, на базі якої і розгортається КІІ.

Для вирішення вказаних проблем необхідна розробка і дослідження нових інформаційних технологій, які враховують взаємозв'язок показників критичності

систем і рівня гарантованості підтримуваних ІТ-сервісів і процесів. Вагомий вплив на дослідження проблем критичних інфраструктур надали роботи вітчизняних і зарубіжних вчених: О.Г. Корченка [100, 101], С.О.Гнатюка [102, 103], В.С. Харченка [104–106], А.Л. Станівського [107], М.А. Ястребенецького [108, 109], С. Руссо [74], М. Fusani [75], О.І.Роліка та С.Ф.Теленика [90, 91, 110–112].

Розробка математичних та комп'ютерних моделей процесів проектування та функціонування критичної інформаційної інфраструктури відноситься до числа складних проблем як в теоретичному, так і практичному плані, через те, що опис об'єкта здійснюється в динамічному режимі з урахуванням оцінки показників всіх її елементів.

З огляду на це, актуальними, в науковому та практичному аспектах, є:

- проблема пошуку нових підходів до проектування критичних інформаційних інфраструктур, які б системно враховували всю множину зазначених вище факторів, і можливості засобів, які входять до їх складу;
- проблема розробки і розвитку методів моделювання та підтримки прийняття рішень в КІІ та розробка на їх основі єдиної системи проектування, що дозволяє проектувати, аналізувати, розвивати КІІ та підтримувати гарантований рівень якості критичних ІТ-сервісів і процесів, синхронно зі змінами в ній.

Зв'язок роботи з науковими програмами, планами, темами. Тема дисертаційної роботи відповідає планам науково-дослідної та навчальної роботи кафедри автоматики та управління в технічних системах Національного технічного університету України «Київський політехнічний інститут» та науково-дослідної роботи Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова. Дисертаційна робота розпочата на кафедрі автоматики та управління в технічних системах Національного технічного університету України «Київський політехнічний інститут», завершена в Інституті проблем моделювання в енергетиці ім. Г.Є. Пухова і виконувалась в рамках наступних НДР: «Розроблення і дослідження моделей, методів та технологій проектування,

програмування і управління хмарними іт-інфраструктурами» (номер державної реєстрації № 01113U002285), «Платформа розроблення, експлуатації і розвитку критичних інформаційних інфраструктур для роботи з великими даними» (номер державної реєстрації № 0116U003801) та «Хмарна платформа розроблення і управління функціонуванням критичних інформаційних інфраструктур, що опрацьовують великі обсяги даних» (номер державної реєстрації № 0117U000537).

Метою дисертаційної роботи є підвищення ефективності процесів проектування критичної інформаційної інфраструктури шляхом розробки методів представлення процесу прийняття рішень щодо вибору та обґрунтування архітектури критичної інформаційної інфраструктури, оцінки альтернативних проектних рішень, вибору оптимальної конфігурації компонент та вдосконалення критичної інформаційної інфраструктури..

Для досягнення зазначеної мети в роботі сформульовані і вирішені такі наукові проблеми та завдання:

- виконано аналіз сучасного стану і особливостей проблеми проектування, обґрунтування рішень і вдосконалення критичних інформаційних інфраструктур та досліджено потенційні можливості сучасних фрейворків опису архітектури підприємства та підходів до обґрунтування рішень щодо архітектури при проектуванні критичних інформаційних інфраструктур;

- проведено аналіз та визначено основні критерії та показники оптимальності проектування та функціонування критичних інформаційних інфраструктур;

- досліджено і розроблено комплекс математичних та комп'ютерних моделей та методів проектування, вдосконалення і розв'язання повсякденних завдань КІІ;

- оцінено ефективність застосування розроблених методів і моделей проектування КІІ;

- розроблено єдину систему проектування КІІ.

Об'єктом дослідження є процеси проектування критичної інформаційної інфраструктури.

Предметом дослідження є методи і засоби математичного та комп'ютерного моделювання процесів представлення процесу прийняття рішень щодо вибору та обґрунтування архітектури критичної інформаційної інфраструктури, оцінки альтернативних проектних рішень, вибору оптимальної конфігурації компонент та вдосконалення критичної інформаційної інфраструктури.

Методи дослідження. Теоретичною і методологічною основою дослідження є:

- теорія складних багаторівневих ієрархічних систем;
- теорія інформації та управління;
- теорія надійності;
- теорія експертних систем;
- теорія прийняття рішень;
- теорія матриць;
- теорія ймовірностей і математична статистика;
- теорія множин, математична логіка і теорія алгоритмів;
- основи обчислювальної математики;
- основи теорії математичного моделювання;
- елементи теорії штучного інтелекту.

Обґрунтованість і достовірність отриманих результатів забезпечується коректним застосуванням апробованого і загальноприйнятого математичного апарату, ідентифікацією, верифікацією даних на основі комп'ютерного і математичного моделювання, зіставленням розрахунків на основі розроблених моделей з відомими еталонними чисельними рішеннями, перевіреними натурними експериментами, розбіжність з якими не перевищує заданого рівня достовірності, широкою апробацією отриманих результатів на Міжнародних та Всеукраїнських конференціях, публікацією в реферованих журналах ДАК

України, а також експертизою технічних рішень, підтверджених актами про впровадження та використання результатів дисертаційного дослідження.

Наукова новизна полягає в вирішенні актуальної науково-прикладної проблеми, яка полягає в удосконаленні процесів вибору і обґрунтування проектних рішень щодо критичної інформаційної інфраструктури на підставі: використання розширених UML-моделей; розроблених методів представлення процесу прийняття рішень щодо вибору та обґрунтування архітектури критичної інформаційної інфраструктури, оцінки альтернативних проектних рішень, вибору оптимальної конфігурації компонент критичної інформаційної інфраструктури; методів розвитку критичної інформаційної інфраструктури; подальшого розвитку концепції проектування критичної інформаційної інфраструктури; створення та застосування єдиної системи проектування критичної інформаційної інфраструктури. Наукова новизна одержаних результатів полягає в наступному:

1. Вперше запропоновано модель перетворення розширених UML-діаграм, які відображають процес прийняття рішення щодо вибору та обґрунтування архітектури критичної інформаційної інфраструктури, в розмічені транзиційні системи, що дозволяє використати в подальшому весь спектр методів та алгоритмів, розроблених для розмічених транзиційних систем, для дослідження проблем, пов'язаних з проектуванням критичної інформаційної інфраструктури.

2. Вперше запропоновано постановку задачі верифікації параметризованих моделей архітектурних рішень критичної інформаційної інфраструктури в термінах теорій розмічених транзиційних систем та темпоральної логіки, що дозволяє використати в подальшому весь спектр методів та алгоритмів, розроблених для цих теорій, для верифікації/генерації проектних рішень критичної інформаційної інфраструктури.

3. Вперше запропоновано метод обґрунтування проектних рішень щодо архітектури критичної інформаційної інфраструктури, в якому для порівняння альтернативних проектних рішень запропоновано застосування множини з трьох показників ризику, що враховують досяжність цілей проектного рішення,

можливість його імплементації та дотримання вимог щодо критичності, та ентропійного підходу для оцінювання їх взаємного впливу для задачі проектування критичної інформаційної інфраструктури, що дозволяє обрати найкращий варіант проектного рішення та оцінити вплив окремих проектних рішень або елементів на інші проектні рішення щодо архітектури критичної інформаційної інфраструктури або на весь дизайн архітектури в цілому.

4. Вперше запропоновано метод структурної оптимізації нейронних мереж прямого поширення, який використовує розширений набір атомарних операцій над нейронною мережею та дозволяє отримувати оптимальну для вхідних даних структуру нейронної мережі, що значно підвищує можливості адаптаційного вибору моделей нейронних мереж для розв'язання задач функціонування критичних інформаційних інфраструктур.

5. Вперше запропоновано керовану моделлю систему розподілу ресурсів критичної інформаційної інфраструктури, яка використовує методи оптимізації структури нейронних мереж прямого поширення та навантаження на елементи критичної інформаційної інфраструктури, що дозволяє в автоматичному режимі розподіляти ресурси критичної інформаційної інфраструктури з метою їх оптимального використання та задоволення потреб сервісів та компонент, що їх використовують.

6. Удосконалено метод представлення та обґрунтування архітектури критичної інформаційної інфраструктури на основі розширених UML-діаграм, який, на відміну від відомих, дозволяє фіксувати пропозиції архітектора в процесі прийняття рішення щодо вибору та обґрунтування архітектури критичної інформаційної інфраструктури, що спрощує та пришвидшує пошук архітектури критичної інформаційної інфраструктури під конкретну задачу, а також дає можливість накопичувати історію проведених міркувань та обґрунтувань при її виборі.

7. Удосконалено метод оцінки та вибору оптимальної конфігурації компонент критичної інформаційної інфраструктури, що на відміну від вже відомих використовує Марківський процес прийняття рішень за декількома

критеріями пошуку одночасно, який дозволяє оцінити та вибрати оптимальну конфігурацію компонент критичної інформаційної інфраструктури відповідно до заданих критеріїв, що полегшує та пришвидшує вибір конкретної конфігурації критичної інформаційної інфраструктури під визначені цілі та завдання її проектування.

8. Удосконалено метод розподілу ресурсів критичної інформаційної інфраструктури на базі генетичного алгоритму з чіткими параметрами, який відрізняється від вже відомих використанням модифікованої фітнес-функції, що надає можливість знайти оптимальне співвідношення між марнуванням ресурсів та забезпеченням сервісів і визначити оптимальну схему розподілу ресурсів в центрах обробки даних з врахуванням критичності окремих процесів та сервісів.

9. Набула подальшого розвитку концепція проектування критичної інформаційної інфраструктури на базі компенсаційно-декомпенсаційного підходу пошуку архітектурних рішень, яка на відміну від вже відомих, визначає фреймворк опису архітектури підприємства, відповідні критерії вибору та стратегії їх застосування, метод обґрунтування рішень щодо вибору альтернативних архітектурних рішень при проектуванні критичної інформаційної інфраструктури, що значно підвищує швидкість її проектування.

Теоретична та практична цінність роботи. Теоретична значимість роботи полягає в розробці науково-методичного апарату, який може представляти загальнонауковий інтерес для проблеми проектування критичних інформаційних інфраструктур. Теоретична частина роботи може розглядатися в якості прикладного елемента методології системного аналізу при вирішенні проблем зазначеного класу.

Практична значимість роботи полягає в розробці ефективних комп'ютерних методів проектування та вдосконалення архітектури критичної інформаційної інфраструктури, що дозволяють вирішувати проблему автоматизованого проектування КІІ відповідно до поставлених цілей та з забезпеченням необхідного рівня функціональної надійності її експлуатації.

Теоретичні дослідження і наукові результати роботи доведені до інженерних рішень у вигляді реалізованих в інформаційній технології методик, моделей, алгоритмів, придатних для практичного використання при аналізі, синтезі й розвитку КІІ. Запропоновані методи, моделі та алгоритми дозволяють підвищити ефективність функціонування існуючих КІІ або побудувати нові ефективні КІІ.

Реалізація наукових результатів. Розроблений науково-методичний апарат і технічні рішення використовувалися при проектуванні та побудові єдиної інформаційної системи МВС України як критичної інформаційної інфраструктури органів системи МВС.

Використання результатів дисертаційних досліджень підтверджено відповідними актами впровадження (МВС України, ДП «ІНФОТЕХ», ТОВ «Оллі Транс», ТОВ «І-Хаб»).

Особистий внесок здобувача.

Всі положення, які виносяться на захист, належать особисто автору. Частина з них наведена в одноосібних наукових працях [2–12].

В роботах, які опубліковано в співавторстві, особисто здобувачу належать: в [1] – порівняльний аналіз методів обґрунтування архітектурних рішень; в [13] – порівняльний аналіз ФОАП; в [14] – принципи побудови критичної інформаційної інфраструктури міністерства; в [15] – метод структурної оптимізації нейронної мережі; в [16] – метод структурної оптимізації нейронної мережі; в [17] – метод структурної оптимізації нейронної мережі; в [18] – метод структурної оптимізації нейронної мережі; в [19] – підхід до оцінювання ризику в критичних інформаційних інфраструктурах; в [20] – модель системи управління ризиком критичної інформаційної інфраструктури; в [21] – підхід до оцінювання ризику в критичних інформаційних інфраструктурах; в [22] – модель розподілу ресурсів критичної інформаційної інфраструктури; в [23] – метод оптимізації навантаження критичної інформаційної інфраструктури; в [24] – метод інтеграції для розподілених систем при реалізації проекту критичної інформаційної інфраструктури; в [25] – підхід до проектування систем

управління безпекою інформації; в [26] – аналіз методів верифікації параметричних моделей, в [27] – аналіз ФОАП в рамках створення системи управління безпекою інформації, в [28] – метод структурної оптимізації нейронної мережі; в [29] – метод оптимізації навантаження критичної інформаційної інфраструктури; в [30] – порівняльний аналіз фреймворків для реалізації нейронних мереж.

Апробація результатів наукових досліджень. Результати досліджень доповідалися на наукових семінарах, конференціях і симпозіумах:

- 2014 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom) (Одеса, 2014 р.) [31];
- International Scientific-Practical Conference “Problems and Prospects of Intergration of Science and Technology” (Лондон, 2015 р.) [32];
- II міжнародна науково-практична конференція «Інформаційні технології та взаємодії» (Київ, 2015 р.) [33];
- II міжнародна науково-практична конференція «Актуальні проблеми розвитку науки і техніки» (Київ, 2015 р.) [34];
- Міжнародна наукова конференція ІТБ-2015 (Київ, 2015 р.) [35];
- V міжнародна науково-технічна конференція «Проблеми інформатизації» (Київ, 2015 р.) [36];
- The Congress on Information Technology, Computational and Experimental Physics 2015 (CITCEP 2015) (Краков, 2015 р.) [37];
- Second International Scientific-Practical Conference “Problems of Infocommunications. Science and Technology” (IEEE PIC S&T’ 2015) (Харьків, 2015 р.) [38];
- Summer Infocom 2016: 2-а Міжнародна науково-практична конференція (Київ, 2016 р.) [39];
- V Міжнародна науково-технічна конференція «Захист інформації і безпека інформаційних систем» (Львів, 2016 р.) [40];
- Міжнародна науково-практична конференція «Безпека інформації в інформаційно-телекомунікаційних системах» (Київ, 2016 р.) [41];

- V Міжнародна науково-практична конференція «Кібербезпека інформаційних технологій /// Фізико-технологічні проблеми передавання, оброблення та зберігання інформації в інфокомунікаційних системах» (Чернівці, 2016 р.) [42];
- II Міжнародна науково-практична конференція «Актуальні проблеми моделювання ризиків і загроз виникнення надзвичайних ситуацій на об'єктах критичної інфраструктури» (Київ, 2016 р.) [43];
- 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS' 2017) (Бухарест, 2017 р.) [44];
- V заочная научная конференция «Фундаментальные и прикладные исследования в современной науке» (Харьков, 2017 р.) [45];
- VI Міжнародна науково-практична конференція (I Міжнародний симпозіум «Практичне застосування нелінійних динамічних систем в інфокомунікаціях», PREDT-2017) (Чернівці, 2017 р.) [46];
- III міжнародна науково-практична конференція «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації» (Київ, 2017 р.) [47];
- 14th IEEE International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET2018) (Lviv-Slavske, 2018 р.) [48];
- VII Міжнародна науково-практична конференція «Фізико-технологічні проблеми передавання, оброблення та зберігання інформації в інфокомунікаційних системах (PREDT-2018)» (Чернівці, 2018 р.) [49];
- XI Міжнародна науково-практична конференція «ІНТЕРНЕТ-ОСВІТА-НАУКА-2018» (Вінниця, 2018 р.) [50];
- Summer InfoCom 2018: VI Міжнародна науково-практична конференція з інформаційних систем та технологій (Київ, 2018 р.) [51].

Публікації результатів наукових досліджень. Основні результати наукових досліджень опубліковані в 51 науковій праці: 2 монографіях, 28 наукових статтях в фахових виданнях, що входять до науково-метричних баз

даних (SCOPUS – 2) , 21 публікація у працях і матеріалах наукових конференцій (SCOPUS – 4). 11 публікацій підготовлено одноосібно, англійською мовою – 12.

Структура та обсяг роботи. Дисертаційна робота складається зі списку використовуваних скорочень, вступу, 6 розділів, висновків, списку використаних джерел, що містить 286 найменувань та 4 додатки. Загальний обсяг дисертації складає 370 сторінок. Основний зміст роботи викладено на 315 сторінках (з них п'ять повністю зайнято ілюстраціями). Дисертація містить 108 рисунків, 31 таблицю.

РОЗДІЛ 1

АНАЛІЗ ПРОБЛЕМ, ТЕНДЕНЦІЙ І ПЕРСПЕКТИВ РОЗВИТКУ ЕФЕКТИВНОГО ПРОЕКТУВАННЯ КРИТИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ

1.1 Критична інформаційна інфраструктура як об'єкт дослідження

На сьогодні створення критичних інформаційних інфраструктур є невід'ємною складовою розвитку ключових галузей, які є життєво необхідними для забезпечення безпеки та функціонування соціуму.

Існування критичної інформаційної інфраструктури тісно пов'язане з поняттям власне критичної інфраструктури – інфраструктури, що є життєво важливою для держави, відмова або знищення якої може суттєво негативно вплинути на національну безпеку.

У проекті закону [113] наведено альтернативне визначення КІІ як сукупності інформаційно-телекомунікаційних систем державного та приватного сектору, які забезпечують функціонування та безпеку стратегічних інститутів, систем і об'єктів держави (органів центрального та місцевого управління, систем управління енергетикою, транспортом, зв'язком, банківським сектором, підприємств, під час діяльності яких використовуються та/або виробляються небезпечні речовини тощо) і безпеку громадян (системи управління правоохоронних структур і оборонного сектору тощо), несанкціоноване втручання в роботу яких може загрожувати економічній, екологічній, соціальній та іншим видам безпеки або завдати шкоди міжнародному іміджу держави. До критичних інфраструктур відносяться: фінанси держави, енергетичний сектор держави, харчова промисловість, медицина, виробництво, транспортна система, водопостачання, державне управління та ін. Критична інформаційна інфраструктура повинна:

- забезпечувати функціонування екологічно небезпечних та соціально значимих виробництв та технологічних процесів, порушення штатного режиму яких може призвести до надзвичайної ситуації техногенного характеру;
- виконувати функції інформаційної системи, порушення (зупинення) функціонування якої може призвести до негативних наслідків в політичній, економічній, соціальній, інформаційній, екологічній та інших галузях;
- забезпечувати надання значного об'єму інформаційних послуг, часткове або повне зупинення яких може призвести до значних негативних наслідків для національної безпеки в багатьох галузях.

На цей час Україна активно веде розробку таких глобальних комплексів управління і деякі галузі вже мають розвинуті ІТ-інфраструктури, наприклад, диспетчерські системи управління транспортом, об'єктами енергетики.

Швидкий розвиток ІТ і телекомунікацій, їх широке впровадження в процеси управління створили ситуацію, коли сам процес надання інформаційних послуг стає об'єктом управління. Дійсно, великий вибір різноманітних сервісів та ресурсів в умовах використання інформаційно-телекомунікаційних мереж, що історично склалися, перетворює підтримку бізнес-процесів підприємств з критичною інфраструктурою в часі і просторі в режимі реального часу у досить складну проблему.

Основні постачальники комп'ютерного та телекомунікаційного обладнання і ІТ вийшли на ринок з рішеннями, призначеними для розв'язання цієї проблеми. Остання розробка, яка досить детально описує проблематику та майже повністю перекриває попередні розробки – COBIT [114]. Всі розробки такого класу є методологічним апаратом, який потрібно використовувати для створення конкретної ІТ-інфраструктури. Специфіка життя в окремо взятій країні також накладає свої обмеження на створення тієї чи іншої критичної інформаційної інфраструктури.

Натепер в нашій країні вже ведеться робота в напрямку розвитку проблематики розробки критичних інформаційних інфраструктур. Декілька колективів авторів вже почали розглядати деякі аспекти створення критичних

інформаційних інфраструктур. Наприклад, в роботі [115] автори досліджують питання забезпечення сервісного обслуговування критичних інформаційних інфраструктур, в [116] розглядається синтез параметрів для критичних ІТ-інфраструктур, в [117] – проблеми моніторингу критичних інформаційних інфраструктур.

Основна проблема розглянутих робіт – відсутність узагальненої структури критичної інформаційної інфраструктури, вимог до надійності, якості та безпеки їх функціонування.

1.1.1 Аналіз проблем управління критичною інформаційною інфраструктурою в Україні

Аналіз стану управління критичною інформаційною інфраструктурою в Україні вказує на наявність наступних проблем:

- відсутність зведеного набору термінології щодо критичних інформаційних інфраструктур – на цей час існує досить обмежена термінологічна база, прийнята відповідними законодавчими актами [86];
- відсутність переліку об'єктів з критичною інфраструктурою, хоча процедура формування такого переліку визначена [118];
- слабкий розвиток інформаційної складової самих критичних інфраструктур (тобто, об'єктів, які можна інтуїтивно віднести до критичних інфраструктур);
- відсутність узгодженості пріоритетів бізнес-рівня та ІТ. Дуже часто ці рівні визначають критичними зовсім різні речі;
- відсутність інтеграції процесів управління критичною інформаційною інфраструктурою;
- відсутність конкретних математичних моделей, алгоритмів, які можна було б використати для управління критичною інформаційною інфраструктурою;

- направленість досліджень на методи створення, організації та управління ІТ-підрозділів в цілому.

1.1.2 Вимоги до функціонування критичної інформаційної інфраструктури

Забезпечення безпеки критичної інформаційної інфраструктури реалізується через виконання вимог до її створення та експлуатації, включаючи також комплекс заходів зі створення системи безпеки критичної інформаційної інфраструктури правового, організаційного та технічного характеру. В рамках створення системи безпеки розробляється та впроваджується система моніторингу інформаційної безпеки критичної інформаційної інфраструктури. Ця система призначена для забезпечення вибору, контролю та актуалізації заходів та засобів управління інформаційною безпекою критичної інформаційної інфраструктури на всіх стадіях її життєвого циклу, а також в умовах постійного виникнення загроз безпеки.

Надійність функціонування критичної інформаційної інфраструктури забезпечується вибором таких методів та способів проектування, розробки, створення, введення в експлуатацію, безпосередньо експлуатації, модернізації, виведення з експлуатації, що дозволяють виключити або знизити до прийняттого рівня збитки власника критичної інформаційної інфраструктури в результаті порушення її штатного функціонування в нормальних умовах та умовах реалізації загроз безпеки [14].

Заходи щодо забезпечення надійності функціонування критичної інформаційної інфраструктури повинні передбачатися на всіх етапах її життєвого циклу. Для забезпечення надійності функціонування критичної інформаційної інфраструктури є важливим дотримання таких вимог:

- резервування апаратних та програмних засобів критичної інфраструктури на рівні апаратних та програмних засобів, каналів зв'язку та передачі даних, а також засобів захисту;

- забезпечення допустимого часу простою критичної інформаційної інфраструктури при перемиканні на резервні засоби;
- виключення можливості запуску та функціонування критичної інформаційної інфраструктури, оминаючи систему безпеки.

При розгляді питань віртуалізації ресурсів критичної інформаційної інфраструктури [89, 92] виникає ряд додаткових обмежень.

При створенні нового критичного процесу або сервісу потрібно враховувати наступне:

1. Нова репліка критичного процесу/сервісу розміщується на віртуальну машину сервера з найменшою середньою завантаженістю дисків.
2. Число нових створюваних критичних процесів/сервісів на кожній віртуальній машині серверів є обмеженим.
3. Розподіл віртуальних машин відбувається між серверами в різних стійках.
4. Як тільки число реплік падає нижче встановлюваної користувачем величини, потрібно знову виконати репліку критичного процесу або сервісу, при цьому вибирається процес або сервіс з найбільшим пріоритетом і копіюється з однієї з реплік серверу, який є найбільш забезпеченим. Створення реплік постійно балансується.

1.2 Аналіз підходів проектування архітектури підприємства з критичною інфраструктурою

Одна з задач цього дослідження полягає в систематичному аналізі та порівнянні фреймворків опису архітектури підприємства (ФОАП) з використанням загальних характеристик архітектури або елементів для використання в якості основи для проектування КІІ. Таким чином, описується модель розуміння архітектури, що пояснює певну неоднозначність. В ході аналізу структури архітектури виявлено деякі недоліки, для усунення яких запропоновано декілька шляхів.

Дослідники запропонували різні визначення та пояснення архітектури. У статті [119] автори припускають, що архітектура стосується питань, що виходять за межі алгоритмів і структур обчислення даних. Автори в [120] відрізняють архітектуру від дизайну, вважаючи, що архітектура пов'язана з вибором архітектурних елементів, їх взаємодією та їх обмеженнями, а дизайн пов'язаний з модульністю і деталізованими інтерфейсами елементів дизайну. Монро та інш. в [121] припустили, що архітектура не стосується деталей реалізації. Визначення архітектури IEEE стверджує, що вона є «фундаментальною організацією системи, втіленою в її складових, їх взаємозв'язками один з одним і навколишнім середовищем, і принципами, що регулюють її дизайн і еволюцію» [122]. Була також спроба формально виокремити архітектурну діяльність від проектної діяльності [123].

Open Group стверджує, що визначення архітектури полягає у розумінні всіх різних елементів, і того, як ці елементи пов'язані один з одним [124]. США, однак, визначає архітектуру як стратегічну базу, яка об'єднує стратегічні цілі, бізнес-вимоги та технологічні рішення [125]. В [126] автор розглядає архітектуру як програму управління, а також метод документації, які разом можуть дати інтегрований погляд на стратегії підприємства, бізнес-послуги, інформаційні потоки та ресурси. Тим не менш, більшість дослідників сходяться на думці, що архітектура почалась з Zachman та її ФОАП Zachman Framework. Цей фреймворк показує, як інформаційні системи вписуються в організацію, шляхом аналізу відповідей на шість питань - що, як, де, хто, коли і чому [127]. Архітектура виявляє всі компоненти в організації і є полегшувачем для узгодження ІТ і бізнес-цілей, які повинні включати такі пов'язані з бізнесом питання, як організаційні цілі, бізнес-процеси, продуктивність. Якщо в організації відсутнє таке узгодження, то для неї більш складною є адаптація до змін у бізнес-стратегії [128]. Для кращого розуміння того, що таке архітектура, зазвичай використовується візуалізація, показана на рис. 1.1, яка складається з 4 основних організаційних рівнів [120–122, 129]. Бізнес-рівень (Business Layer) представляє стратегічні цілі, які будуть керувати ІТ-рішеннями, ролями бізнесу, бізнес-

функціями, бізнес-процесами та потоками послуг, об'єктами бізнес-інформації. Архітектура даних (Data Architecture) показує дані, які повинні бути зібрані, організовані і пізніше розподілені. Прикладний рівень (Application Layer) описує людей і системи, які знаходяться в організації, застосунки, компоненти програмного забезпечення та корпоративні послуги. Технологічний рівень (Technology Layer) представляє технічну інфраструктуру, що складається з систем, мережевих пристроїв.

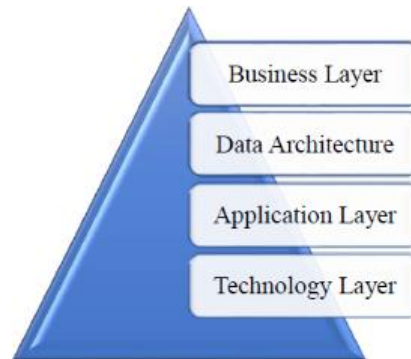


Рис. 1.1. Шари архітектури підприємства [130]

Кожен з рівнів описує або модель, яка вже існує, такою, якою вона є, або такою, якою буде існувати в майбутньому. Крім того, важливо пам'ятати, що всі рівні пов'язані один з одним. Наприклад, ІТ-система моделюється на прикладному рівні як застосунок, що надає послуги, тоді як на технологічному рівні це набір програмних компонентів, що дозволяють працювати на сервісному рівні. Для створення та майбутнього використання архітектури необхідні відповідні ФОАП.

На сьогоднішній день існує більше сотні ФОАП, які застосовуються для оборонної промисловості, уряду, відкритих джерел, а також приватні фреймворки тощо.

ФОАП допомагають покращити розуміння предмета, надаючи систематичні підходи до розвитку архітектури, але багато аспектів архітектури залишаються неоднозначними. Неоднозначність полягає в наступному :

- масштаб архітектури – чи повинна архітектура охоплювати лише програмні компоненти або включати інші аспекти розвитку критичної інформаційної інфраструктури?

– роль архітектора – роль архітектора в життєвому циклі розвитку КІІ часто незрозуміла.

– результати – яким має бути результат роботи з архітектурою? Результати можуть варіюватися від документів бізнес-функцій до детальних проектів критичних інформаційних інфраструктур.

– архітектурна діяльність – передбачає проектування і моделювання, але потребує відповідей на питання: який рівень деталізації архітектури і коли починається детальна проектна робота?

– перевірка архітектури – на якому рівні потрібно вимірювати, перевіряти чи підтверджувати результати щодо архітектурного рішення?

– участь – яка система, якого розміру і складності потребує архітектура? Чи можуть системи різних розмірів і складності вимагати однакових архітектурних рішень?

– рівень архітектури – який зв'язок між архітектурою підприємства і архітектурою КІІ?

1.2.1 Базис аналізу архітектури складної системи

Для архітектури складної системи, такої як критична інформаційна інфраструктура, наведені міркування в декількох вимірах, таких як бізнес-вимоги, технічні вимоги, критерії, поточна архітектура і майбутня архітектура тощо [131]. Виміри або входять до процесу створення архітектури взаємопов'язані і не можуть розглядатися окремо. Наприклад, архітектура системи з гнучкістю і мобільністю може впливати на продуктивність, вартість і графіку. Ключовим результатом процесу є створення моделі для архітектурних проектів. Модель розглядає комплексні виміри для досягнення збалансованих компромісів і мінімізації ризиків досягнення цілей КІІ. Ризики, які виникають під час побудови або розвитку системи, можуть бути в багатьох сферах. Вони представляють невизначеність щодо досягнення цілей КІІ. Процес проектування архітектури усуває невизначеність внаслідок моделювання та специфікації КІІ до тих пір,

поки вирішена проблема не стане добре зрозумілою, а отримане рішення не буде мати високу ймовірність досягнення поставлених цілей.

У даному розділі пропонується проаналізувати архітектуру з різних точок зору. Спочатку архітектура аналізується з точки зору її цілей, вхідних даних та результатів, як у роботі [132].

Моделювання архітектури зазвичай використовує абстракції високого рівня, які називаються представленнями. ФОАП використовують визначені точки зору для створення представлень, які відображають різні розрізи моделі критичної інформаційної інфраструктури. Загальними точками зору є бізнес-архітектура, інформаційна архітектура, архітектура програмного забезпечення та технічна архітектура. Конкретні фреймворки, що вивчаються, можуть мати основні цілі зосередитися на розподілених системах, архітектурі підприємства або галузевих системах. Після вивчення різних архітектурних фреймворків і рекомендованої практики для опису архітектури програмно-інтенсивної системи IEEE [122], в цій роботі висунуто набір спільних цілей для архітектури, які пропонується використати і для КП. Ці цілі не залежать від галузі, стилю архітектури та розмірів системи:

- процес створення архітектури – використовується чітко визначений процес для побудови архітектури;
- визначення та розуміння архітектури – використовуються стандартні терміни, принципи та керівні принципи для послідовного застосування фреймворків для передачі інформації про архітектуру зацікавленим сторонам;
- архітектурний аналіз – надає набір точок зору для управління збором та аналізом інформації для вибору варіантів архітектури;
- підтримка еволюції архітектури – використовуються процеси і механізми, що підтримують еволюцію систем;
- моделі архітектури – забезпечують узгоджені стандарти документування архітектурних специфікацій для планування, управління, комунікації та виконання діяльності, пов'язаної з розробкою системи;

– обґрунтування дизайну – документування причин дизайнерських рішень для перевірки у певному вигляді (наприклад, «обрано рішення з такої-то причини») [133];

– проектні компроміси – обирається єдиний варіант дизайну з декількох варіантів шляхом вирішення багатовимірних конфліктуючих вимог;

– стандартизація – забезпечення збереження розробок і архітектурних стандартів;

– перевірка архітектури – надає достатню інформацію або пояснення щодо архітектури для її перегляду та перевірки;

– база знань архітектури – наявність репозиторія для збереження проектів архітектури та їх обґрунтувань.

Входи представляють інформацію, яка розглядається при архітектурному моделюванні. Виходи представляють інформацію щодо фактичних та очікуваних результатів. Типовими входами до процесу створення архітектури є наступні:

– технологічні ресурси – стратегічний напрямок розроблення архітектури, який включає технологічні платформи, майбутню архітектуру, системну сумісність та нові стандарти технологій;

– бізнес-драйвери – бізнес-цілі, напрямки, принципи, стратегії та пріоритети;

– оточуюче середовище інформаційної системи – бюджет, графік, технічні обмеження, ресурси та експертиза, організаційна структура, інші обмеження, база знань підприємства;

– бізнес-вимоги – вимоги користувачів, функціональні вимоги, вимоги до даних та інші вимоги, що стосуються системи;

– нефункціональні вимоги – деякі з цих вимог також називаються атрибутами якості (QA) або якістю послуг (QoS). Ці вимоги включають доступність, надійність, масштабованість, безпеку, продуктивність, взаємодію, змінюваність, ремонтпридатність, зручність і керованість;

– поточна архітектура – поточні стандарти та інфраструктура.

Використання іншого ФОАП для розробки КІІ призведе до схожих, але дещо інших результатів, оскільки кожен фреймворк має різні архітектурні представлення. З іншого боку, використання одного і того ж автофокуса для розробки КІІ різної складності вимагатиме різних типів вхідних даних і даватиме різні результати. Результати ФОАП відображають цілі, які мають бути досягнуті. З метою збереження нейтральності, потрібно уникати використання специфічної термінології для представлення результатів. Результатами є:

- бізнес-модель (Business Model) – описує бізнес-моделі, бізнес-вимоги, бізнес-процеси, ролі системи, правила політики;
 - інформаційна модель (Information Model) – містить модель даних, методи перетворення даних та інтерфейс даних;
 - модель системи (System Model) – моделі основних компонентів системи.
- З метою отримання моделі архітектури системи, мають бути прийняті основні компроміси та дизайнерські рішення. Також повинні бути враховані майбутні удосконалення системи;
- модель конфігурації програмного забезпечення (Software Configuration Model) – описує, як програмне забезпечення упаковується, зберігається, конфігурується, керується та спільно використовується;
 - модель обчислення (Computation Model) – містить функціональний опис системи, системний потік процесу, системні операції, компоненти програмного забезпечення та взаємодії;
 - програмне забезпечення для обробки моделі (Model Processing Software)
- описує, як структуруються процеси програмного забезпечення, потоки програмного забезпечення та середовище виконання;
- модель впровадження (Implementation Model) – описує фізичну структуру системи, таку як операційне середовище, апаратні компоненти та компоненти мережі, а також процеси реалізації моделей, такі як інсталяція, розгортання, конфігурація та управління;

- платформи (Platforms) – описують програмне забезпечення платформи, наприклад операційні системи, апаратні засоби та мережеві компоненти, протоколи та стандарти;

- перехідний дизайн (Transitional Design) – описує проекти та плани для підтримки переходу та еволюції системи;

- дизайн нефункціональних вимог (Non-functional Requirements Design) – моделює структуру системи для відображення дизайну нефункціональних вимог;

- обґрунтування дизайну (Design Rationale) – документує причини дизайну, заснованого на аналізі та компромісах, що включають декілька вимірів входів.

По-друге, в даному розділі кожен ФОАП аналізується у термінах концепцій, моделювання та процесу, описаних в праці [134]:

- концепції – концепції архітектури, важливі для підприємств загалом і для реалізації стратегії: визначення архітектури, співвідношення між бізнесом і ІТ, важливість репозиторію, асоціації та взаємодії між артефактами та їх стратегією реалізації, управління, ролі та процеси архітектури [135].

- моделювання – оскільки концепції архітектури забезпечують основу для методології впровадження (МВ), то моделювання портретного дизайну стосовно цих понять як правило, є основною частиною будь-якої МВ. Типове моделювання складається з наступних основних компонентів: нотації, синтаксису і семантики. Моделювання різних ракурсів підприємства є важливою частиною моделювання, яку необхідно використовувати в МВ. Отже, за допомогою відповідного моделювання МВ може значно знизити складність поточної та бажаної архітектури та плану переходу [136].

- процес – як згадувалося вище, моделювання розглядається як обов'язкова частина будь-якої МВ. Однак МВ визначає набір процесів і частин, що є частинами життєвого циклу архітектури. Успішна МВ повинна охоплювати наступні етапи: моделювання підприємства, аналіз існуючої архітектури, аналіз очікуваної архітектури, управління та надання детального дизайну проектів, що

описують контрольований план переходу і впровадження. МВ, що охоплює всі частини розробки архітектури з урахуванням концепцій архітектури є послідовною і повною методологією [135].

По-третє, кожний ФОАП аналізується відносно наступних п'яти вимог:

- Організаційна сумісність (Organisational Interoperability) – пов'язана з гармонізацією бізнес-процесів та інформаційних технологій, яка охоплює як між-, так і внутрішньо-організаційні межі.

- Спільна інфраструктура та сумісність (Common infrastructure and interoperability) - здатність організацій обмінюватися інформацією та знаннями в межах та за їх межами. Основою для ефективної сумісності є стандартизована спільна інфраструктура.

- Технічна сумісність (Technical Interoperability) – відноситься до технологічних аспектів підключення комп'ютерних систем для обміну інформацією або використання функціональних можливостей.

- Адаптивність (Agility) - здатність організації керувати змінами, що є суттєвою характеристикою для виживання підприємств, які змушені працювати в динамічних умовах, де зміни є постійними.

- Повторне використання (Reusability) – модулі багаторазового використання скорочують час і витрати на реалізацію, збільшують ймовірність модифікації, коли потрібна зміна в реалізації.

І врешті-решт, для останньої оцінки обрано наступні шість вимог:

- Зусилля, необхідні для розробки та підтримки – складність інструментів і методів моделювання, прийнятих в контексті ФОАП - чим легше моделювати і пізніше підтримувати архітектуру, тим краще.

- Сервіс-орієнтованість – можливість застосування парадигми продуктивності щодо того, що забезпечує реалізацію розділів оперативних можливостей для окремих завдань.

- Оцінка та управління – здатність фреймворків проводити оцінку ефективності та зрілості різних агентів при використанні відповідної архітектури

або процесу управління, щоб гарантувати, що інвестиції в ІТ тісно пов'язані з бізнес-цілями.

- Еталонні моделі – можливість описати все, використовуючи одну мову.

- Документація – наявність документації (так як політика та зацікавлені сторони постійно змінюються, документація з розробки архітектури важлива і повинна враховуватися для поширення знань та обміну досвідом).

- Економічна ефективність – додаткові грошові інвестиції (чи є система вільною або потребує додаткових грошових інвестицій).

Всі ці підходи (точки зору) мають певні точки перетину, але в цілому дають нам повний огляд недоліків та переваг кожного ФОАП.

1.2.2 Компаративний аналіз ФОАП

У цьому розділі представлений компаративний аналіз ФОАП. Слід зазначити, що існує велика кількість авторів, що досліджують питання опису та аналізу архітектури, а також питання, пов'язані з впровадженням відповідної архітектури в різних місцях [137–141]. В той же час, не виявлено жодного дослідження, присвяченого питанням щодо проектування архітектури для українських державних установ або компаній. Управління, засноване на процесах, робить організації більш прозорими, дозволяючи розвивати ефективну систему вимірювання продуктивності та підвищувати можливості відстеження витрат і ресурсів.

Як уже згадувалося, для аналізу обрані наступні ФОАП: The Zachman Framework [127, 140], The Open Group Architecture Framework [141–143], The Federal Enterprise Architecture Framework [141, 144, 145], The Department Of Defense Architecture Framework [146–148], The British Ministry of Defense Architecture Framework [149–154], The NATO Architecture Framework [155, 156], The Treasury Enterprise Architecture Framework [157, 158], The Gartner Enterprise Architecture Framework [158], Open Distributed Processing – Reference Model (RM-

ODP) [159] та Service-Oriented Architecture [160–163]. Детальний опис обраних ФОАП наведено в [13].

Оскільки аналіз для вибору найкращого ФОАП як основи для проектування критичної інформаційної інфраструктури не проводився, для визначення критеріїв аналізу першим кроком був пошук пов'язаних робіт, які б допомогли ці критерії визначити. Для загального розуміння визначення ФОАП, яке використовується по всьому світу, використано запит «архітектура підприємства» для пошуку в Google і перевірено 75 перших сторінок. Найбільш широко використовуваним ФОАП виявився TOGAF (Швейцарія, Південна Корея, Південна Африка), FEAF (Сполучені Штати, Австралія, Сінгапур). FEAF. Популярність використання ФОАП наведено в табл. 1.1.

Табл. 1.1. Популярність ФОАП

ФОАП	Популярність, %
Zachman Framework	25%
TOGAF	11%
FEAF	9%
DoDAF	11%
MODAF	2%
NAF	1%
TEAF	1%
GEAF	3%
RM-ODP	N / A
SOA	15%

При проведенні огляду літератури вирішено, що деякі ФОАП будуть виключені з подальшого порівняння через їх відносну не популярність. Таким чином, для подальшого порівняння були обрані наступні ФОАП: ZF, TOGAF, FEAF, DoDAF, MODAF, NAF, TEAF, GEAF, RM-ODP, SOA.

У [164] порівнюються п'ять фреймворків – Zachman, DoDAF, FEAF, TEAF, TOGAF – за допомогою трьох різних критеріїв порівняння – за представленнями/перспективами, за абстракціями, фазами розробки програмного забезпечення (SDLC). Відповідно до праці [164], фреймворк Захмана є найповнішою структурою серед досліджуваних. Він використовує декілька точок зору, пов'язаних з різними аспектами. Більшість фреймворків представляють лише невелику кількість точок зору та аспектів. В іншій статті

[165] порівняння ФОАП не проводилось, але натомість були представлені критерії оцінки впровадження, які можуть бути корисними для аналізу в майбутньому після розробки архітектури. Стаття [141] фокусується на оцінці ФОАП для електронного уряду, орієнтованого на покращення інтероперабельності та інтеграції, зниження витрат, покращення управління змінами та ризиками, оцінці узгодженості бізнес-ІТ.

В даній роботі використовуються 4 різні групи основних елементів для аналізу архітектури [13]. Показано, що всі архітектурні фреймворки підтримують мету розвитку архітектури. Зокрема, RM-ODP має особливу спрямованість на розробку архітектури програмного забезпечення. TOGAF, DoDAF і FEAF більш спрямовані на питання щодо архітектури підприємства, такі як архітектурне планування, еволюція та системна сумісність. Вони використовують різні точки зору для моделювання архітектури підприємства і мають різні ступені специфічності для цих точок зору.

Основною метою ФОАП є спрощення визначення, загального розуміння та стандартизація практик опису та створення архітектури. Їх довгостроковими цілями є підтримка стратегічного планування щодо архітектури, використання бази знань щодо архітектурних рішень для підтримки її еволюції. Моделі бізнесу та архітектури, створені за допомогою цих фреймворків, описують напрямки розвитку архітектури, архітектури “to-be” та стратегії переходу від поточної архітектури до майбутньої архітектури для підприємства.

Хоча розроблення архітектури передбачає її проектування, мета створення архітектурного рішення включає не тільки опис деталей самого проекту. Заходи щодо розробки і проектування зосереджені в області, де архітектура пов’язана з структурою, моделюванням і плануванням КІІ на більш високому рівні. У жодному з досліджених ФОАП не існує керівних принципів щодо моменту або критерію, коли діяльність по створення архітектурного рішення перетворюється на наддетальний опис проекту архітектури. Автор вважає, що рівень деталізації проекту архітектури повинен базуватися на рівні невизначеності архітектури. Якщо невизначеність архітектури для точного моделювання КІІ є відносно

невеликою, тоді проектування може здійснюватися на етапі детального проектування. З іншого боку, якщо невизначеність висока, то для зниження ризику потрібно більше заходів архітектурної діяльності.

Оцінка щодо визначених цілей. В цьому розділі наведено порівняння та аналіз десяти ФОАП. Оскільки проектування архітектури фокусується на різних точках зору (перспективах) відносно того, як модель архітектури повинна бути представлена, їх можна порівняти лише тоді, коли ФОАП охарактеризовані основними елементами, такими як їхні цілі, вхідні дані та результати.

У табл. 1.2 наведено огляд та порівняння фреймворків за визначеними цілями. Якщо фреймворк явно підтримує відповідний елемент, то отримує 2 бали. Якщо фреймворк не підтримує елемент або про цей елемент не згадується у документації, то він отримує 0 балів. У разі, якщо фреймворк частково підтримує або відсутня підтримка відповідного елемента, він отримує 1 бал. Ступінь підтримки та інтерпретації кожного елемента у кожного ФОАП може відрізнятися, навіть якщо вони мають однакові оцінки.

Табл. 1.2. Оцінка ФОАП відповідно до визначень цілей

	Z F	TOGA F	FEA F	DoDA F	MODA F	NA F	TEA F	GEA F	RM - OD P	SO A
Цілі										
Визначення та розуміння архітектури	1	2	2	2	2	2	2	2	2	2
Процес архітектури	0	2	2	2	2	2	2	2	0	2
Підтримка еволюції архітектури	0	2	2	2	2	2	1	2	1	2
Аналіз архітектури	2	2	2	2	2	2	2	2	2	2
Моделі архітектури	2	2	2	2	2	2	2	2	2	2
Проектні компроміси	1	1	1	2	2	2	1	2	1	2
Обґрунтування дизайну	1	2	1	1	1	1	1	2	2	2
Стандартизація	0	2	1	2	2	2	1	2	2	2
База знань архітектури	0	2	2	2	2	2	1	2	2	2

	Z F	TOGA F	FEA F	DoDA F	MODA F	NA F	TEA F	GEA F	RM - OD P	SO A
Перевірка архітектури	0	2	0	0	0	1	0	2	1	1
Входи										
Бізнес-драйвери	1	2	2	2	2	2	2	2	1	2
Технологічні входи	0	2	2	2	2	2	1	2	1	2
Вимоги до бізнесу	2	2	2	2	2	2	2	2	2	2
Навколишнє середовище інформаційної системи	1	2	2	2	2	2	2	2	2	2
Поточна архітектура	1	2	2	2	2	2	2	2	2	2
Не функціональні вимоги	1	2	1	1	1	1	0	1	2	2
Результати										
Бізнес-модель	2	2	2	2	2	2	2	2	2	2
Модель системи	2	2	2	2	2	2	2	2	2	2
Інформаційна модель	2	2	2	2	2	2	2	2	2	2
Модель обчислення	2	2	2	2	2	2	2	2	2	2
Модель конфігурації програмного забезпечення	0	2	0	0	0	0	0	1	1	2
Модель обробки програмного забезпечення	2	2	2	2	2	2	2	2	2	2
Модель впровадження	1	2	1	2	2	2	2	2	2	2
Платформи	2	2	2	2	2	2	2	2	2	2
Дизайн нефункціональних вимог	1	2	1	1	1	1	1	1	2	2
Перехідний дизайн	0	2	2	2	2	2	1	2	0	2
Обґрунтування дизайну	0	1	0	1	1	1	0	1	1	1
Усього	27	52	42	46	46	47	38	50	42	52

Оцінка щодо концептуальних визначень. Цей розділ описує структуру для оцінки обраної архітектури за другим набором критеріїв. Вона включає в себе набір критеріїв, які стосуються як загальних атрибутів архітектури, так і функцій,

які однозначно знайдені в ФОАП. Вона охоплює три основні аспекти кожної ФОАП: концепції, моделювання та процес.

Концепції: TOGAF надає, на відміну від інших, відповідні можливості управління та репозиторій, використовуючи для них специфічну модель. Хоча TOGAF описує необхідну бізнес- та ІТ-архітектуру в ADM, він більше орієнтований на розвиток ІТ і не може забезпечити належне узгодження між бізнесом та ІТ. Оскільки FEA походить від EAP, то майже всі їхні атрибути є однаковими. І далі, так як DODAF призначений для певного домену, він практично розглядає всі поняття архітектури у відповідний спосіб. І останнє, на відміну від інших ФОАП, у Gartner (GEAF) більше уваги приділяється процесу розробки і підтримки адекватних концепцій архітектури.

Моделювання: використання відповідного моделювання як для бізнесу, так і ІТ доменів має важливе значення для ФОАП. GEAF і DODAF/MODAF не надають методів роботи з цілісністю та простежуваністю. Хоча FEAF, NAF і TOGAF надають відповідні методи моделювання, вони відрізняються в навчанні і використанні. TOGAF надає детальну документацію про свої методи і процеси, але доступ до них та їх використання потребує більшого часу, ніж для інших. TOGAF зазначає, що розробники архітектури повинні вибрати необхідний процес для проекту з наявних фаз TOGAF, а це і є вузьким місцем, яке викликає труднощі використання завдяки своїй складності на проекті. Динамічний аспект та складність архітектури є ще однією проблемою, яка не вирішується всіма вибраними ФОАП.

Процес: TOGAF розглядає реалізацію архітектури як безперервний процес, таким чином, він більше зосереджується на континуумі та репозиторіях.

Крім того, TOGAF використовує процес вимог для підтримки фаз ADM, в той час, як інші ФОАП не використовують цю функцію. NAF і FEAF, як і для першого набору критеріїв, мають однаковий стан. DODAF використовує необхідні дії в кожному атрибуті процесу з метою реалізації архітектури для підприємства, але він не використовує процес вимог належним чином. Незважаючи на те, що Gartner не розглядає всі атрибути концепцій ефективно,

він розглядає реалізацію архітектури за допомогою ефективного плану, отриманого завдяки їх величезному досвіду.

Використані позначення [13]:

- 2 : високий розгляд або детальний і чіткий опис;
- 1 : середній розгляд або слабкий опис;
- 0 : низький розгляд або опис верхнього рівня.

В результаті проведеного аналізу отримано наступні результати (табл. 1.3):

- У концепціях: більшість згаданих ФОАП охоплюють всі поняття.

Стратегії та артефакти підтримуються більшістю ФОАП; і, навпаки, групування та репозиторії не використовуються в більшості ФОАП.

– У моделюванні: SOA має найбільший бал, TOGAF має коливання у оцінках (в деяких атрибутах має високу оцінку, а в інших - низьку). FEAF, Gartner та RM-ODP отримали найнижчі оцінки. Обрані ФОАП не мають визначеного плану для відображення складності та динамічних аспектів архітектури.

– У процесі: хоча, покрокова структура, детальний дизайн і реалізація є найбільш використовуваними атрибутами в ФОАП, вимоги, обслуговування і безперервність необхідно розглядати додатково через відсутність розгляду в більшості ФОАП.

Табл. 1.3. Оцінка ФОАП відповідно до концептуальних визначень

	ZF	TOGAF	FEAF	DODAF	MODAF	NAF	TEAF	GEAF	RM-ODP	SOA
Концепції										
Групування	1	1	0	1	1	1	1	1	1	2
Артефакти	2	2	1	1	1	2	1	1	1	2
Управління	1	2	0	1	1	2	0	1	1	2
Репозиторій	1	1	1	1	1	1	0	1	1	2
Стратегія	2	2	2	2	2	2	1	1	1	2
Моделювання										
Простий у використанні	1	0	1	1	1	1	1	1	1	2
Легкий для навчання	1	0	1	1	1	1	1	1	1	2

Простежуваність	1	2	1	0	0	1	0	0	0	1
Цілісність	1	2	1	0	0	1	0	0	0	1
Різні представлення	2	1	1	1	1	1	1	0	1	2
Складність	0	0	0	0	0	0	0	0	0	1
Динамічний аспект	0	0	0	0	0	0	0	0	0	1
Процес										
Наявність вимог	2	2	0	0	1	1	1	0	1	2
Покроковість	1	1	1	1	1	1	1	1	1	2
Детальний дизайн	1	1	1	1	1	1	0	1	1	2
Реалізація	1	1	1	1	1	1	0	1	1	2
Наявність керівництв	2	2	2	1	1	2	1	0	0	2
Технічне обслуговування	1	1	1	0	1	1	0	0	0	2
Безперервність	1	2	0	0	0	0	0	0	0	1
Усього										
	22	23	14	13	15	20	8	10	12	33

Оцінка якісних вимог. У табл. 1.4 представлено порівняння за третім набором критеріїв. Ці оцінки є суб'єктивними і базуються на огляді літератури, особистому розумінні архітектури і потребах у застосуванні архітектури для розробки критичних інформаційних інфраструктур. Використано наступну шкалу оцінювання: 0 - не підтримує (критерії не можуть бути реалізовані в ФОАП), 1 - частково підтримує (критерії можуть бути застосовані до певної міри), 2 - повністю підтримує (критерії можуть бути повністю реалізовані в ФОАП).

Табл. 1.4. Оцінка ФОАП відповідно до якісних вимог

	ZF	TOGAF	FEAF	DoDAF	MODAF	NAF	TEAF	GEAF	RM-ODP	SOA
Організаційна сумісність	2	2	2	2	2	2	1	1	1	1

	ZF	TOGAF	FEAF	DoDAF	MODAF	NAF	TEAF	GEAF	RM-ODP	SOA
Спільна інфраструктура та сумісність	1	1	1	1	1	1	1	1	1	1
Технічна сумісність	1	1	2	1	1	1	0	1	1	1
Рухливість	2	2	1	1	1	1	1	1	1	2
Повторне використання	1	1	2	1	1	0	0	1	0	2
Усього	7	7	8	6	6	5	3	5	4	7

Оцінка вимог до розвитку. Табл. 1.5 показує порівняння десяти фреймворків за вибраними критеріями четвертого набору. Була використана така ж шкала оцінювання, що і для попереднього набору: 0 - не підтримує, 1 - частково підтримує, 2 - повністю підтримує.

Табл. 1.5. Оцінка ФОАП відповідно до вимог розробки

Критерії	ZF	TOGAF	FEAF	DoDAF	MODAF	NAF	TEAF	GEAF	RM-ODP	SOA
Зусилля, необхідні для розробки та підтримки	2	3	2	2	2	2	2	2	1	2
Орієнтація служби	2	2	2	2	2	2	1	2	1	3
Оцінка та управління	2	2	3	1	1	1	1	2	1	3
Довідкові моделі	1	2	3	2	2	3	1	2	2	2
Документація	2	2	3	1	1	1	1	2	2	1
Економічна ефективність	1	3	3	2	2	3	1	2	1	3
Усього	10	14	16	10	10	12	7	12	8	14

Підсумкове оцінювання. У цьому розділі роботи наведено результати порівняльного аналізу ФОАП. На підставі порівняння цілей, концептуальних визначень, вимог до розвитку та якісних вимог, в табл. 1.6 наведена остаточна загальна оцінка ФОАП. На основі загальної оцінки обрано наступні два ФОАП для подальшого критичного аналізу та використання при проектуванні та впровадженні критичної інформаційної інфраструктури: TOGAF та SOA. Можна зробити висновок про те, що ці види ФОАП є найкращими для проектування критичних інформаційних інфраструктур, оскільки вони відповідають більшості

критеріїв. TOGAF прийнятий багатьма урядами і має багато переваг, серед яких наявність зрілого архітектурного процесу та його інтеграція з домінуючою мовою ArchiMate. Що стосується SOA, він не настільки широко використовується як TOGAF, але має безліч переваг, таких як бізнес-орієнтована розробка, можливість повторного використання, гнучкість, незалежність платформи. Однак жоден з ФОАП не є завершеним. Всі розглянуті ФОАП мають сильні і слабкі сторони та доповнюють один одного.

Табл. 1.6. Загальна оцінка ФОАП

Вимога	ZF	TOGAF	FEAF	DoDAF	MODAF	NAF	TEAF	GEAF	RM-ODP	SOA
Мета	27	52	42	46	46	47	38	50	42	52
Концептуальні визначення	22	23	14	13	15	20	8	10	12	33
Якісні вимоги	7	7	8	6	6	5	3	5	4	7
Вимоги до розвитку	10	14	16	10	10	12	7	12	8	14
Усього	66	96	80	75	77	84	56	77	66	106

Всі проаналізовані ФОАП або опускають або мають дуже стислий опис обґрунтування дизайну архітектури, незважаючи на те, що вони мають вирішальне значення для проектування КІІ. Архітектури повинні бути правильно розроблені та перевірені на ранніх стадіях проектування КІІ. Автор пропонує включити обґрунтування дизайну в ФОАП, які використовуються для проектування критичної ІТ-інфраструктури з такими функціями:

- перехресні вимоги на розробку архітектури для перевірки її цілісності та відстежуваності;
- документування компромісних проектних рішень на основі кількісного визначення набору критеріїв, переваг та ризиків;
- опис компромісів та зроблених вдосконалень щодо вимог;
- використання сценаріїв для відображення аналізу проектів;
- опис доцільності, можливості або неможливості використання варіанту архітектури.

Для даного набору входів завжди існує більше одного можливого варіанту архітектури. Кожен вибір дизайну архітектури в кожній точці прийняття рішення

пов'язаний з набором критеріїв, переваг і ризиків, які вимірюються як кількісно, так і якісно. Критерії не тільки відображають ресурси, необхідні для реалізації проекту, але також є компромісним або альтернативним критерієм з точки зору функціональності та інших факторів. Аналогічно, переваги відображають відносні переваги архітектури. Ризики представляють рівень невизначеності для бажаних бізнес-цілей через технічні, комерційні чи інші причини. Архітектурні рішення потребують набору багатовимірних входів $1...m$ у вигляді проектних міркувань. Кожен варіант проекту архітектури $1...n$, пов'язаний з набором критеріїв [37], переваг і ризиків.

Обраний варіант архітектури $Arch$ є результатом функції $ArchDesign$, як показано нижче (1.1) .

$$ArchDesign_{[1..n]}(I) \rightarrow Arch_{[1..m]}[Criteria, Benefits, Risks] \quad (1.1)$$

Вибір архітектурного дизайну з усіх можливих варіантів дизайну, які найкраще підходять для критичної інформаційної інфраструктури з збалансованим представленням мінімізованих ризиків та критеріями з максимальним значенням є завжди важковирішуваною проблемою. Це процес пошуку архітектурного компромісу. В даний час деякі ФОАП мають можливість фіксувати компромісні рішення шляхом документування системних обмежень і припущень. Автор вважає, що опис архітектурного проектування слід розширити додатковими документами, які включають критерії, переваги та ризики.

1.3 Аналіз існуючих методів та підходів представлення та обґрунтування архітектури критичної інформаційної інфраструктури

Мови представлення архітектури підприємств (АП) розглядаються як інструмент цілісного описування підприємства критичної інфраструктури, що пов'язують всі напрямки діяльності підприємства, які зазвичай розглядаються окремо. Маючи такий опис, можна розглянути загальносистемний вплив змін на всіх рівнях від бізнес-процесів до ІТ.

Метою аналізу є визначення основних особливостей підходів до раціоналізації архітектури та можливості їх використання для проектування та обґрунтування проектних рішень щодо критичної інформаційної інфраструктури.

Задачею даного аналізу є визначення недоліків основних підходів до раціоналізації архітектури, їх порівняння з точки зору використання як підходу до раціоналізації архітектури критичної інформаційної інфраструктури.

Прийняття рішення щодо вибору дизайну системи включає наступні компоненти:

- обґрунтування проектних рішень;
- стратегії прийняття цих рішень;
- чинники, що впливають на їх прийняття.

Обґрунтування рішення. Просте пояснення обґрунтування проектного рішення – *явне / чітке* уявлення про процес міркування над проектом. Існують і інші аспекти проектування. Принцип проектування може бути наміром мотивувати створення артефакту проектування, бути обмеженням або припущенням, яке впливає на артефакт проектування, компромісом між вимогами, судженням для вибору з ряду варіантів проектування та аргументацією «за» чи «проти» проектної пропозиції.

Є різні підходи до процесу проектного міркування. Один з них – аргументація. Основним аргументаційним представленням є використання вузлів та посилань для позначення знань та відносин. Він бере свій початок від аргументованого представлення Тулміна, яке будується за допомогою даних, вимог, гарантій, підстав та спростувань [166]. У схемі Тулміна прикладом вузла є дані або претензія, які являють собою спостереження та висновок відповідно. *Тип/вид* зв'язку типу «так» може бути використаний для з'єднання вузлів для представлення індуктивних відносин. Цей підхід став базою для численних аналогічних аргументальних підходів, таких як інформаційна система на основі проблем (IBIS) [167] та мова обґрунтування проектування (DRL) [168]. Вони в основі демонструють проблему, аргумент та вирішення аргументації проектування.

Інший підхід до представлення концепції проектування полягає у використанні методологій на основі шаблонів. Ці методи використовують стандартні шаблони, які входять в процес розробки, щоб полегшити схему обґрунтування проектування. На відміну від методів, що базуються на аргументації, практики, які використовують методи на основі шаблонів, не створюють аргументаційні діаграми для обговорення, а натомість фіксують результати міркування. Цей підхід орієнтований на практичне застосування обґрунтування проектування в галузі. Прикладами цього підходу є шаблон опису рішення архітектури (ADDT) [169] та шаблон Views and Beyond (V&B) [133].

Використання підходів обґрунтування проектів заснованих на аргументації іноді недостатньо, щоб визначити правильне рішення серед альтернатив. На проектне рішення може впливати багато факторів та вимог. Деякі з них мають більш високе значення або пріоритет, ніж інші. Тому, додатково у процесі прийняття рішень можуть використовуватися кількісні методи оцінки пріоритетів, витрат та переваг. Прикладом такого підходу є метод аналізу витрат (CBAM) [170].

Проаналізовано наступні основні методи обґрунтування проектних рішень [5]:

- інформаційна система на основі проблем (IBIS) [167, 171];
- процедурна ієрархія проблем (PHI) [172, 173];
- gIBIS [174];
- система представлення та обслуговування процесу знань (REMAP) [175];
- мова обґрунтування проектування (DRL) [176, 177];
- інженерія програмного забезпечення на базі обґрунтування проектного рішення (SeuRAT) [178];
- нотація «запитання, варіанти та критерії» (QOC) [178, 179];
- *Views and Beyond (V&B)*[133];
- шаблон опису рішення щодо архітектури (ADDT) [169];
- архітектура обґрунтування та зв'язування елементів (AREL) [132];

- метод аналізу недоліків архітектури (АТАМ) [133];
- метод аналізу витрат (СВАМ) [170].

Детальний опис основних методів представлення та обґрунтування проектних рішень наведено в [5].

Незважаючи на те, що методи раціонального проектування заклали основу для конструкторських міркувань, більшість з них все ще мають проблеми з точки зору зручності використання. Для їх порівняння використані наведені нижче критерії успішного впровадження концепції розробки:

- *Ефективне представлення обґрунтування дизайну* – на основі аргументації моделей фіксуються як міркування, так і структура аргументації обґрунтування дизайну. Структура аргументації є часовитратною і її важко простежити. Тому її слід спростити, не втрачаючи основну інформацію про обґрунтування дизайну.

- *Ефективна комунікація обґрунтування дизайну* – дизайнери хочуть знати проблеми, обґрунтування, потенційні альтернативи та елементи дизайну, на які впливає рішення. Отже, необхідність повторювати процес обговорення, як це пропонується аргументованими моделями, таким чином зменшена. Отже, проблеми в моделях, що базуються на аргументації, розташовуються вище над представленням їх обговорення та нище, ніж їх представлення взаємозв'язків з дизайнерськими артефактами.

- *Фокус на артефактах дизайну* – вимоги та специфікації дизайну об'єктів використовуються для підтримки системи оцінки та технічного обслуговування. Тому для обґрунтування дизайну необхідно пояснити артефакти дизайну, що містяться в цих специфікаціях.

- *Простежуваність та аналіз впливу* – основне використання концепції дизайну полягає в тому, щоб допомогти дизайнерам зрозуміти обґрунтування та проблеми дизайну з метою виконання подальшого технічного обслуговування. Одним з таких заходів є аналіз впливу змін. Методи обґрунтування дизайну повинні підтримувати аналіз ефектів пульсації (хвильовий вплив), використовуючи можливість простежувати обґрунтування

конкретних проектних рішень з метою пояснення конструктивної залежності.

– *Обґрунтування комплексного дизайну* – обґрунтування дизайну складається з багатьох типів інформації. Міркування може ґрунтуватися на аргументації або на кількісному аналізі. Тому методи обґрунтування дизайну повинні бути всеосяжними та гнучкими, щоб відобразити ці різні типи міркувань.

– *Загальна підтримка інструментів* – розробники архітектури завжди стикаються з дуже обмеженим графіком проектування. Використання іншого інструменту для представлення конструкторського обґрунтування збільшує накладні витрати на документацію та відокремлює знання від дизайну. Обґрунтування дизайну слід зафіксувати як побічний продукт, за допомогою того самого інструмента, в якому розробляється сам дизайн.

– *Корисність підходу для розробки критичних інформаційних інфраструктур* – можливість, корисні засоби та елементи у підходах, які можна використати для проектування та обґрунтування дизайну критичних інформаційних інфраструктур.

Проаналізуємо корисність існуючих методів обґрунтування дизайну за критеріями, викладеними вище. Кожен метод оцінюється відповідно до того, чи відповідає він критеріям повністю (2 бали), частково (1 бал) чи не відповідає (0 балів) (табл. 1.7) [1].

Табл. 1.7. Аналіз корисності існуючих методів обґрунтування дизайну

Метод Критерій	gIBIS	PHI	REMAP	DRL	SeuTAT	QOC	V&B	ADDT	AREL
Ефективне представлення обґрунтування дизайну	0	0	0	0	0	0	2	2	2
Ефективна комунікація обґрунтування дизайну	0	0	0	0	0	0	2	2	2

Фокус на артефактах дизайну	1	1	2	1	2	0	2	1	2
Простежуваність та аналіз впливу	0	0	1	0	1	0	0	1	2
Обґрунтування комплексного дизайну	0	0	0	0	0	0	1	1	2
Загальна підтримка інструментів	0	0	0	0	0	0	2	2	2
Корисність підходу для розробки критичних інформаційних інфраструктур	0	0	0	0	0	0	0	1	1
Всього	1	1	3	1	3	0	9	10	13

Підхід QOC підходить для простого, аргументованого проектування, де рішення мають чіткі варіанти, а критерії та елементи можуть бути узагальнені. Він розроблений для надання архітекторам простого методу, що стимулює міркування. Однак, це не настільки однозначно у випадку складного дизайну. Проектування архітектури є досить складним завданням і вимагає зазвичай залучення багатьох варіантів з власними критеріями та оцінками. Прості конструктивні рішення можуть легко стати заплутаними і незрозумілими.

Як і IBIS, QOC, DRL призначена для того, щоб забезпечити архітекторам модель і словниковий запас, що стимулює обговорення проектування. Велика різниця методу, на відміну від IBIS і QOC, полягає в тому, що DRL призначений для асинхронного використання. Іншими словами, DRL слід використовувати після того, як сам процес проектування вже здійснений. Він використовується для побудови обґрунтування за допомогою аналізу історичних записів

дизайнерських сеансів. Очевидний і його недолік при аналізі моделі, її складність швидко зростає при ускладненні дизайну.

Використовуючи задокументоване обґрунтування рішення, SeuRAT дозволяє супроводжувачим користувачам отримувати та перевіряти обґрунтування для проведення технічного обслуговування. Бург виявив, що SeuRAT забезпечує кращі результати серед протестованих людей, які не є експертами в Java, ніж результати їх відповідної контрольної групи для виявлення проблем та виконання завдань [180].

Мета ADDT і V&B полягає в наданні методів обґрунтування дизайну для практиків. Тому вони мають більш прагматичний підхід, ніж інші методи. З точки зору представлення та комунікації дизайну обґрунтування, їх структура менш формальна, ніж інші методи. Через це вони можуть бути реалізовані за допомогою різних типів програмних інструментів і легко можуть бути прийняті організацією. Проте обмежене графічне представлення в цих двох методах обмежує їх здатність надавати обґрунтоване обчислення і можливість відстеження. Інший підхід AREL не має вад двох вище наведених методів обґрунтування дизайну, але також не може бути використаний у вихідному стані для проектування та обґрунтування проектних рішень критичної інформаційної інфраструктури. Тому пропонується використати новий підхід, створений та адаптований для обґрунтування проектних рішень при побудові критичної інформаційної інфраструктури.

1.4 Висновки до першого розділу

В даному розділі наведено порівняльний аналіз архітектурних фреймворків для проектування критичної інформаційної інфраструктури. Введено модель розуміння для ФОАП на основі фундаментальних елементів архітектури. За цією моделлю розуміння можна визначити або адаптувати ФОАП, що є прийнятним для проектування архітектури критичної

інформаційної інфраструктури. Вибрані елементи з декількох фреймворків можуть бути використані спільно для задоволення конкретних потреб розробки.

Для аналізу ФОАП, які мають різні точки зору, у якості критеріїв використано 4 різні набори елементів. Базуючись на підтримці цих елементів у рамках архітектурних фреймворків, визначено два ФОАП з різними характеристиками, але достатньо хорошими, для використання при проектуванні архітектури КІІ.

Також, в ході аналізу виявлено поширені недоліки в ФОАП, особливо щодо питань обґрунтування дизайну архітектури. Як результат, запропоновано концепцію використання критеріїв, переваг і ризиків як основу для визначення компромісних проектних рішень. Обґрунтування проекту може бути використано для об'єднання архітектурних рішень з метою забезпечення відстеження та перевірки. Жоден з проаналізованих ФОАП не визначає свої вимоги до рівня деталізації архітектурного проекту. Автор вважає, що ця відмінність важлива з точки зору життєвого циклу розвитку КІІ. Тому запропоновано використовувати аналіз ризиків щодо архітектури з метою визначення необхідного архітектурного проекту.

Також представлено детальний компаративний аналіз підходів до обґрунтування проектних рішень, визначені їх недоліки. Таким чином, зроблено перший крок на шляху до створення інструментарію опису, документування та обґрунтування процесу прийняття проектних рішень щодо архітектури критичної інформаційної інфраструктури.

РОЗДІЛ 2

МОДЕЛІ І МЕТОДИ ПРЕДСТАВЛЕННЯ АРХІТЕКТУРНИХ РІШЕНЬ КРИТИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ

У науковій літературі зустрічаються різні формальні і неформальні описи розподілених систем (РС) [181–183]. Характерною властивістю РС є наявність взаємодіючих компонентів системи. У даній роботі основна увага приділяється моделюванню та верифікації моделей особливого класу розподілених систем – критичним інформаційним інфраструктурам, тому далі під компонентами РС розуміються взаємодіючі процеси, критичні та не критичні. Процеси виконуються на одному або декількох обчислювальних пристроях. У загальному випадку, обчислювальні пристрої можуть бути фізично віддалені один від одного. У даній роботі затримка, що виникає при передачі даних, не враховується, і взаємодія процесів вважається миттєвою.

Далі під розподіленою системою будемо мати на увазі критичну інформаційну інфраструктуру.

Для аналізу поведінки розподіленої системи часто в якості моделі РС використовуються транзиційні системи (Transition System, TC) [112]. Ключовими складовими системи переходів є поняття стану і переходу. Стан TC є абстракцією стану розподіленої системи. У стані TC відображені значення структур даних процесів РС, необхідних для аналізу її поведінки. Зі стану моделі може здійснюватися перехід в один або кілька наступних станів. Перехід відповідає локальній або комунікаційній події РС, що моделюється. TC можуть супроводжуватися розміткою станів та/або переходів. У цьому випадку говорять про розмічені транзиційні системи (Labelled Transition System, PTC) [112, 184]. При побудові моделі будуються PTC процесів РС. Модель всієї РС утворюється в результаті паралельної композиції PTC процесів.

2.1 Основні означення

2.1.1 Розмічені транзиційні системи

Нехай задана множина можливих міток дій A . Дії відповідають класам еквівалентності подій E . Прикладами дій можуть бути: відсилання/отримання повідомлень процесом або компонентом, зміна стану процесу або компоненту тощо. Далі мається на увазі, що всі дії РТС належать множині A . В множині також знаходиться спеціальна неспостережна дія τ .

Означення 2.1. РТС називається шістка $(S, S^0, A, R, \Sigma, L)$, в якій S - скінченна множина станів, S^0 - підмножина S початкових станів системи, $A \in \mathcal{A}$ - скінченна множина міток дій, R - відношення переходів на множині $S \times A \times S$, Σ - множина змінних системи, L - функція розмітки станів $L: S \rightarrow 2^\Sigma$.

Далі будемо використовувати запис $s \xrightarrow{a} t$, що означає $(s, a, t) \in R$ для $s, t \in S, a \in A$. За допомогою відношення R задаються можливі варіанти дискретних переходів з одного стану до іншого. Переходи супроводжуються мітками дій.

Для кожного стану $s \in S$ за допомогою функції L задаються значення змінних з множини Σ в стані s .

За допомогою символу $\mathcal{M}$ будемо позначати множину всіх РТС.

Означення 2.2. Скінченним шляхом π в РТС $(S, S^0, A, R, \Sigma, L)$ називається така скінченна послідовність станів та дій $s_1, a_1, \dots, a_{n-1}, s_n$, які чергуються, що $s_i \in S, a_j \in A$ для всіх $1 \leq i \leq n, 1 \leq j < n$ та $(s_i, a_j, s_{i+1}) \in R$ для всіх $1 \leq i < n$. Нескінченним шляхом називається така нескінченна послідовність станів та дій $s_1, a_1, \dots, a_{n-1}, s_n, \dots$, які чергуються, що $s_i \in S, a_j \in A$ для $i \leq 1, j \geq 1$ та $(s_i, a_j, s_{i+1}) \in R$ для всіх $i \geq 1$.

Далі скінченний шлях $\pi = s_1, a_1, \dots, a_{n-1}, s_n$ будемо позначати як $s_1 \xrightarrow{a_1, \dots, a_{n-1}} s_n$, а нескінченний шлях $\pi = s_1, a_1, \dots, a_{n-1}, s_n, \dots$ як $s_1 \xrightarrow{a_1, \dots, a_{k-1}} s_k \xrightarrow{a_k} \dots$. Для скінченного або нескінченного шляху π запис $\text{suf}(\pi, k)$ означає суфікс шляху π , який починається

з стану s_k .

Означення 2.3. Для шляхів на множині станів S визначимо наступну функцію станів, які зустрічаються на шляху, $\text{PrS}: S^* \cup S^\omega \rightarrow 2^S$. Стан $s \in \text{PrS}(\pi)$ для шляху $\pi = s_1 \xrightarrow{a_1, \dots, a_{k-1}} s_k \xrightarrow{a_k} \dots$ тоді і тільки тоді, коли знайдеться таке $i \geq 1$, що $s = s_i$.

Означення 2.4. Нехай задана РТС $M = (S, S^0, A, R, \Sigma, L)$ та шляхи $\alpha' = s_1 \xrightarrow{a_1, \dots, a_{m-1}} s_m$, $\alpha'' = t_1 \xrightarrow{b_1, \dots, b_{n-1}} t_n$ і при цьому $s_m = t_1$. Тоді конкатенацією шляхів α' і α'' назовемо шлях $\alpha = s_1 \xrightarrow{a_1, \dots, a_{m-1}} t_1 \xrightarrow{b_1, \dots, b_{n-1}} t_n$. Далі будемо позначати конкатенацію шляхів α' і α'' як $\alpha = \alpha' \circ \alpha''$.

Означення 2.5. Для РТС $M = (S, S^0, A, R, \Sigma, L)$ стан $t \in S$ називається досяжним із стану $s \in S$, якщо знайдеться деякий скінченний шлях $\alpha = s \xrightarrow{a_1} s_1 \dots s_k \xrightarrow{b} t$. Стан t називається досяжним, якщо він є досяжним з деякого початкового стану РТС.

Означення 2.6. Нехай задана РТС $M = (S, S^0, A, R, \Sigma, L)$ та функція перейменування дій $\text{ren}: A \rightarrow A'$. Тоді результатом застосування функції ren до РТС M (позначається як M^{ren}) є така РТС $M_l = (S_l, S_l^0, A_l, R_l, \Sigma_l, L_l)$, що:

- вірні рівності $S_l = S, S_l^0 = S^0, A_l = A, R_l = R, \Sigma_l = \Sigma, L_l = L$,
- перехід $(s, a, t) \in R$ виконується тоді і тільки тоді, коли $(s, \text{ren}(a), t) \in R_l$.

В подальшому будемо використовувати спеціальну функцію ren^i дописування індексу i діям РТС, яка відображає кожну дію a на дію a_i .

Означення 2.7. Нехай задана РТС $M = (S, S^0, A, R, \Sigma, L)$ та функція початку переходу $\text{bg}: R \rightarrow S$. Тоді результатом застосування функції bg до $r_i \in R$ (позначається як $\text{bg}(r_i)$) є стан початку переходу r_i , тобто $\text{bg}(r_i) \in s_i, s_i \in S$.

Означення 2.8. Нехай задана РТС $M = (S, S^0, A, R, \Sigma, L)$ та функція кінця переходу $\text{end}: R \rightarrow S$. Тоді результатом застосування функції end до $r_i \in R$ (позначається як $\text{end}(r_i)$) є стан кінця переходу r_i , тобто $\text{end}(r_i) \in s_i, s_i \in S$.

2.1.1.1. Асинхронна паралельна композиція РТС

Розглянемо деревоподібний хвильовий алгоритм як приклад РТС. В алгоритму приймають участь процеси трьох типів: кореневий процес-ініціатор (прототип Root); процес, який відповідає листовій вершині комунікаційного дерева (прототип Inner); процес, який відповідає внутрішній вершині комунікаційного дерева (прототип Leaf).

За алгоритмом процес-ініціатор посилає повідомлення своїм нащадкам. Процеси типу Inner отримують повідомлення від батьківського процесу та надсилають його далі своїм нащадкам. Процеси типу Leaf отримують повідомлення, запускають задачу прийняття рішення та надсилають у відповідь підтвердження. Процеси типу Inner, отримавши підтвердження від всіх нащадків також запускають задачу прийняття рішення, після чого передають підтвердження батьківському процесу.

На рис. 2.1 представлені діаграми РТС прототипів процесів алгоритму для бінарного дерева. Стани РТС зображені у вигляді еліпсів. Початкові стани: s_0, t_0, u_0 . Переходи позначаються діями $rcv\_left, rcv\_right, snd\_left, snd\_right$. Дія з префіксом rcv відповідають отриманню повідомлення, а дія з префіксом snd відповідає відправці повідомлення. Переходи, для яких не вказана мітка, позначені дією τ .

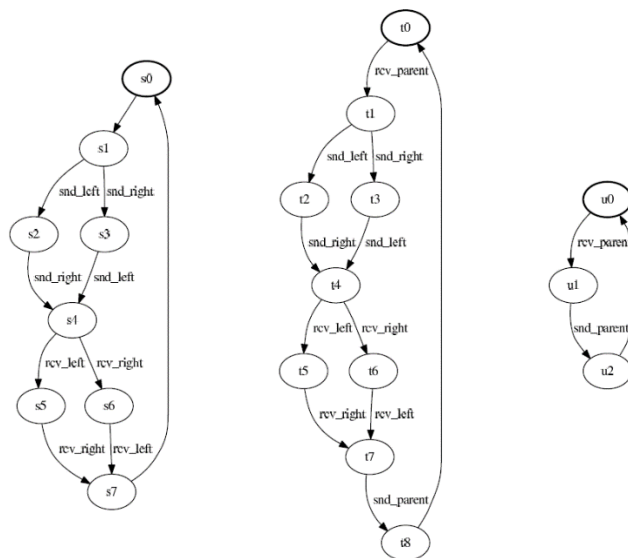


Рис. 2.1. Діаграми РТС процесів типу Root, Inner, Leaf

Для опису правил взаємодії на РТС визначається операція паралельної композиції. За допомогою даної операції будується нова РТС, яка моделює асинхронне виконання та синхронний обмін повідомленнями двох вихідних РТС. Маючи РТС всіх процесів E можна побудувати РТС всієї E , виконуючи ітеративно операцію паралельної композиції.

Означення 2.9. Нехай задані РТС $M_1 = (S_1, S_1^0, A_1, R_1, \Sigma_1, L_1)$ та $M_2 = (S_2, S_2^0, A_2, R_2, \Sigma_2, L_2)$ такі, що $S_1 \cap S_2 = \emptyset, A_1 \cap A_2 = \emptyset, \Sigma_1 \cap \Sigma_2 = \emptyset$. Нехай також задана синхронізаційна пара $\Lambda = (\Delta, \bar{}, \Delta \subseteq A_1, \bar{} : \Delta \rightarrow A_2$. Тоді модель $M = (S, S^0, A, R, \Sigma, L)$ називається паралельною композицією РТС M_1 і M_2 , якщо:

- вірні співвідношення $S = S_1 \times S_2, S^0 = S_1^0 \times S_2^0, A = A_1 \cup A_2 \setminus (\Delta \cup \bar{\Delta}), \Sigma = \Sigma_1 \cup \Sigma_2,$
- вірно $L : S \rightarrow 2^\Sigma, L((s_1, s_2)) = L(s_1) \cup L(s_2),$
- відношення $R \subseteq (S_1 \times S_2) \times A \times (S_1 \times S_2)$ задається наступним чином:
 $((s, u), a, (t, v)) \in R$ тоді і тільки тоді, коли виконується одна з трьох умов:
 - виконується перехід першої моделі: $(s, a, t) \in R_1, u = v$;
 - виконується перехід другої моделі: $(u, a, v) \in R_2, s = t$;
 - виконується синхронний обмін повідомленнями:
 $(s, b, t) \in R_1, (u, b, v) \in R_2, a = \tau$.

Розглянемо операцію композиції на прикладі хвильового алгоритму. Нехай M_1 і M_2 - РТС двох процесів, отриманих з РТС прототипів Inner та Leaf відповідно шляхом перейменування за допомогою функцій ren^1 і ren^2 . Визначимо синхронізатор $\Lambda = (\Delta, \bar{})$ наступним чином: $\Delta = \{snd\_left_1, rcv\_left_1\}, \overline{snd\_left_1} = rcv\_parent_2, \overline{rcv\_left_1} = snd\_parent_2$. Тоді прикладами переходів РТС $M = M_1 \parallel_\Lambda M_2$ є наступні переходи:

- внутрішній перехід першого процесу $(s_0, t_0) \xrightarrow{\tau} (s_1, t_0),$
- внутрішній перехід другого процесу $(s_7, t_8) \xrightarrow{\tau} (s_7, t_0),$

– комунікаційний перехід обох процесів $(s_1, t_0) \xrightarrow{\tau} (s_2, t_1)$.

$M = P_1 \parallel \dots \parallel P_n$, де $P_1, \dots, P_n$ - РТС процесів РС.

2.1.1.2 Деякі приклади використання РТС

Сучасні вимоги до програмного і технічного забезпечення сучасних інформаційних систем та критичних інформаційних інфраструктур спонукають до розробки нових методів їх проектування, обґрунтування та коректної реалізації. Пропонований метод проектування ілюструється прикладами синтезу сервісів критичних інформаційних інфраструктур типу **PaaS** (*Platform as a Service*), **IaaS** (*Infrastructure as a Service*), **SaaS** (*Software as Service*).

IaaS зазвичай надає уніфіковані апаратні і програмні ресурси, але в деяких випадках і на інфраструктурному рівні для установки ПО з оплатою pay as you go (по мірі використання). Замовлена інфраструктура може динамічно масштабуватися. На базі такого підходу побудовані Amazon EC2 Service і Amazon S3.

PaaS надає більш високий рівень сервісу, що дозволяє розробляти, тестувати і впроваджувати власні застосунки. Вбудована масштабованість накладає обмеження на тип розробляються. Яскравим прикладом реалізації такого підходу є сервіс Google App Engine.

SaaS пропонує готове спеціалізоване ПО, що веде до спрощення використання додатків і до зменшення витрат на розробку. Одним з чудових прикладів реалізації за таким підходом є Salesforce та її онлайнова система управління відносинами з клієнтами.

Оскільки в критичних інформаційних інфраструктурах зосереджені потужні і досить дорогі ресурси, виникає проблема їх ефективного використання. Це стає можливим за рахунок правильного проектування та реалізації критичної інформаційної інфраструктури, а в подальшому,

використанням оптимальних методів та алгоритмів розподілу, управління і диспетчерування ресурсів і навантаженням на основі комплексу відповідних математичних моделей і методів.

Приклад 2.1. Синтез критичного процесу для критичної інформаційної інфраструктури [45, 46]. Роботу, пов'язану з запуском критичного процесу, можна представити у вигляді взаємодії скінченної множини підсистем. В термінах транзиційних систем застосування можна представити у вигляді синхронного добутку наступних ТС:

1. а) *ТС 1..N* – інкапсулює дії, що виконуються виключно для пошуку та виділення певних потрібних ресурсів. Залежно від складності та розподіленості підсистема виділення конкретного ресурсу, може бути як одиничною системою зі строго послідовними переходами, так і мати ієрархічну структуру. Основні дії, що належать до зони відповідальності ТС 1..N: а) пошук ресурсу; б) виділення ресурсу; в) використання ресурсу; г) повернення ресурсу в пул ресурсів. ТС у формальному вигляді можна представити наступним чином:

$$A = (S, S^0, A, R, \Sigma, L) = (\{0, 1, 2, 3\}, 0, \{ready, search\_res\_A, res\_A\_alloc, res\_A\_alloc, released\}, R, \Sigma, L)$$

Структура ТС представлена на рис. 2.2.

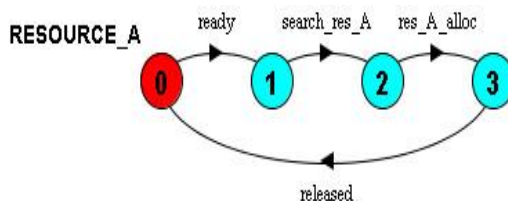


Рис.2.2. Структура ТС виділення ресурсу

Множина станів наступна: 0 – початковий стан, 1 – стан готовності, 2 – ресурс знайдено, 3 – ресурс виділено, ресурс звільнено.

2. е) *ТС N+1* – інкапсулює дії, що виконуються виключно для запуску критичного процесу.

ТС у формальному вигляді можна представити наступним чином:

$$A = (S, S^0, A, R, \Sigma, L) = (\{0, 1, 2, 3, 4, 5\}, 0, \{ready, res\_A\_alloc, res\_B\_alloc, proc\_start, proc\_killed, released\}, R, \Sigma, L)$$

Структура ТС для випадку використання двох видів ресурсу представлена на рис. 2.3.

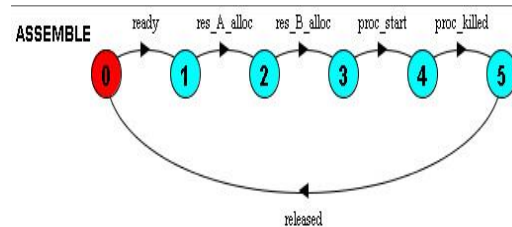


Рис. 2.3. Структура ТС запуску критичного процесу

Множина станів наступна: 0 – початковий стан, 1 – стан готовності, 2 – ресурс А виділено, 3 – ресурс В виділено, 4 – критичний процес запущено, 5 – критичний процес знищено, ресурси звільнено.

Результатом мінімізованої композиції ТС для випадку використання 2-х ресурсів буде наступна ТС (див. рис. 2.4):

$$A = (S, S^0, A, R, \Sigma, L) = (\{0..9\}, 0, A, R, \Sigma, L), \\ \text{де } A = \{ready, search\_res\_A, search\_res\_B, res\_A\_alloc, res\_B\_alloc, proc\_start, proc\_killed, res\_A\_alloc, res\_B\_alloc, proc\_start, proc\_killed, released\}$$

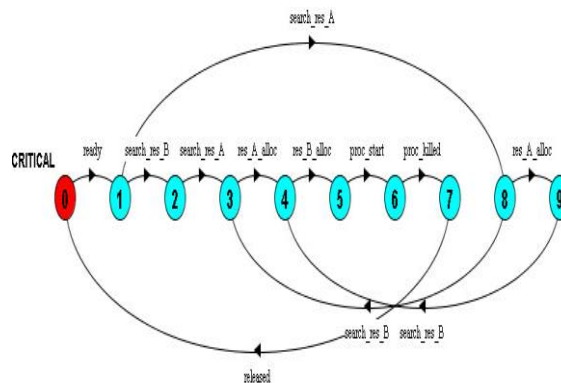


Рис. 2.4. Структура композиції ТС

Приклад 3.2. Взаємодія двох SAAS-систем (дуже спрощена модель).

Представимо нашу взаємодію систем у термінах ТС:

1. а) *ТС 1..2* – інкапсулює дії, що виконуються виключно для певних потрібних ресурсів. Основні дії, що належать до зони відповідальності *ТС 1..2*:

а) отримання даних (ресурсу) з системи; б) запис даних (повернення ресурсу) в систему.

Структура ТС представлена на рис. 2.5.



Рис. 2.5. Структура ТС 1..2

2. а) *ТС 3..4* – інкапсулює дії, що виконуються кожною системою SaaS при взаємодії з іншою. Основні дії, що належать до зони відповідальності *ТС 3..4*: а) отримання даних (ресурсу) з системи; б) запис даних (повернення ресурсу) в систему; в) обмін даними між системами; г) очікування (див. рис. 2.6).

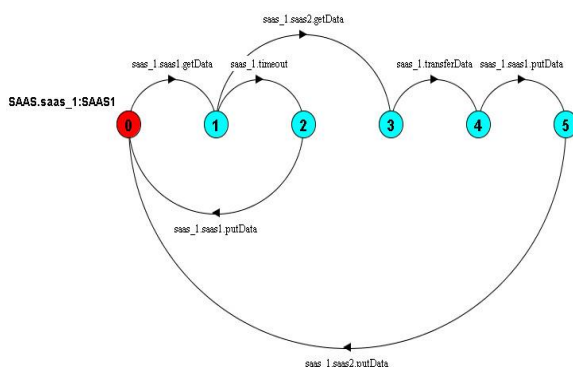


Рис. 2.6. ТС система SaaS

Мінімізована композиція *ТС 1..4* дає можливість побачити взаємодію двох систем SaaS, яка включає взаємний обмін інформацією та її обробку (див. рис.2.7).

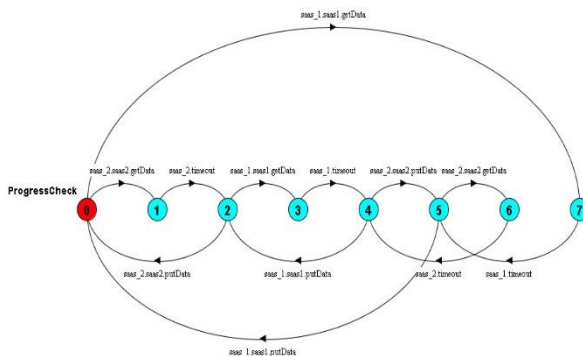


Рис. 2.7. Композиція ТС взаємодії систем SaaS

2.2 Представлення архітектури критичної інформаційної інфраструктури на базі розмічених транзиційних систем

В даному розділі розглядається метод представлення архітектури критичної інформаційної інфраструктури, який отримав назву методу представлення та обґрунтування дизайну критичної архітектури (*Rationale and Evaluation of Critical Architecture Design, RECAD*) [2]. RECAD має на меті допомогти архітекторам створювати та документувати архітектурний дизайн з акцентом на архітектурні рішення та обґрунтування проекту. RECAD охоплює три типи знань архітектури: проектні питання, проектні рішення та результати проектування. Ці об'єкти знань представлені стандартними об'єктами уніфікованого моделювання (UML). Проблема проектування – це матеріали, які впливають на прийняття дизайнерських/проектних рішень. Ця сутність інкапсулює такі поняття, як функціональні вимоги (наприклад, сценарії та діаграма співпраці), нефункціональні вимоги (наприклад, всі атрибути якості) та контексти проекту. Вона також фіксує інформацію про проектні рішення та обґрунтування проекту. Результати проектування включають в себе отримані рішення. Прикладами є класи, компоненти, інтерфейс та спосіб використання. Будь-який індивідуальний тип об'єкту архітектурного знання фіксується шляхом застосування заздалегідь визначеного тегового шаблону стереотипу. Концептуальна модель підходу представлена рис. 2.8.

Фактично RECAD представляється ациклічним графом, який пов'язує елементи архітектури CAE з елементами обґрунтування архітектури AR за допомогою направленою відношення ARtr. CAE є критичним або не критичним архітектурним елементом, який приймає участь в Рішенні як вхідне значення (*Мотиваційна причина*) або як вихідне значення (*Результат дизайну*). AR енкапсулює результат Обґрунтування архітектури для Рішення. Так як AR має відношення типу 1:1 з Рішенням, для даного підходу він представляє точку прийняття рішення щодо обґрунтування архітектури. Відношення між CAE та AR

представляється за допомогою направленої асоціації ARtr, яка є причинно-наслідковим зв'язком.

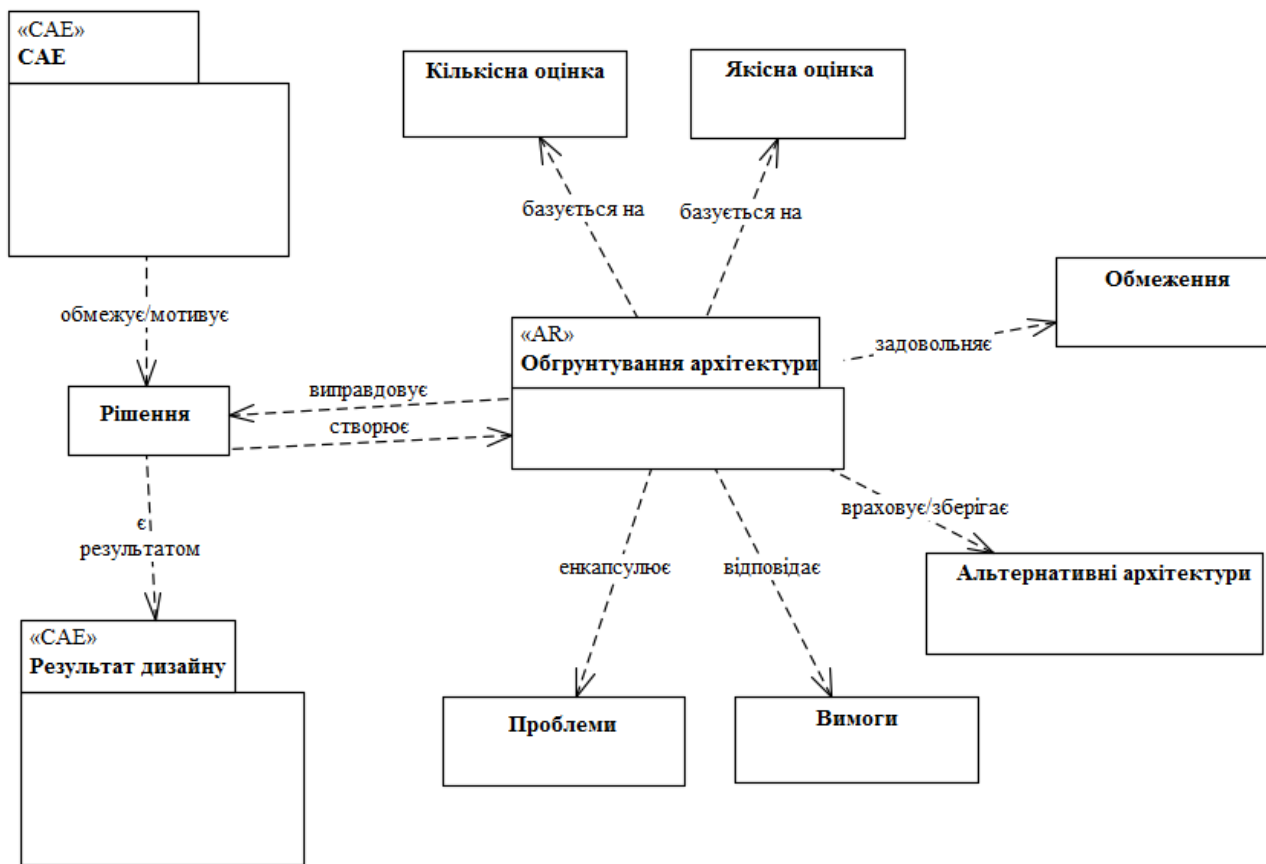


Рис. 2.8. Концептуальна модель RECAD

Означення 2.10. Модель RECAD – це РТС $RECAD = (CAE, CAE^0, AR, A, R, \Sigma, L)$, де CAE – множина вузлів, що представляють критичні та некритичні елементи архітектури, CAE^0 – підмножина CAE початкових станів елементів архітектури, AR – множина вузлів, які представляють обґрунтування архітектури, A – скінченна множина міток дій, Σ – множина змінних системи, L – функція розмітки елементів $L: CAE \rightarrow 2^\Sigma$, R – множина $R \subseteq (CAE \times AR) \cup (AR \times CAE)$ направлених зв'язків між вузлами, для якої виконуються наступні умови [2]:

1. всі вузли AR повинні бути асоційовані з хоча б одною причиною та одним наслідком, тобто для $\forall r \in AR$ існує така причина $e \in CAE$, що $(e, r) \in R$ і існує такий наслідок $e' \in CAE$, що $(r, e') \in R$;

2. не існує підмножини з множини R , яка утворює направлений цикл.

Основною формою модельної конструкції є шлях виду $\{CAE_1, CAE_2, \dots\} \rightarrow AR_1 \rightarrow \{CAE_a, CAE_b, \dots\}$, де CAE_1, CAE_2 - входи, або причини рішення AR_1 , а CAE_a, CAE_b - виходи, або наслідки цього рішення. На рис. 2.9 представлена UML діаграма, яка демонструє означену модельну конструкцію. Кратність відношень діаграми показує, що мотиваційні та результуючі CAE є непустими множинами, з'єднаними через єдиний елемент AR. Обмеження унікальності на діаграмі визначає, що кожний екземпляр CAE не може бути відображеним більше, ніж один раз.



Рис. 2.9. Причинно-наслідковий зв'язок між CAE та AR

Направлені зв'язки ARtr представляють причинно-наслідкові відношення. CAE призводить до AR через мотивацію або обмеження Рішення, AR в свою чергу генерує CAE типу Результат дизайну маючи Обґрунтування архітектури. CAE може бути одночасно вхідним і вихідним, якщо використовується для двох Рішень. У якості вхідного він може представляти собою артефакт наступних типів: вимога, прецедент, клас, імплементація. Як вихідний – новий або переглянутий елемент дизайну.

В RECAD архітектурні елементи CAE є артефактами, які формують частини дизайну архітектури. Вони включають бізнес-потреби, які потрібно задовольнити, технічні та організаційні обмеження, що накладаються на проект архітектури, припущення, які потрібно перевірити, та об'єкти дизайну, які є результатом архітектурного проектування.

Елементи архітектури можуть також бути класифіковані за точками зору на архітектуру. Причина такої класифікації – мати можливість фокусуватись на різних аспектах проектного рішення. В даному дослідженні, згідно з результатами аналізу ФОАП, використовуються точки зору на архітектуру підходу TOGAF. Для класифікації використано наступні точки зору на архітектуру: бізнес-логіка, рівень даних, застосування, технології. А також додану нову точку зору на архітектуру, орієнтовану на критичні елементи.

На рис. 2.10 представлена UML-діаграма класифікації архітектурних елементів відповідно до точок зору на архітектуру. Бізнес-орієнтована точка зору на архітектуру включає архітектурні елементи такі як функціональні та нефункціональні вимоги, бізнес-середовище, середовище інформаційної системи, технологічне середовище. Це основні драйвери проектування архітектури.

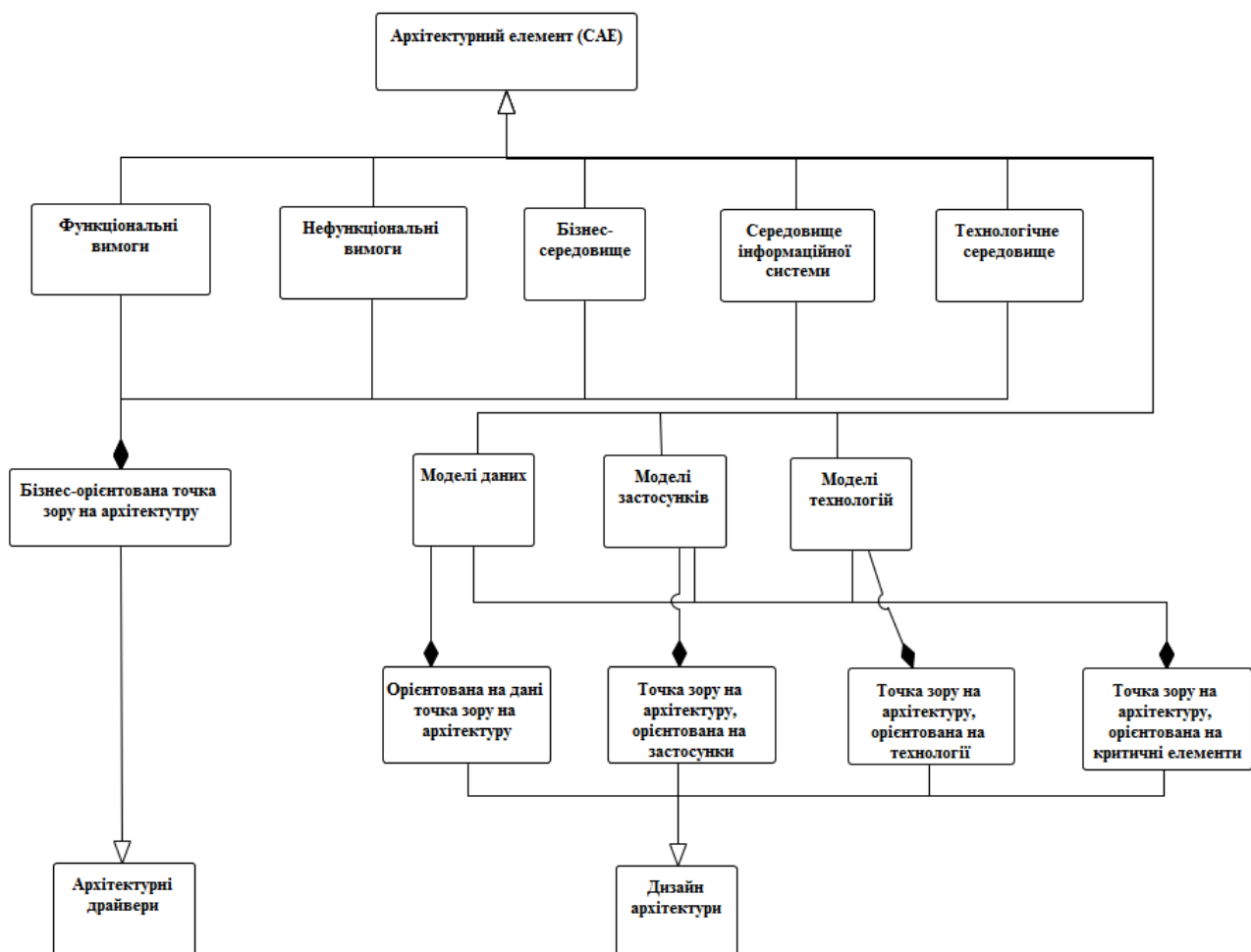


Рис. 2.10. UML-діаграма класифікації архітектурних елементів відповідно до точок зору на архітектуру

Архітектурний елемент як мотиваційна причина може бути представлений у вигляді:

- функціональних та не функціональних вимог;
- припущень – документування невідомих або очікувань надає контекст для прийняття рішення;
- обмежень – можуть бути технічного, бізнес, організаційного або природного характеру;
- об'єктів дизайну – можуть впливати через свою поведінку на дизайн архітектури.

У вигляді результату дизайну архітектурний елемент може бути представлений як [2]:

- об'єкти дизайну – можуть бути результатом рішення стосовно архітектури, які задовольняють певні мотиваційні причини;
- уточнені об'єкти дизайну – можуть бути уточнені в результаті прийняття відповідних рішень.

В даному дослідженні передбачається, що бізнес-орієнтована точка зору включає вимоги та фактори середовищ. Створено п'ять категорій бізнес-орієнтованої точки зору на архітектуру, які описують різні аспекти впливу на архітектурний дизайн. Це і є архітектурні драйвери. Системні вимоги включають в себе функціональні та нефункціональні вимоги. Така класифікація дозволяє архітектору під час аналізу відслідковувати процес обґрунтування рішень до специфічних класів кореневих причин.

Елементи «Дизайн архітектури» є результатом процесу проектування. Вони класифіковані за наступними точками зору на архітектуру:

- орієнтовані на дані – отримані та використовувані застосунками;
- орієнтовані на застосунки – логіка обробки і структура програмного забезпечення, критичні та некритичні сервіси;
- орієнтовані на технології – технології та середовища, які використовуються для впровадження та розгортання системи, критичні та

некритичні компоненти.

Підхід RECAD надає можливість використовувати три типи обґрунтувань архітектури: кількісний, якісний та за допомогою альтернативної архітектури. Якісне обґрунтування представляє процес обґрунтування та аргументи у текстовій формі, фактично, за та проти для кожного проектного рішення. Кількісне обґрунтування використовує різні критерії при оцінюванні проектних рішень. Третій тип передбачає документування та зберігання відкинутих альтернативних проектних рішень та їх подальший перегляд з метою оцінки достатності наявних параметрів оцінювання наявних архітектурних проектів, а також для майбутнього використання в інших проектах.

Слід зазначити, що архітектурне рішення може еволюціонувати з часом. Причиною цьому можуть бути як зміни в бізнес-процесах підприємства, так і зміни самого бізнес-середовища. Еволюціонуючи, можна втратити вихідний архітектурний проект та опис процесу обґрунтування рішень щодо цього проекту. Тому потрібно якимось чином зберігати всю історію еволюції проекту. Для цього пропонується використовувати розширену модель RECAD.

Означення 2.11. Розширена модель RECADe – це РТС $RECADe = (CAE, CAE^0, AR, A, R, \Sigma, L)$, на якій задана бієктивна функція відображення $SSf : (CAE \rightarrow CAE_h) \cup (AR \rightarrow AR_h)$ між архітектурними елементами або обґрунтуваннями архітектури, де:

$RECAD_{cur} = (CAE_{cur}, CAE_{cur}^0, AR_{cur}, A_{cur}, R_{cur}, \Sigma_{cur}, L_{cur})$ - актуальна модель архітектури і виконуються наступні умови:

- $CAE_{cur} \subseteq CAE$,
- $AR_{cur} \subseteq AR$,
- $A_{cur} \subseteq A$,
- $\Sigma_{cur} \subseteq \Sigma$,
- $CAE_{cur}^0 \subseteq CAE^0$,
- $L_{cur} : CAE_{cur} \rightarrow 2^{\Sigma_{cur}}$,

- $R_{cur} \subseteq R \cap ((CAE_{cur} \times AR_{cur}) \cup (AR_{cur} \times CAE_{cur}))$,
- не існує підмножини з множини R_{cur} , яка утворює направлений цикл;

$RECAD_h = (CAE_h, CAE_h^0, AR_h, R_h, \Sigma_h, L_h)$ - історична модель архітектури і

виконуються наступні умови:

- $CAE_h \subseteq CAE$,
- $AR_h \subseteq AR$,
- $A_h \subseteq A$,
- $\Sigma_h \subseteq \Sigma$,
- $CAE_h^0 \subseteq CAE^0$,
- $L_h : CAE_h \rightarrow 2^{\Sigma_h}$,
- $R_h \subseteq R \cap ((CAE_h \times AR_h) \cup (AR_h \times CAE_h))$;
- не існує підмножини з множини R_h , яка утворює направлений цикл;

і для яких виконуються наступні умови:

1. $CAE_h = CAE \setminus CAE_{cur}$,
2. $AR_h = AR \setminus AR_{cur}$,
3. $A_h = AR \setminus A_{cur}$,
4. $\Sigma_h = \Sigma \setminus \Sigma_{cur}$,
5. $CAE_h^0 = CAE^0 \setminus CAE_{cur}^0$,
6. $L_h = L \setminus L_{cur}$.

На рис. 2.11 представлена відповідна моделі UML-діаграма.

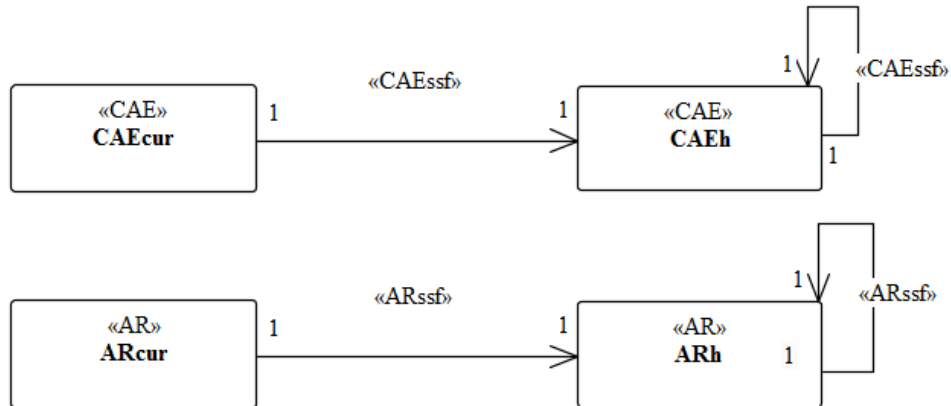


Рис. 2.11. UML-діаграма розширеної моделі RECADE

Імплементація підходу RECAD за допомогою UML. Для імплементації архітектурного елементу CAE визначено новий стереотип <<CAE>>. Даний стереотип розширює можливості UML-конструктів типу «об'єкт» та «клас» щодо підтримки властивостей відстежуваності та критичності. Це означає, що до існуючих конструктів додаються додаткові характеристики та атрибути з метою покращення процесу обґрунтування рішень. Додаткові атрибути стереотипу <<CAE>> імплементовані як теговані значення (tagged values):

- *elementID* – унікальний ідентифікатор архітектурного елементу;
- *elementVersion* – ціле число, яке ідентифікує версію архітектурного елементу і використовується для моделі RECADE;
- *elementType* – тип елементу відповідно до введеної класифікації за точками зору на архітектуру;
- *elementSubType* – підтип елементу відповідно до введеної класифікації за точками зору на архітектуру;
- *currFlag* – прапорець актуальності елементу (“Y” – остання версія, “N” – історична версія);
- *criticalFlag* – прапорець критичності елементу (“Y” – критичний елемент, “N” – некритичний елемент);
- *lastUpdateDateTime* – час останньої модифікації елементу;
- *author* – автор змін;
- *documentLocation* – місце розташування будь-яких зовнішніх документів, які описують елемент.

Під час перевірки відстежуваності параметри *elementType* та *elementSubType* дозволяють обирати фокус, щодо якого потрібно провести дослідження архітектури. Параметри *elementVersion* та *currFlag* дозволяють відстежувати процес еволюції архітектури, *criticalFlag* – критичність елементів та процесів [2].

З метою моделювання бізнес-орієнтованих точок зору, які містять вимоги, фактори середовищ та припущення використовується створений стереотип

<<CAE>> для розширення елементів UML таких як клас, об'єкт, артефакт та прецедент (рис. 2.12).

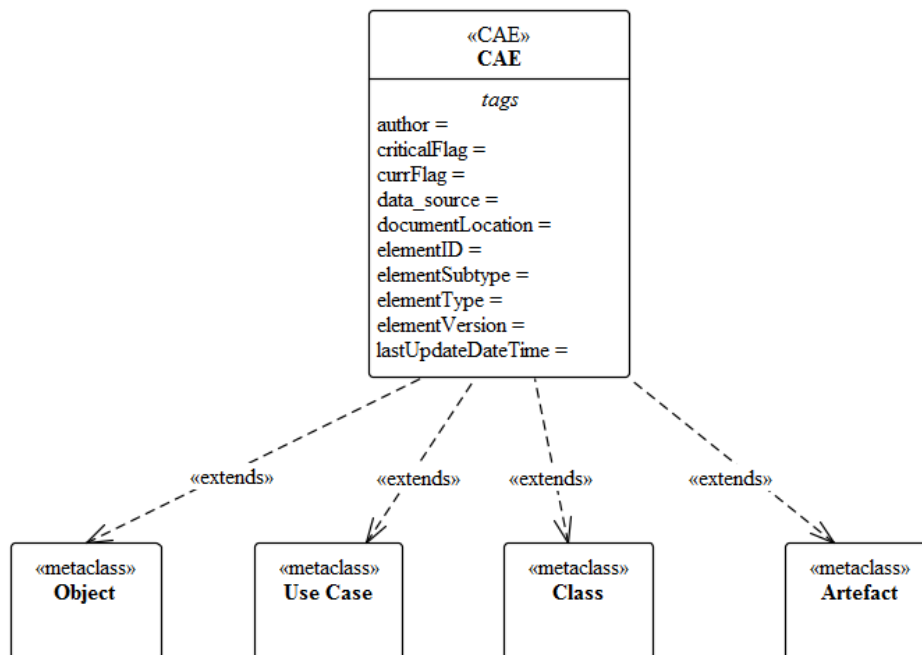


Рис. 2.12. Розширення елементів UML стереотипом <<CAE>>

Використання таких елементів дозволяє створювати моделі з різними точками зору на архітектуру.

Для імплементації елементу «обґрунтування рішення» використано стереотип <<AR>>, який розширяє можливості UML-конструкту «Пакет». Додаткові атрибути стереотипу <<AR>> імплементовані як теговані значення:

- *arID* – унікальний ідентифікатор елементу «Обґрунтування архітектури»;
- *currFlag* – прапорець актуальності елементу (“Y” – остання версія, “N” – історична версія);
- *criticalFlag* – прапорець критичності елементу (“Y” – критичний елемент, “N” – некритичний елемент);
- *lastUpdateDateTime* – час останньої модифікації елементу;
- *author* – автор особи, яка приймає рішення.

На рис. 2.13 представлена UML-діаграма взаємозв’язків стереотипів <<AR>> та <<AAR>>. Їх імплементація базується на конструкті «Пакет». AR

може включати нуль або більше елементів AAR (обґрунтування альтернативної архітектури) в залежності від кількості відкинутих альтернативних проектних рішень.

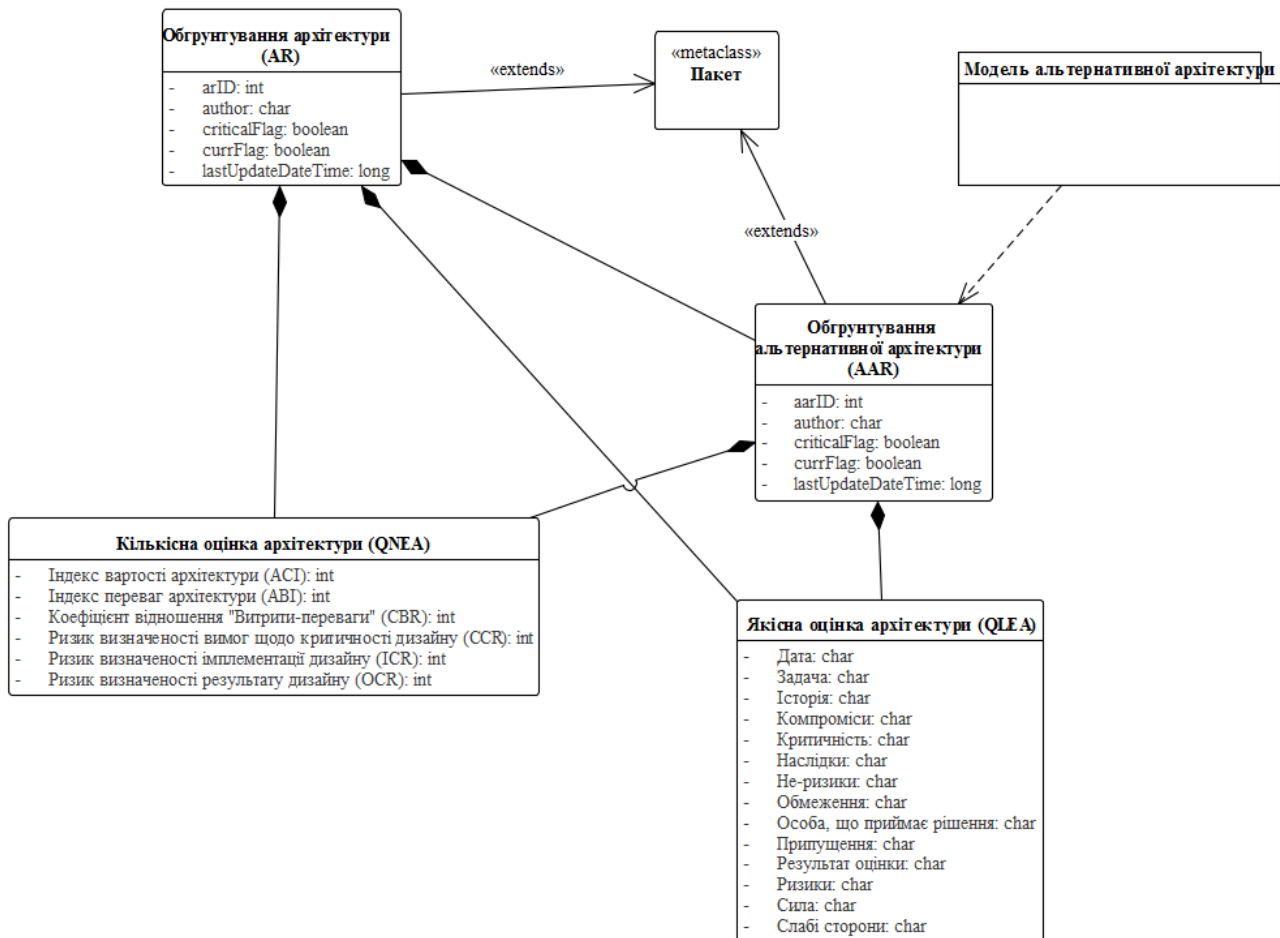


Рис. 2.13. UML-діаграма взаємозв'язків стереотипів <<AR>> та <<AAR>>

Стереотип <<AAR>> надає шаблон для документування альтернативних проектних рішень, які були відкинуті.

Відношення між CAE та AR представляється за допомогою направленої асоціації з новоствореним стереотипом <<ARtr>>.

Введені моделі RECAD та RECADe дозволяють представити та описати окремі конкретні проектні рішення щодо систем, компонент та процесів, які входять до архітектури критичної інформаційної інфраструктури.

Визначимо для моделі RECAD її асинхронну паралельну композицію (Означення 2.12).

Означення 2.12. Нехай задані PTC $RECAD_1 = (CAE_1, CAE_1^0, AR_1, A_1, R_1, \Sigma_1, L_1)$ та

$RECAD_2 = (CAE_2, CAE_2^0, AR_2, A_2, R_2, \Sigma_2, L_2)$ такі, що $CAE_1 \cap CAE_2 = \emptyset, AR_1 \cap AR_2 = \emptyset, A_1 \cap A_2 = \emptyset, \Sigma_1 \cap \Sigma_2 = \emptyset$. Нехай також задана синхронізаційна пара $\Lambda = (\Delta, \bar{\Delta}), \Delta \subseteq A_1, \bar{\Delta} : \Delta \rightarrow A_2$. Тоді модель $RECAD = (CAE, CAE^0, AR, A, R, \Sigma, L) = RECAD_1 \parallel_{\Lambda} RECAD_2$ називається паралельною композицією РТС $RECAD_1$ і $RECAD_2$, якщо:

- вірні співвідношення $CAE = CAE_1 \times CAE_2, AR = AR_1 \times AR_2, S^0 = S_1^0 \times S_2^0, A = A_1 \cup A_2 \setminus (\Delta \cup \bar{\Delta}), \Sigma = \Sigma_1 \cup \Sigma_2,$
- вірно $L : CAE \rightarrow 2^{\Sigma}, L((cae_1, cae_2)) = L(cae_1) \cup L(cae_2),$
- відношення $R \subseteq ((CAE_1 \times CAE_2) \times A \times (AR_1 \times AR_2)) \cup ((AR_1 \times AR_2) \times A \times (CAE_1 \times CAE_2))$ задається наступним чином: $((s, u), a, (t, v)) \in R$ тоді і тільки тоді, коли виконується одна з трьох умов:
 - виконується перехід першої моделі: $(s, a, t) \in R_1, u = v;$
 - виконується перехід другої моделі: $(u, a, v) \in R_2, s = t;$
 - виконується синхронний обмін повідомленнями: $(s, b, t) \in R_1, (u, b, v) \in R_2, a = \tau.$

Таким чином, модель архітектурного рішення критичної інформаційної інфраструктури можна представити наступним чином (Означення 2.13).

Означення 2.13. Модель архітектурного рішення критичної інформаційної інфраструктури визначається у вигляді паралельної композиції всіх окремих проектних рішень щодо систем, компонент та процесів, які входять до архітектури критичної інформаційної інфраструктури, заданих у вигляді РТС $RECAD$, тобто: $M = RECAD_1 \parallel_{\Lambda_1} \dots \parallel_{\Lambda_{n-1}} RECAD_n$, де $RECAD_1, \dots, RECAD_n$ - відповідні РТС, $\Lambda_1, \dots, \Lambda_{n-1}$ - задані на моделі пари синхронізації.

2.3 Верифікація моделей на базі РТС

У методах верифікації моделей (ВМ) специфікації РС зазвичай задаються за допомогою формул темпоральної логіки [112]. Формули темпоральної логіки

описують тимчасові властивості обчислень (послідовностей переходів) моделі. Найбільше застосування в методі ВМ отримали логіки дерев обчислень (Computational Tree Logic, CTL), лінійного часу (Linear Time Logic, LTL) і узагальнення обох логік – логіка CTL*. Огляд методів верифікацій моделей наведено в [26].

Задача верифікації моделей архітектурних рішень критичних інформаційних інфраструктур. Задача верифікації моделей архітектурних рішень критичних інформаційних інфраструктур формулюється наступним чином:

Задача 2.1. Дана модель архітектурного рішення $M = P_1 \parallel \dots \parallel P_n$, де $P_1, \dots, P_n$ - РТС RECAD критичної інформаційної інфраструктури. Задана формула (специфікація) Φ темпоральної логіки відносно змінних моделі M . Необхідно перевірити виконуваність формули Φ моделі M (позначається як $M \models \Phi$).

Задача верифікації параметризованих моделей архітектурних рішень критичних інформаційних інфраструктур.

У даній роботі розглядається узагальнення проблеми верифікації моделей. Методи верифікації застосовуються до моделей КІІ, представлених у вигляді UML-діаграм, що складаються з кінцевої множини елементів архітектури та обґрунтувань, заданих у вигляді РТС. Як зазначалося у вступі, існують КІІ, число процесів в яких залежить від початкової конфігурації або настройки КІІ, а множина таких конфігурацій нескінченна. При цьому число процесів не змінюється під час роботи. Для КІІ, в яких число процесів залежить від початкової конфігурації, можуть бути побудовані моделі з кінцевим числом станів для кожної початкової конфігурації системи. Так як множина початкових конфігурацій нескінченна, то і множина моделей архітектурних рішень з різним числом процесів нескінченна. Верифікація декількох, випадково обраних моделей, з цієї множини не гарантує виконуваність специфікації на всіх моделях множини. Для таких систем розглядається задача, яка в загальному випадку формулюється наступним чином:

Задача 2.2. Дано нескінченне сімейство скінченних моделей архітектурних рішень КІІ $\mathcal{F} = \{M_n\}$, параметризоване за параметром $n \in N$. Задана формула (специфікація) Φ темпоральної логіки. Необхідно перевірити виконуваність формули Φ на всіх моделях $\mathcal{F}$, тобто $M_n \models \Phi$ для всіх n . Ця задача в літературі отримала назву верифікації параметризованих моделей (ВПМ).

Постановка задачі 2.2 потребує уточнення, так як в загальній постановці явно не вказується спосіб визначення сімейства $\mathcal{F}$ та специфікації Φ . Тому, фактично будемо розглядати наступний варіант задачі 2.2.

Задача 2.3. Дано нескінченне сімейство скінченних моделей архітектурних рішень КІІ $\mathcal{F} = \{M_n\}$, параметризоване за параметром $n \in N$. Кожна модель $M_n = Q \| P_1 \| \dots \| P_n$ складається з РТС фіксованої моделі RECAD Q та n екземплярів РТС альтернативних моделей P_i . Зафіксована кінцева множина $I \subseteq \mathbb{N}$ індексів альтернативних моделей, що наблюдаються, екземплярів прототипів P . Специфікація Φ темпоральної логіки задається відносно змінних визначених моделей $P_i, i \in I$ та змінних моделі Q . Необхідно перевірити виконуваність формули Φ на всіх моделях сімейства $\mathcal{F}$, тобто $M_n \models \Phi$ для всіх n .

Фактично, в постановці задачі 2.3 використовується лише одна фіксована модель Q та n екземплярів прототипів P . Можна розглядати варіант задачі, в якому присутні декілька фіксованих моделей $Q_1, \dots, Q_m$ та декілька альтернативних моделей $P^a, P^b, \dots, P^z$, а модель $M_n = Q_1 \| \dots \| Q_m \| P_1^a \| \dots \| P_{n_a}^a \| \dots \| P_1^z \| \dots \| P_{n_z}^z$, де $n_a + \dots + n_z = n$. В деяких випадках, ця задача зводиться до постановки задачі 2.3 за допомогою побудови РТС моделі Q у вигляді паралельної композиції моделей $Q_1, \dots, Q_m$, а РТС прототипу P у вигляді паралельної композиції РТС прототипів $P^a, P^b, \dots, P^z$.

Запропонована постановка задач у вигляді (2.1-2.3) дозволяє застосувати для верифікації побудованих моделей широке коло формалізованих методів верифікації параметризованих моделей, заданих у вигляді РТС. В даній роботі для верифікації побудованих моделей при реалізації єдиної системи проектування використано один з методів пошуку інваріантів [185].

2.4 Критерій оптимальності проектування та функціонування КІІ

В ході аналізу процесу проектування нової критичної інформаційної інфраструктури потрібно враховувати:

- велику вартість та складність створюваної критичної інформаційної інфраструктури;
- наявність сучасних методологій, засобів та інструментів її створення.

Критична інформаційна інфраструктура повинна створюватись за наступними принципами:

- мінімізація чисельності необхідного людського професійного ресурсу та досягнення встановлених цілей експлуатації при мінімальних витратах цього ресурсу;
- мінімізація загальної тривалості робіт, які проводяться для критичної інформаційної інфраструктури;
- мінімізація часу зниження готовності критичної інформаційної інфраструктури;
- мінімізація часу відновлення готовності критичної інформаційної інфраструктури;
- мінімізація стороннього впливу при проведенні робіт з критичною інформаційною інфраструктурою;
- захищеність критичної інформаційної інфраструктури від можливих несанкціонованих дій;
- максимізація ефективності критичної інформаційної інфраструктури, її ресурсів та строку її служби.

Аналіз ІТ-інфраструктур та критичних інформаційних інфраструктур, цілей їх функціонування, структури та взаємозв'язків між елементами дозволяє визначити наступні основні їх особливості:

- наявність мети функціонування, що визначає її призначення та характер функціонування;
- наявність великої кількості різноманітних об'єктів управління;
- наявність ієрархічної структури управління;
- наявність мети та цілей управління для кожного підрівня ієрархічної структури;
- наявність великої кількості внутрішніх зв'язків в кожній підсистемі між її елементами;
- безперервна зміна стану функціонування елементів підсистем і системи в цілому, кількості її елементів та підсистем, критеріїв функціонування.

Оскільки в критичних інформаційних інфраструктурах зосереджені потужні і досить дорогі ресурси, виникає проблема їх ефективного використання. Це стає можливим за рахунок розподілу, управління і диспетчерування ресурсів, управління навантаженням на основі комплексу відповідних математичних моделей і методів. Формалізація проблеми ефективного використання ресурсів привела до задач нечіткого програмування [186] з широким вибором критеріїв оптимізації і врахуванням критичності, ресурсних, часових, технологічних та інших обмежень.

Для формування критерію оптимальності проектування і подальшого функціонування критичної інформаційної інфраструктури пропонується використати наступну множину параметрів [34]:

- доступність – визначає можливість мати доступ до всіх наявних ресурсів з будь-якої точки в будь-який час;
- надійність – визначає можливість безперервного користування наявним функціоналом критичної інформаційної інфраструктури;
- живучість – визначає можливість виконувати свої функції при втраті ресурсів, підсистем і т. ін.;

- забезпеченість – визначає показник максимальної кількості процесів та сервісів, що обслуговуються;
- відновлюваність – визначає показник тривалості відновлення готовності до експлуатації;
- економічність – визначає показник витрат різноманітних ресурсів на забезпечення функціонування критичної інформаційної інфраструктури;
- безпечність – визначає показник неможливості виконання несанкціонованих дій, спрямованих на порушення роботи критичної інформаційної інфраструктури чи її частин;
- адаптивність – визначає можливість критичної інформаційної інфраструктури динамічно реагувати та адаптуватися до змін в бізнес-процесах, наявних ресурсах і т. ін.;
- строк життя;
- ефективність – визначає можливість поєднання вище згаданих параметрів в кожному окремому випадку під визначену задачу.

При формуванні вимог критерію проектування критичної інформаційної інфраструктури необхідно також враховувати особливості розв'язуваних нею задач. Слід зазначити, що визначення всієї множини параметрів не можна повністю звести до системи формалізованих процедур, бо деякі з них вимагають якісного аналізу. Для такого аналізу слід використати метод структуризації, який дозволяє поділити задачу на підзадачі, визначитись за допомогою експертів або без них з методами розв'язання цих підзадач, обмеженнями використання цих розв'язків та методами поєднання розв'язків.

Враховуючи викладені аспекти та стадії життєвого циклу критичної інформаційної інфраструктури пропонується особливо звернути увагу на наступні питання при створенні критичної інформаційної інфраструктури:

- формування образу критичної інформаційної інфраструктури;
- визначення цілей, побудову дерева цілей та вибір шляхів їх досягнення, формування вимог та критерію оптимальності критичної інформаційної інфраструктури;

- оцінку альтернативних варіантів побудови критичної інформаційної інфраструктури;
- обґрунтування вибору архітектурних рішень.

Образ критичної інформаційної інфраструктури формується з врахуванням прогнозу її розвитку та наробіток, які були напрацьовані на етапі життєвого циклу «Планування» та «Проектування». При цьому попередньо визначаються структура критичної інформаційної інфраструктури (тобто, перелік її наявних та можливих складових, можливі конструктивні рішення і таке інш.), умови експлуатації (перелік характеристик, що визначають умови використання; фактори зовнішнього середовища, що впливають на функціонування і тд) (див. розділ 6).

Наступний крок – визначення цілей функціонування системи. На даному етапі потрібно врахувати вимоги замовника, стан технічного та наукового розвитку в даній галузі. На базі ітеративного процесу визначення цілей функціонування будується дерево цілей та шляхів їх досягнення, яке покриває всі елементи системи та етапи їх експлуатації. Виконується оцінка можливості досягнення поставлених цілей. Побудоване дерево дає можливість визначити основні вимоги до критичної інформаційної інфраструктури, а також можливі рекомендації по конструктивним рішенням окремих підсистем та елементів.

Для визначення найбільш ймовірного варіанту побудови критичної інформаційної інфраструктури потрібно:

- оцінити можливість якісного стрибка в розвитку на базі нових технічних ідей;
- оцінити можливість розвитку на базі попередніх рішень шляхом покращення окремих характеристик.

Підсумовуючи все вищесказане, пропонується визначити досяжність поставлених перед критичною інформаційною інфраструктурою цілей шляхом визначення узагальнених показників (про деякі з них згадувалось вище).

Узагальнені показники визначаються за допомогою відповідних математичних моделей, які будуються для кожної критичної інформаційної інфраструктури.

Задача побудови критичної інформаційної інфраструктури відноситься до задач багатокритеріального аналізу і в загальному випадку має наступний вигляд (2.1):

$$U_{opt} = opU(x) \quad (2.1)$$

де:

U_{opt} - оптимальне рішення;

$x = \{x_i \mid i = 1..n\}, x \in D_{x_i}$ - вектор експлуатаційних параметрів, а D_{x_i} - множина обмежень цих параметрів;

$U = \{U_i \mid i = 1..n\}, U \in D_{U_i}$ - вектор узагальнених експлуатаційних вимог, а D_{U_i} - підмножина їх допустимих значень;

op - оператор оптимізації.

При оцінці ефективності роботи критичної інформаційної інфраструктури на ранніх стадіях створення потрібно враховувати витрати на реалізацію, оскільки наявність ресурсів обмежена.

В загальному випадку, для кожного окремого випадку функціонування критичної інформаційної інфраструктури задача зводиться до пошуку компромісу між необхідними характеристиками функціонування. Наприклад, в одному випадку, вимагається максимальна безпека функціонування, а всі інші характеристики не такі важливі, в іншому – потрібно забезпечити максимальну надійність функціонування.

2.5 Висновки до другого розділу

У даному розділі представлений детальний опис методу представлення архітектури критичної інформаційної інфраструктури на базі розширених конструктів UML, наведено основні базові елементи математичного апарату

розмічених транзиційних систем, використаних для опису моделей даного методу представлення архітектури у формалізованому вигляді. Запропонований метод дозволяє представити сам процес відображення процесу міркувань щодо архітектури критичної ІТ-архітектури в досить зручний спосіб та дає можливість досліднику накопичувати історію цих міркувань. Також представлена адаптація задачі верифікації параметризованих моделей, яка дозволяє застосувати для верифікації моделей архітектурних проектів, представлених у вигляді UML-діаграм і формалізованих у вигляді РТС математичний апарат темпоральних логік і, як результат, дозволяє використовувати для розв'язання цієї задачі широке коло різноманітних методів верифікації моделей без їх значного доопрацювання та модифікації.

Наприкінці розділу описаний узагальнений критерій оптимальності проектування та функціонування КІІ.

РОЗДІЛ 3

МОДЕЛІ І МЕТОДИ ОБГРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ КРИТИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ

3.1 Метод обґрунтування рішень на базі ентропійного підходу

Даний метод базується на методі представлення процесу обґрунтування рішень RECAD. В процесі проектування архітектури виконуються наступні дві основні задачі:

- перегляд/уточнення вимог – в процесі часто виникають конфліктні вимоги або обмеження, які потребують перегляду або уточнення;
- сам процес проектування – відбувається до певного рівня деталізації, який задовольняє висунутим вимогам до проектування.

Як вже було зазначено вище, RECAD підтримує декілька варіантів оцінювання процесу обґрунтування архітектури. В цій роботі особливу увагу приділено кількісним методам оцінювання процесу.

В даному підході прийняття рішення базується на оцінюванні вартості, переваг та ризиків варіантів дизайну архітектури.

У випадку, коли є декілька варіантів побудови системи, які досягають тієї ж самої мети, архітектор спробує якимось чином обрати найкращий варіант. Для цього, в першу чергу, будуть відкинуті всі варіанти, які не працюють або не задовольняють вимогам. Наступним кроком архітектор спробує знайти варіант дизайну, який при максимальних перевагах дизайну потребує мінімальних витрат на реалізацію. Поставлена задача може виявитися надто складною через велику кількість варіантів дизайну архітектури.

Пропонований метод зберігає відповідні розрахунки за допомогою елементу UML «Кількісна оцінка архітектури» (QNEA). Індекс вартості архітектури (architecture cost index, ACI) являє собою відносний зважений показник вартості архітектури за шкалою від 1 до 10. Даний індекс враховує безліч різних витрат на архітектуру, серед яких:

- витрати на розробку – сюди відносяться витрати на розробку критичної інформаційної інфраструктури та вимоги щодо навчання персоналу;
- витрати на платформу – вартість обладнання та програмного забезпечення для побудови критичної інформаційної інфраструктури;
- витрати на підтримку – витрати на підтримку працездатності критичної інформаційної інфраструктури, модифікацію та адаптацію програмного забезпечення, обладнання тощо;
- потенційні витрати – витрати, які можуть потенційно виникнути в ході дизайну або експлуатації критичної IT- інфраструктури.

Індекс переваг архітектури (architecture benefits index, ABI) – відносний зважений показник переваг архітектури за шкалою від 1 до 10, що визначає рівень задоволення вимог, висунутих перед архітектурою.

Розглянемо наступний приклад. Потрібно визначити, яку СУБД використати при реалізації підсистеми ЄІС МВС «Єдине розподілене сховище даних» (див. розділ 5.1.3.9) (рис. 3.1).

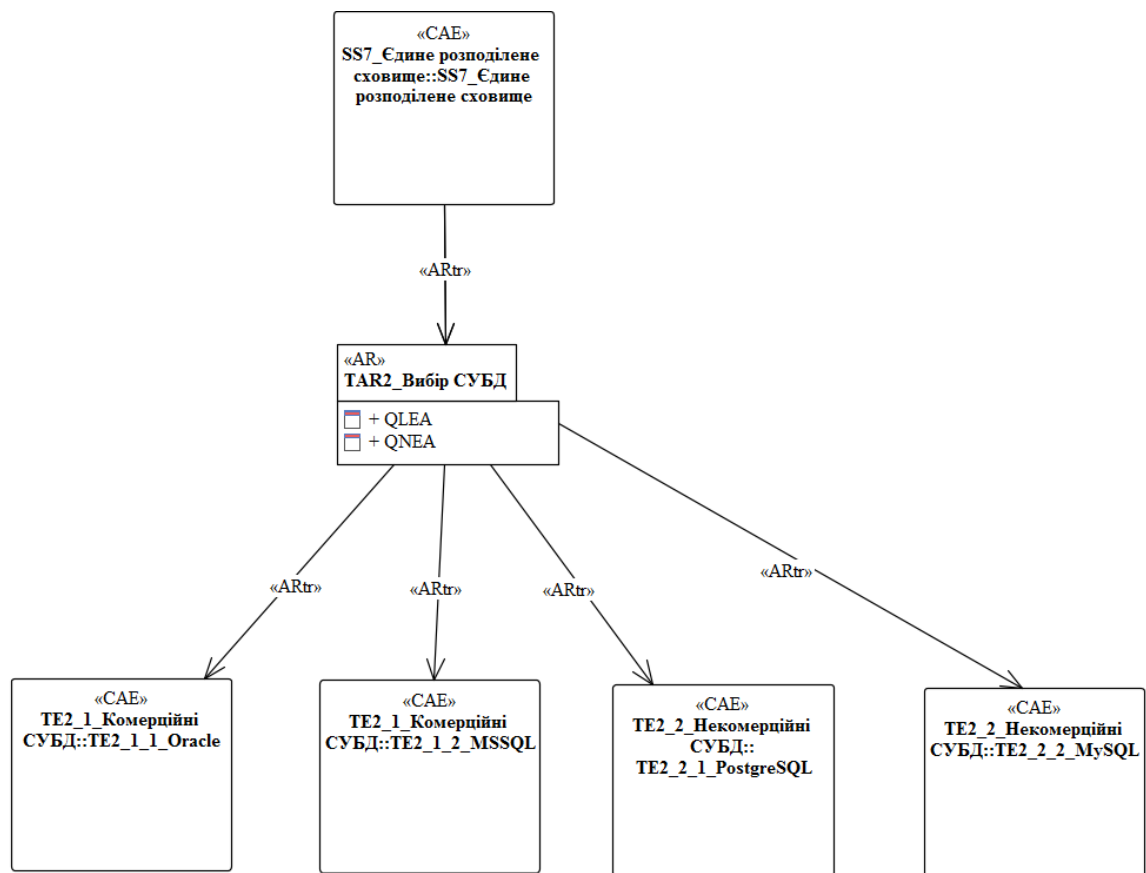


Рис. 3.1. Приклад обґрунтування рішення

Для визначення відповідного варіанту архітектури до уваги беруться наступні фактори:

- витрати:
 - вартість ліцензій;
 - витрати на підтримку;
 - витрати на навчання персоналу;
 - витрати на впровадження;
 - витрати на інтеграцію;
- переваги:
 - продуктивність;
 - гнучкість рішення;
 - надійність рішення;
 - масштабованість рішення;
 - безпечність рішення.

Кожний фактор оцінюємо за шкалою від 0 до 2. Результати оцінювання наведено в табл. 3.1.

Табл. 3.1. Результати оцінювання вибору СУБД

Фактори	Oracle	PostgreSQL	MySQL	MSSQL
Витрати				
Вартість ліцензій	2	0	1	1
Витрати на підтримку	2	0	0.5	0.5
Витрати на навчання персоналу	2	1	1	2
Витрати на впровадження	1	1	0	1
Витрати на інтеграцію	2	1	1	1
Індекс вартості архітектури (ACI)	9	3	3.5	5.5
Переваги				
Продуктивність	2	2	1	1
Гнучкість рішення	2	1.5	1	1
Надійність рішення	2	2	1	1
Масштабованість рішення	2	1	1	1
Безпечність рішення	2	1	1	1

Індекс переваг архітектури (ABI)	10	7.5	5	5

Як бачимо з табл. 3.1, СУБД Oracle має найбільший індекс вартості архітектури та індекс переваг архітектури. В залежності від того, які вимоги стоять перед архітектором, і буде зроблено вибір СУБД. Обидва ці параметри фіксуються в відповідних тегованих значеннях сутності QNEA.

При проектуванні архітектури часто виникають деякі невизначеності дизайну. Явне представлення цих невизначеностей дозволяє архітектору в певний момент часу до них повернутись та визначити їх. В даному методі використовуються наступні показники:

- ризик визначеності результату дизайну (outcome certainty risk, OCR)
- визначає ризик або рівень невизначеності досягнення архітектурним проектом результату дизайну, представленого індексом переваг архітектури;
- ризик визначеності імплементації дизайну (implementation certainty risk, ICR) – визначає ризик або рівень невизначеності виникнення непередбачених ситуацій при імплементації архітектурного рішення;
- ризик визначеності вимог щодо критичності дизайну (criticality certainty risk, CCR) – визначає ризик або рівень невизначеності порушення вимог до заданого рівня критичності при імплементації архітектурного рішення.

В табл. 3.2 наведена оцінка ризиків щодо прикладу, зазначеного вище.

Табл. 3.2. Результати оцінювання вибору СУБД (ризики)

Фактори	Oracle	PostgreSQL	MySQL	MSSQL
Ризик визначеності результату дизайну (OCR)	0.2	0.2	0.6	0.4
Ризик визначеності імплементації дизайну (ICR)	0.2	0.3	0.5	0.5
Ризик визначеності вимог щодо критичності дизайну (CCR)	0.2	0.5	0.8	0.8

Оцінювання ризиків є важливим процесом при проектуванні критичної інформаційної інфраструктури, так як надає архітектору інструментарій для всебічної оцінки можливих перешкод, які можуть виникати в процесі

проектування та обґрунтування рішень щодо вибору оптимальної архітектури критичної інформаційної інфраструктури [19, 21, 49].

Для оцінювання та порівняння альтернативних варіантів архітектурних рішень використання наведених вище показників є досить незручним. Для оптимізації процесу порівняння введемо декілька нових параметрів:

– рівень очікуваної переваги (expected benefit, EB), який визначається як:

$$EB = (1 - OCR)(1 - CCR) * ABI, \quad (3.1)$$

– рівень очікуваної вартості (expected cost, EC), який визначається як:

$$EC = (1 + ICR) * (1 + CCR) * ACI, \quad (3.2)$$

– коефіцієнт відношення «Витрати-переваги» (cost-benefit ratio, CBR), який визначається як:

$$CBR = \frac{EB}{EC} = \frac{(1 - OCR)(1 - CCR) * ABI}{(1 + ICR) * (1 + CCR) * ACI}. \quad (3.3)$$

Саме останній коефіцієнт і є тим зручним елементом, який дозволяє порівнювати між собою альтернативні проектні рішення.

В табл. 3.3 наведений розрахунок нових параметрів щодо прикладу, зазначеного вище.

Табл. 3.3. Результати оцінювання вибору СУБД (закінчення)

СУБД	Oracle	PostgreSQL	MySQL	MSSQL
Індекс вартості архітектури (ACI)	9	3	3.5	5.5
Індекс переваг архітектури (ABI)	10	7.5	5	5
Ризик визначеності результату дизайну (OCR)	0.2	0.2	0.6	0.4
Ризик визначеності імплементації дизайну (ICR)	0.2	0.3	0.5	0.5
Ризик визначеності вимог щодо критичності дизайну (CCR)	0.2	0.5	0.8	0.8
Рівень очікуваної переваги (EB)	6.4	3	0.4	0.6
Рівень очікуваної вартості (EC)	12.96	5.85	9.45	14.85

Коефіцієнт «Витрати-переваги» (CBR)	відношення	0.49	0.51	0.04	0.04

Як видно з табл. 3.3, найкращим рішенням для імплементації підсистеми ЄІС МВС «Єдине розподілене сховище даних» є використання СУБД PostgreSQL. Відповідно до представлених розрахунків СУБД має найбільше значення коефіцієнту CBR.

Для визначення значень ризиків OCR, ICR та CCR використовується метод експертних оцінок. Для дослідження впливу одних проектних рішень на інші пропонується використати ентропійний підхід [21]. Розглянемо даний підхід.

Нехай для деякого проектного рішення A апіорі відома множина з n загроз безпеці проектування цього рішення, які впливають на результат і імплементацію дизайну та порушення вимог критичності і впорядкована множина з m станів збитку внаслідок реалізації цих загроз:

$$\begin{aligned} x_1, x_2, \dots, x_i, \dots, x_m, \\ i = \overline{1, m}. \end{aligned} \quad (3.4)$$

Очевидно, що $n \leq m$. Це вказує на існування не тотожних загроз безпеці, які призводять до однакового збитку. Прикладом цьому можуть бути загрози внаслідок реалізації яких збитки рівні нулю.

Крім цього, під впорядкованістю розуміється, що

$$0 \leq x_1 \leq x_2 \leq \dots \leq x_i \leq \dots \leq x_m \leq x_{\max}, \quad (3.5)$$

де $x_{\max}$ – максимальний збиток, рівний повному ліквідуванню усіх елементів проектного рішення A за нескінченно малий проміжок часу без будь-яких залишків. Крім цього вважатимемо, що відомий розподіл імовірностей p_i на множині x_i , а саме: збиток x_1 виникає з імовірністю p_1 , збиток x_2 – з імовірністю p_2 , збиток x_i – з імовірністю p_i .

З огляду на це, повною множиною подій $x_1, x_2, \dots, x_i, \dots, x_m$ назовемо таку множину станів збитку, що внаслідок реалізації загрози безпеці проектування рішення обов'язково наступить один з них. Оскільки стани збитку $x_1, x_2, \dots, x_i, \dots, x_m$ повної множини подій задані з їх імовірностями

$$p_1, p_2, \dots, p_i, \dots, p_m, \quad (3.6)$$

$$p_i \geq 0, \sum_{i=1}^m p_i = 1,$$

то вважатимемо заданою кінцеву схему

$$X = \begin{pmatrix} x_1 & x_2 & \dots & x_i & \dots & x_m \\ p_1 & p_2 & \dots & p_i & \dots & p_m \end{pmatrix}. \quad (3.7)$$

За допомогою кінцевої схеми описуватимемо стан невизначеності при забезпеченні результату та імплементації дизайну, виконання вимог щодо критичності дизайну. При цьому ступінь цієї невизначеності різна для різних схем. Тому для оцінювання ступеня невизначеності заданої кінцевої схеми використовується ентропійна міра [21]

$$H_A(p_1, p_2, \dots, p_i, \dots, p_m) = - \sum_{i=1}^m p_i \lg p_i, \quad (3.8)$$

де $H_A(p_1, p_2, \dots, p_i, \dots, p_m)$ – ентропія кінцевої схеми (див. рис. 3.2), яку пропонується використовувати для оцінювання ризиків OCR, ICR та CCR. Якщо одне зі значень імовірності дорівнює одиниці, то функція $H_A(p_1, p_2, \dots, p_i, \dots, p_m) = 0$. Цьому відповідає випадок, коли завчасно можна передбачити реалізацію загрози проектному рішенню з повною достовірністю і, як наслідок, відсутністю невизначеності. Тоді як при фіксованому m найбільша невизначеність описуватиметься кінцевою схемою з рівноймовірними реалізаціями загроз.

Крім цього, коректність використання ентропійного підходу для оцінювання ризиків підтверджується такими властивостями ентропії [187, 188]:

1. Величина

$$H_A(p_1, p_2, \dots, p_i, \dots, p_m) \geq 0. \quad (3.9)$$

Це означає, що ризики OCR, ICR та CCR можуть або дорівнювати нулю, або бути більшим за нього та, як наслідок, не можуть бути від'ємним. Цим пояснюється відмінність відповідних ризиків від комерційних ризиків, де може існувати від'ємний ризик, що еквівалентний доходу на противагу збиткам при позитивному ризику.

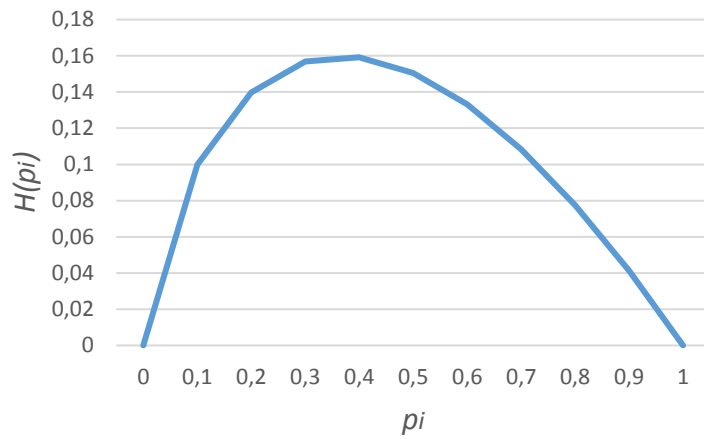


Рис.3.2. Графічне відображення залежності ентропії кінцевої схеми

2. Якщо кінцеві схеми двох проектних рішень A і B

$$X_1 = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1i} & \dots & x_{1m} \\ p_{11} & p_{12} & \dots & p_{1i} & \dots & p_{1m} \end{pmatrix}, \quad X_2 = \begin{pmatrix} x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2n} \\ p_{21} & p_{22} & \dots & p_{2i} & \dots & p_{2n} \end{pmatrix}$$

взаємно незалежні, то

$$H_{AB}(X_1 X_2) = H_A(X_1) + H_B(X_2). \quad (3.10)$$

Як наслідок, ризик двох проектних рішень A і B дорівнює сумі ризиків кожного з рішень. Це узгоджується з інтуїтивним уявленням про ризик.

3. Якщо кінцеві схеми двох проектних рішень A і B

$$X_1 = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1i} & \dots & x_{1m} \\ p_{11} & p_{12} & \dots & p_{1i} & \dots & p_{1m} \end{pmatrix}, \quad X_2 = \begin{pmatrix} x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2n} \\ p_{21} & p_{22} & \dots & p_{2i} & \dots & p_{2n} \end{pmatrix}$$

взаємно залежні, то

$$H_{AB}(X_1 X_2) = H_A(X_1) + H_B^{X_1}(X_2). \quad (3.11)$$

Як наслідок, ризик проектного рішення B зменшується, якщо відомий результат реалізації загрози кінцевої схеми проектного рішення A .

4. Якщо два проектних рішення A і B мають однакові розподіли ймовірностей нанесення збитку внаслідок реалізації загроз, то ризик для таких проектних рішень однаковий.

5. Якщо

$$p_1 = p_2 = \dots = p_i = \dots = p_m = \frac{1}{m}, \quad (3.12)$$

то величина $H_A(p_1, p_2, \dots, p_i, \dots, p_m)$ набуває найбільшого значення.

Інтерпретація цієї властивості така: якщо для даного проектного рішення нічого не відомо про ймовірності реалізації загроз, то ризик максимальний. Використання засобів і заходів його оброблення по будь якій загрозі зменшує імовірність нанесення збитку при відповідному збільшенні ймовірності нульового збитку. Як наслідок, величина $H_A(p_1, p_2, \dots, p_i, \dots, p_m)$ буде зменшуватися, що свідчить про зменшення ризику [188].

6. Для двох проектних рішень A і B визначеність вища у того проектного рішення, ризик якого менший. Якщо $H_A(X_1) > H_B(X_2)$, то різниця $[H(A) - H(B)]$ показує, наскільки дизайн проектного рішення B кращий, ніж проектного рішення A .

7. Якщо кінцева схема проектного рішення A доповнюється неможливою подією

$$H_A(p_1, p_2, \dots, p_i, \dots, p_m, 0) = H_A(p_1, p_2, \dots, p_i, \dots, p_m), \quad (3.13)$$

то ентропія і, як наслідок, ризик не змінюються.

Враховуючи вищенаведене, для кожного проектного рішення AR визначається наступна кінцева схема:

$$X = \begin{pmatrix} x_1 & x_2 & x_3 \\ OCR & ICR & CCR \end{pmatrix}. \quad (3.14)$$

За допомогою властивостей (3.10) і (3.11) архітектор має можливість оцінювати вплив одних проектних рішень на інші. Наприклад, оцінити вплив збою критичного елементу на весь дизайн в цілому.

3.2 Оцінка та вибір оптимальної конфігурації компонент критичної інформаційної інфраструктури за допомогою Марківського процесу прийняття рішень

В даному розділі запропоновано модель підтримки прийняття рішень для забезпечення прийняття відповідних проектних рішень, що співпадають з

стратегією підприємства з точки зору централізації/децентралізації, в яку включено знання про майбутні недоліки/переваги кожного проектного рішення на базі запропонованих критеріїв при проектуванні ІТ-інфраструктури.

Пропонована модель має наступні *параметри* [8]:

F – скінченна множина філій підприємства ($F = \{1, \dots, f\}$);

P – скінченна множина платформ ІТ ($P = \{1, \dots, p\}$);

T – скінченна множина відліків часу ($T = \{1, \dots, t\}$);

I – скінченна множина критеріїв проектування;

Z – множина категорій запитів на зміну архітектури ($Z = \{1, \dots, z\}$);

$f \in F$ – значення індексу філії підприємства;

$p \in P$ – значення індексу платформи ІТ;

$z \in Z$ – значення індексу запиту на зміну архітектури;

$i \in I$ – значення індексу критерію проектування;

$b_t \in I$ – бюджет нового проекту в момент часу t ;

c_{fp} – вартість проектування/переходу на нову платформу p для філії f з врахуванням всіх витрат на ПЗ, апаратні засоби, інтеграцію та впровадження;

a_{fpz} – вартість виконання запиту на зміну архітектури z на платформу p для філії f ;

b_{fpzi} – виграш від виконання запиту на зміну архітектури z на платформу p для філії f за критерієм i .

Простір станів моделі описується наступними змінними:

x_{zf} – кількість запитів на зміну архітектури z , що ще не виконані для філії f ;

cur_{fp} – поточна платформа філії f ;

X – матриця значень x_{zf} ;

CUR – матриця значень cur_{zf} ;

S – стан процесу ($S = [X, CUR, t]$).

Випадкові величини, що використовуються в моделі:

pr_{zft} – кількість запитів на зміну архітектури z для філії f в момент часу t

;

PR – матриця значень pr_{zft} .

Простір рішень моделі описується наступними змінними:

y_{zft} – кількість запитів на зміну архітектури z для філії f в момент часу t ,

що потрібно виконати;

l_{fpt} – флаг переходу на платформу p для філії f в момент часу t , що

потрібно виконати;

Y – масив змінних простору рішень;

$\mathfrak{R}(S)$ – множина можливих рішень для стану S ;

$C^i(Y)$ – виграш за критерієм i , пов'язаний з рішенням Y ;

$C(Y)$ – виграш за всіма критеріями, пов'язаний з рішенням Y ;

$RWD_n^i(S)$ – максимальне очікуване значення виграшу на n -му етапі в стані

S за критерієм i ;

$RWD(S)$ – максимальне очікуване значення виграшу на n -му етапі в стані

S за критерієм i .

Модель має певний ряд обмежень. Для стану $S = [X, CUR, t]$ змінні простору станів в Y повинні задовольняти наступні обмеження:

– обмеження бюджету проекту:

$$\sum_f \sum_p c_{fp} l_{fpt} + \sum_z \sum_f a_{zpf} y_{zft} \leq b_t;$$

(3.15)

– обмеження об'єму проекту:

$$y_{zft} \leq x_{zf}; \tag{3.16}$$

– вимоги щодо платформ:

$$\sum_p l_{fpt} = 1, \forall f, \quad (3.17)$$

$$y_{zft} \geq 0. \quad (3.18)$$

Всі рішення Y , що задовольняють вимогам (3.15)–(3.18), для стану S формують множину можливих рішень $\mathfrak{R}(S)$. Модель є гнучкою. Можна додати додаткові обмеження. Наприклад, можна врахувати виконання проектних рішень, що забезпечують цілісність системи та її безпеку.

Кожне можливе прийняте рішення має певну кількість безпосередньо очікуваних витрат та вигравів. По-перше, очікуваний виграв b_{fzi} за критерієм i при виконанні запиту на зміну архітектури z для філії f . Також підприємство бере на себе витрати a_{fz} на виконання запиту на зміну архітектури z для філії f . Додатково, можуть також бути витрати c_{fp} , пов'язані з міграцією філії f на платформу p .

Враховуючи наведене вище, виграв на черговому етапі проектування за критерієм i , пов'язаний з рішенням Y , можна розрахувати за формулою:

$$C^i(Y) = \sum_z \sum_f \sum_i b_{zfi} y_{zft} - \sum_f \sum_p c_{fp} l_{fpt} - \sum_z \sum_f a_{fz} y_{fzt}. \quad (3.19)$$

Значення $C^i(Y)$ може бути як позитивним, так і негативним. Якщо $\sum_z \sum_f \sum_i b_{zfi} y_{zft} > \sum_f \sum_p c_{fp} l_{fpt} + \sum_z \sum_f a_{fz} y_{fzt}$, то виграв, пов'язаний з вибраними проектними рішеннями, представлений першим термом функції виграву на черговому етапі проектування, переважає витрати, які потрібно зробити.

Невизначеність поставленої задачі полягає у частоті запитів на зміну від кожної філії. В даній роботі вважається, що значення PR є статично

незалежними. Нехай $S = [X]$, $[CUR]$ – поточний стан, $Y \in \mathfrak{R}(S)$ – вибраний масив рішень і $S' = [X']$, $[CUR']$ – стан після виконання запиту на зміну. Тоді значення стану S' змінюється відповідно до (3.20)–(3.22):

$$x'_{zf} = x_{zf} - y_{zft} + pr_{zft}, \quad (3.20)$$

$$cur'_{fp} = l_{fpt}, \quad (3.21)$$

$$t' = t + 1. \quad (3.22)$$

Ймовірність переходу з стану S в стан S' для рішення Y визначається як (3.23):

$$P_{SS'(Y)} = \prod_{f \in F} \prod_{z \in Z} \{ pr_{zft} \mid cur_{fp} = x'_{zf} - x_{zf} + y_{zft} \}. \quad (3.23)$$

Функції пошуку моделі наступні:

$$RWD_1^i(S) = \max_{Y \in \mathfrak{R}(S)} C^i(Y), \quad (3.24)$$

$$RWD_n^i(S) = \max_{Y \in \mathfrak{R}(S)} \left\{ C^i(Y) + \sum_{S'} P_{SS'}(Y) RWD_{n-1}^i(S') \right\}, \quad n > 1, \quad (3.25)$$

$$RWD(S) = \max_{Y \in \mathfrak{R}(S)} \sum_i \sum_j^n RWD_j^i(S). \quad (3.26)$$

При роботі з критичними інформаційними інфраструктурами, оцінка вибору конкретного варіанту реалізації архітектури є однією з проблем, з якою стикаються всі методи, які використовуються для проектування. Запропонована модель є водночас модульною та масштабованою в тому сенсі, що має достатню гнучкість у виборі та використанні як простих, так і складних критеріїв вибору архітектури для критичної інформаційної інфраструктури. Модульність досягається за рахунок використання різних конфігурацій елементів, тоді як масштабованість представлена у двох формах:

– масштабованість при побудові моделі (топология і функціональність) критичної інформаційної інфраструктури;

– масштабованість з точки зору використання різного роду критеріїв, необхідних для порівняння варіантів реалізації. З точки зору моделювання, запропонований підхід дозволяє створювати моделі оцінки варіантів реалізації на

базі різних критеріїв, які можна далі використовувати як вхідні моделі для подальшого порівняння за допомогою інших критеріїв, що в свою чергу, дає можливість знайти оптимальну архітектуру критичної інформаційної інфраструктури. Таким чином, модель дозволяє накопичувати моделі оцінки варіантів реалізації для багаторазового використання.

Модель побудована на використанні Марківського процесу прийняття рішень і надає всі необхідні інструменти для побудови, планування, дослідження, управління, оцінювання варіантів архітектури критичних інформаційних інфраструктур.

Результати використання запропонованої моделі наведено в розділі 5.

3.3 Оцінка впливу забезпечуючих систем на архітектуру критичної інформаційної інфраструктури

Хоча кожна критична інформаційна інфраструктура зазвичай розглядається як окрема система, всі її системи сильно взаємопов'язані з різним рівнем взаємозалежності між ними. Як приклад, для роботи інформаційно-телекомунікаційної системи (ІТС) на рівні заявленої якості обслуговування потрібна безперебійна робота системи постачання електроенергії, в той час як якість роботи самої системи електропостачання залежить від стабільної роботи каналів передачі інформації ІТС. Ці двонаправлені відносини між системами критичної інформаційної інфраструктури підвищують їх загальну продуктивність, але в той же час збільшують її складність та чутливість до різного роду атак [189, 190].

Моделювання взаємозалежності систем – це новий науковий напрямок, що включає кілька інноваційних підходів до моделювання. Існуючі моделі проаналізовані в працях [191, 192]. Серед найпопулярніших – це методи введення-виведення, агентне моделювання та мережеві підходи.

Методи введення-виведення ґрунтуються на теорії економічної рівності В. Леонтьєва і дозволяють оцінити цілісний рівень непрацездатності (відсоток

несправності) інфраструктур шляхом використання коефіцієнтів залежності (коефіцієнтів Леонтьєва). Однак ці коефіцієнти важко правильно визначити, і тому, як правило, вони є наближенням більш високого рівня, виходячи з припущення про те, що взаємозалежність інфраструктур пов'язана з їхньою економічною взаємодією [193].

Методи агентного моделювання (АМ) розглядають критичні інфраструктури як гнучкі адаптивні системи (ГАС), тобто як комплекс взаємодіючих компонентів, стан яких може змінюватися в процесі навчання. Методи АМ використовують стратегію проектування знизу-вгору і тому різні компоненти ІТ-інфраструктури представлені як автономні агенти з своїми атрибутами, поведінкою та правилами прийняття рішень, в той час як взаємозалежності між ними виникають при їх взаємодії [90, 194].

Мережеві підходи зазвичай припускають, що кожна інфраструктура складається з множини мережевих компонентів (як правило, представлених у вигляді вузлів), що утворюють мережу, і будь-які існуючі залежності представлені як відносини між вузлами, що належать до різних мереж [195]. Використання мережевих моделей для дослідження критичних взаємозалежних інфраструктур дає можливість досить легко виконати топологічний аналіз (тобто, якісно описати існуючі зв'язки для будь-якого набору компонентів). Недоліком даних моделей є досить мала кількість інформації для проведення функціонального аналізу. Зазвичай, такі моделі дають можливість дослідити лише спрощені гіпотези та отримати тільки базові характеристики мереж та повністю відсутня можливість дослідити складні ефекти, пов'язані з технологічними аспектами їх реалізації [196].

Також є ряд інших підходів до моделювання взаємозалежності систем критичних інфраструктур. Наприклад, в працях [197–199] представлені методи, побудовані на мережах Петрі, мережах стохастичної активності і байєсівських мережах.

На даний момент підходи, що представлені в літературі, використовуються для різних цілей, мають свої сильні і слабкі сторони, але як такий, єдиний підхід

до вирішення проблеми відсутній. Крім того, труднощі, пов'язані з доступом до даних через їх конфіденційність та приватність, в поєднанні з тим, що структура критичної інформаційної інфраструктури стає різноманітнішою та складнішою, робить проблему перевірку взаємозалежності її компонентів та систем дуже нетривіальною проблемою. Отже, є потреба для подальшого розвитку підходів дослідження взаємозалежності компонентів, систем та цілих інфраструктур.

Розширений відкритий гібридний автомат (РВГА). В даній роботі пропонується використати модифікацію РВГА та методу РТС для проектування архітектури критичної інформаційної інфраструктури [7].

Нехай критична інформаційна інфраструктура S представлена у вигляді сукупності систем і компонент Ω . Тоді, представимо нашу Ω у вигляді РВГА [10]:

$$\Omega = (D, S, S_0, I, O, Z, L, G, T, \tau, F, SP, P, R, V), \quad (3.27)$$

де D – скінченна множина дискретних станів системи (компоненти) критичної інформаційної інфраструктури Ω . Множина поділяється на наступні множини: D_s – множину безпечних станів, D_c – множину критичних станів та D_t – множину термінальних станів;

S – скінченна множина неперервних станів системи (компоненти) критичної інформаційної інфраструктури Ω . Множина поділяється на наступні множини: S_s – множину безпечних станів, S_c – множину критичних станів та S_t – множину термінальних станів;

$S_0 \subseteq S \times D$ – скінченна множина початкових станів, $s_0 \in S_s, d_0 \in D_s$;

$I = \Xi \cup \Psi$ – скінченна множина входів, яка поділяється на: Ξ – множина внутрішніх входів (в рамках одного компонента), Ψ – множина зовнішніх входів (між компонентні входи);

O – скінченна множина вихідних станів.

Запис $(s, d) \in S \times D$ описує зміну стану компоненти Ω , яка має:

– початковий стан $S_0 \subseteq S \times D$;

– динаміку зміни станів, що описується вектором $\phi: S \times D \times I \rightarrow R^n$;

- функцію виходу $\varphi: S \times D \times I \rightarrow O$;
- множину дозволених станів та входів $Z \rightarrow 2^{S \times I}$;
- $L \subseteq D \times D$ – скінченну множину відміток переходів, що включає спеціальний символ $\dagger$;
- множину умов $G: L \rightarrow 2^{S \times I}$, що ініціюють перехід між дискретними станами;
- відношення скидання $T: L \times S \times S$, що скидає значення входу $s \in S$ перед кожним переходом;
- τ – диспетчер часу;
- F – розподіл втручань в роботу компонентів;
- SP – набір специфікацій;
- P – набір політик;
- R – набір вимог безпеки;
- V – набір вразливостей.

Перехід є детермінованим та відбувається за умовою G у випадку, коли $I \in L \setminus \{\dagger\}$, або ймовірнісним, у випадку, коли $I = \dagger$. В останньому випадку, значення стану формується випадково згідно розподілу F .

Взаємодіючими учасниками в такій моделі є:

- компоненти (телекомунікаційні, промислові і т. ін.);
- системи;
- інфраструктури;
- оператори систем критичної інформаційної інфраструктури;
- супротивники;
- середовище.

Їх логіка функціонування описується наборами специфікацій SP , політиками P та вимогами безпеки R .

Середовище контролює часові та просторові аспекти всіх подій в моделі та диспетчеризує всі зміни станів відповідно до τ , використовуючи для цього розподіли F . Розподіл дає можливість створювати стратегії виходу з ладу

доступних компонентів та використовується для вирішення проблем одночасного виникнення подій в критичній інформаційній інфраструктурі та їх обробки [7].

Таку модель дуже зручно представити у вигляді направленої графу (рис. 3.3).

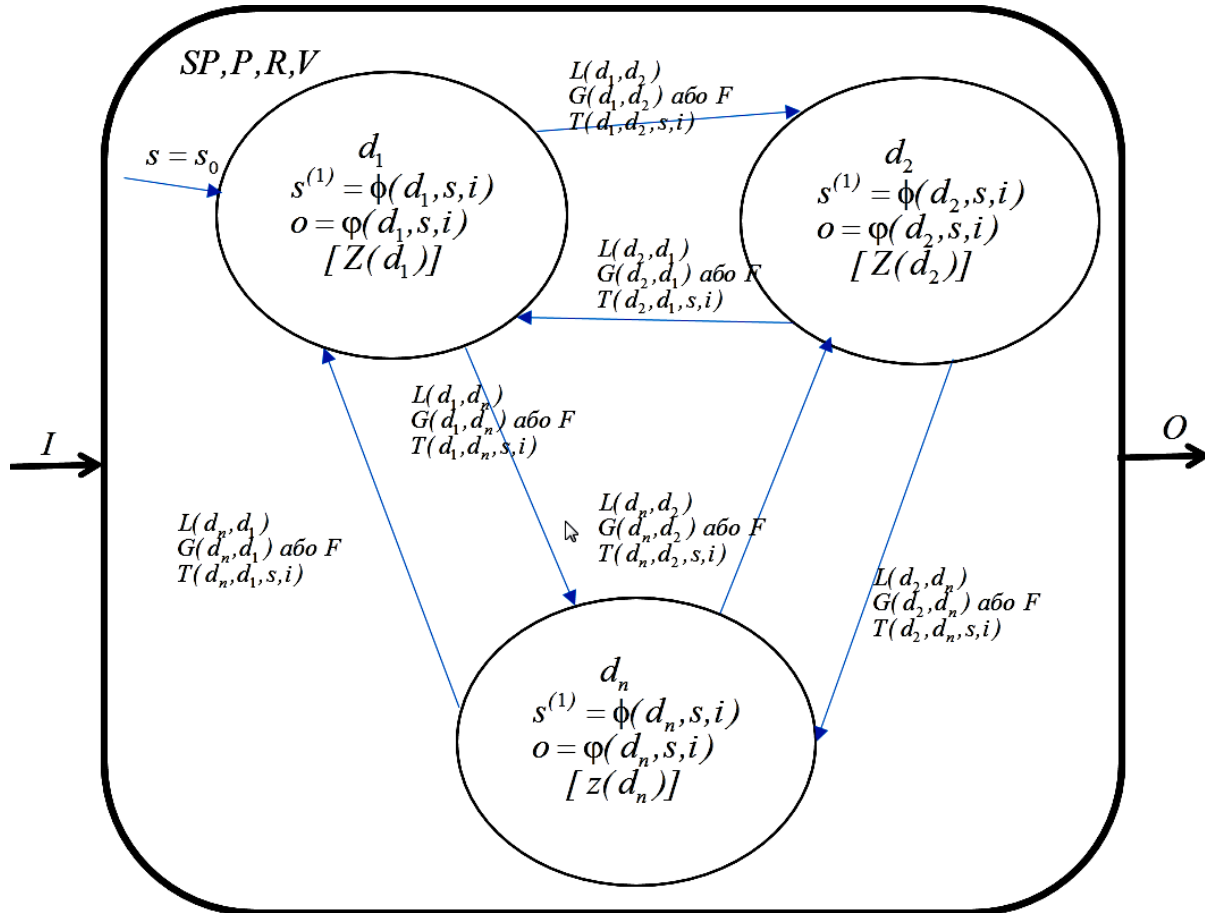


Рис. 3.3. Модель у вигляді направленої графу

Кожна вершина такого графу представляє собою дискретний стан $d \in D$. Ребра направленої графу представляють собою дискретні переходи між станами. Наприклад, ребро $(d_1, d_2) \in L$ починається в вершині $d_1 \in D$ і закінчується в вершині $d_2 \in D$. Кожний перехід відбувається при виконанні умови $G(d_1, d_2)$ або випадково, якщо $L(d_1, d_2) = \dagger$. В кінці переходу, при зміні значення неперервного стану, відбувається скидання відношення T .

Простий шлях (спрацювання РВГА) складається з послідовності інтервалів τ безперервної еволюції, що змінюються дискретними переходами [10].

Виконання починається з деякого початкового стану $(d_0, s_0) \in S_0$. Модель залишається в дискретному стані d_i доки неперервний стан $s_i \in S$ та/або значення входу $i \in I$ мають допустимі значення Z . В той же час, значення виходу $o \in O$ визначається як $\varphi(s_i, d_i, i)$. Якщо $s_i \in S$ та/або значення входу $i \in I$ досягає умови переходу $G(d_i, d_j)$, то зміна стану відбувається миттєво, а значення безперервного стану визначається шляхом відношення T .

Кожну критичну інформаційну інфраструктуру можна представити як композицію різних РВГА. На рис. 3.4 представлена композиція двох РВГА.

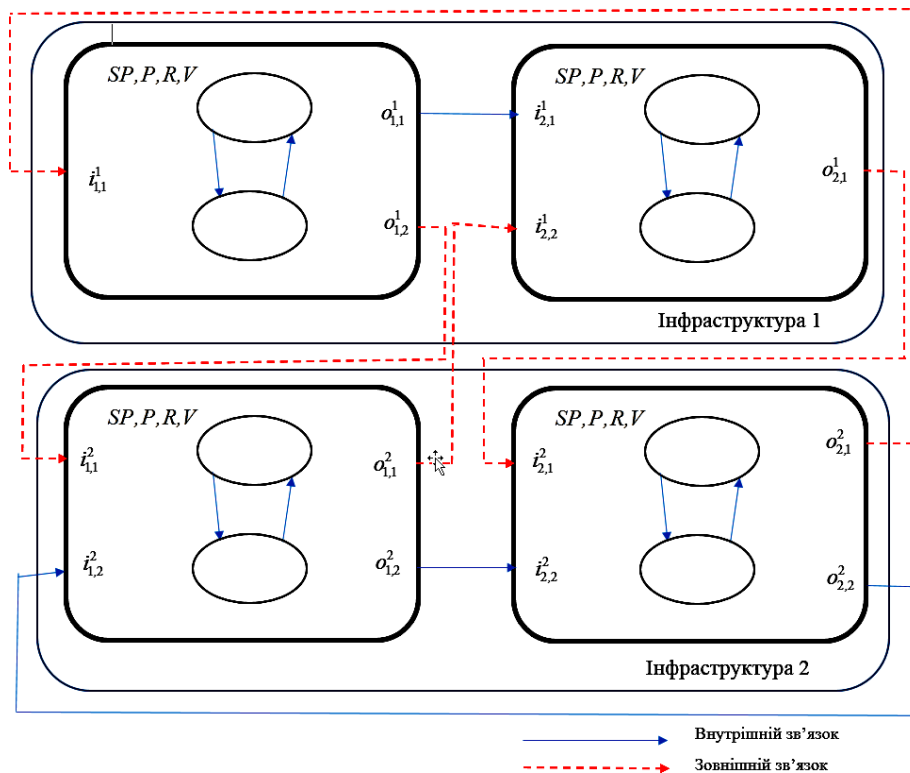


Рис. 3.4. Композиція автоматів

При роботі з критичними інформаційними інфраструктурами, масштабованість є однією з проблем, з якою стикаються всі методи, які використовуються для моделювання взаємозалежності. Запропонований підхід є водночас модульним та масштабованим в тому сенсі, що має достатню гнучкість у виборі та використанні як високоточних, так і простих моделей для критичної інформаційної інфраструктури. Модульність досягається за рахунок використання композиції елементів, тоді як масштабованість представлена у двох формах: масштабованість при побудові моделі (топология і

функціональність) критичної інформаційної інфраструктури та масштабованість з точки зору обчислювальної потужності, необхідної для запуску моделей. З точки зору моделювання, запропонований підхід дозволяє створювати композиції моделей, які можна далі використовувати як компоненти вищого рівня, які в свою чергу, можуть бути використані для створення компонентів ще вищого рівня тощо. Таким чином, підхід дозволяє накопичувати портфоліо компонентів для багаторазового використання. Наприклад, електричну підстанцію можна представити як набір кількох генераторів. Таким чином, можна побудувати модель одного генератора, а потім повторно її використати для створення моделі електричної підстанції. А якщо додати ще в цей набір вузли безперебійного живлення, то отримаємо модель електричної інфраструктури як частини більш складної критичної інформаційної інфраструктури. З точки зору необхідної обчислювальної потужності для запуску моделі підхід надає ту ж саму можливість формування необхідного рівня абстракції.

Фактично, запропонований підхід побудовано на математичному апараті відкритих гібридних автоматів, який модифіковано шляхом введення додаткових елементів, що значно розширюють його функціональність. Множина міток переходів $L \subseteq D \times D$ дозволяє формувати РТС компонент та систем критичної інформаційної інфраструктури для подальшого їх дослідження на предмет досяжності та безпечності станів. Розподіл F дозволяє додати ймовірнісний характер в поведінку елементів, а набори SP, P, R, V - надати компонентам та системам критичної інформаційної інфраструктури якісні характеристики.

Використання запропонованої моделі на прикладі ІТС наведено в розділі 5.

3.4 Висновки до третього розділу

У даному розділі представлений детальний опис методу обґрунтування архітектури критичної інформаційної інфраструктури на базі ентропійного

підходу, наведено основні базові елементи математичного апарату даного підходу, використаних для опису моделей даного методу обґрунтування архітектури у формалізованому вигляді. Запропонований метод дозволяє автоматизувати процес обґрунтування архітектури критичної інформаційної архітектури в досить зручний спосіб та дає можливість досліднику відстежувати вплив змін окремого елемента архітектури на всю архітектуру в цілому. Далі наведений опис моделі оцінки та вибору оптимальної конфігурації компонент критичної інформаційної інфраструктури за допомогою Марківського процесу прийняття рішень, яка дозволяє обирати оптимальну конфігурацію компонент на базі визначених критеріїв. І на останок, в розділі наведена модель оцінки впливу забезпечуючих систем на архітектуру критичної інформаційної інфраструктури, побудовану на базі розширених гібридних відкритих автоматів, яка дозволяє оцінити вплив забезпечуючих систем на архітектуру критичної інформаційної інфраструктури з можливим врахуванням ймовірнісного характеру поведінки окремих її елементів та деяких якісних показників, що можуть впливати на критичну інформаційну інфраструктуру.

РОЗДІЛ 4

МОДЕЛІ І МЕТОДИ ВДОСКОНАЛЕННЯ КРИТИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ

4.1 Аналіз стану і оцінювання показників елементів проектних рішень критичної інформаційної інфраструктури з використанням методу структурної оптимізації нейронних мереж при навчанні

4.1.1 Алгоритм структурної оптимізації нейронних мереж

Алгоритм структурного навчання використовується на багат шарових мережах прямого поширення та має ітеративний характер: на кожній ітерації виконується пошук структури мережі кращої за попередню. Пошук мережі виконується шляхом перебору усіх можливих мутацій мережі, вибору та комбінації кращих (селекція та схрещення) [18].

Розглянемо основні параметри алгоритму.

Параметри навчання:

- швидкість навчання, η ;
- коефіцієнт інерції, μ ;
- коефіцієнт затухання ваг, ε ;
- вірогідність активації нейрону прихованого шару, p_h ;
- вірогідність активації нейрону вхідного шару, p_i .

Параметри структурного навчання:

- початкова кількість нейронів прихованого шару;
- функція активації прихованого шару;
- функція активації та функція ціни вихідного шару;
- максимальна кількість мутацій при схрещенні;
- кількість епох навчання початкової мережі;
- кількість епох навчання у ітерації;

- види допустимих мутацій;
- частина навчальної вибірки, що використовується для навчання.

Елементарні структурні операції над нейронною мережею. Згідно з [9] введено наступні елементарні структурні операції над мережею:

- додавання синапсу між двома випадково вибраними незв'язаними вузлами або нейронами мережі – операція Syn_{ADD} ;
- видалення синапсу між двома випадково вибраними незв'язаними вузлами або нейронами мережі – операція Syn_{DEL} ;
- переміщення синапсу між двома випадково вибраними незв'язаними вузлами або нейронами мережі – операція Syn_{MOD} ;
- зміна функції активації нейрона для випадково вибраного нейрона – операція A_{MOD} ;
- серіалізація вузла або нейрона – операції - Ser_{NODE} і Ser_{NR} ;
- паралелізація вузла або нейрона – операції Par_{NODE} і Par_{NR} ;
- додавання вузла або нейрона – операції Add_{NODE} і Add_{NR} ;
- створення нового шару – операція L_{ADD} ;
- видалення шару НМ – операція L_{DEL} .

Використання чи невикористання наведених структурних операцій залежить від складності поставленої задачі.

Для задач розпізнавання, що будуть описані в розділі 5, використані наступні операції (мутації) [17, 39]:

- пуста (змін в структурі не відбувається) (рис. 4.1);

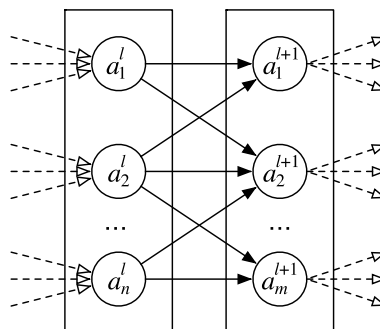


Рис.4.1. Пуста мутація

- видалення синапсу між випадково вибраними нейронами (рис. 4.2);

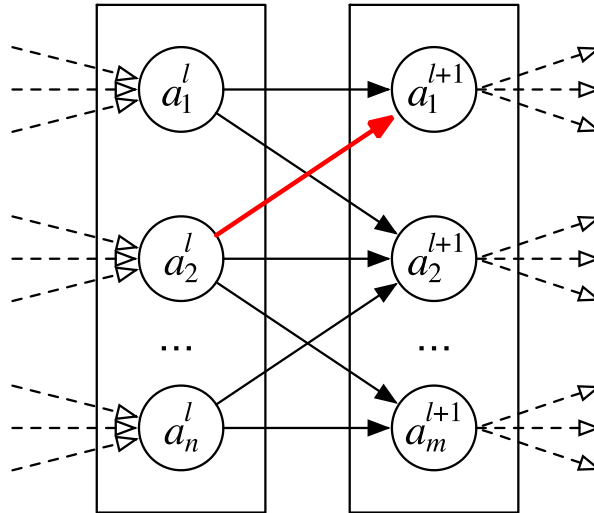


Рис.4.2. Видалення синапсу

- додавання синапсу між випадково вибраними нейронами (рис. 4.3);

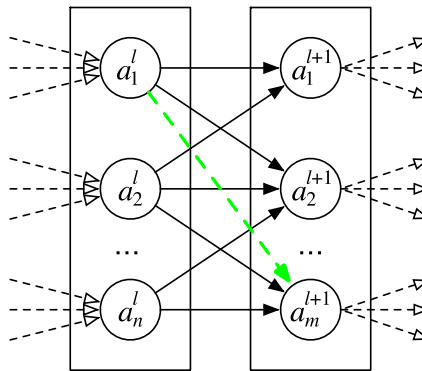


Рис. 4.3. Додавання синапсу

- видалення випадково вибраного нейрону прихованого шару та усіх його синапсів (рис. 4.4);

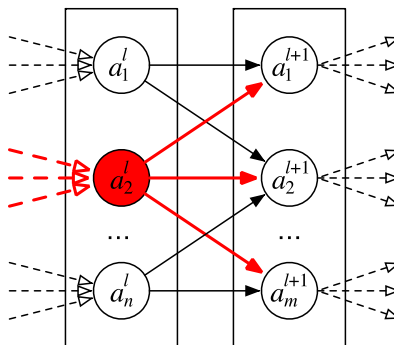


Рис. 4.4. Видалення нейрону

- додавання нового нейрону у вихідному шарі (рис. 4.5);

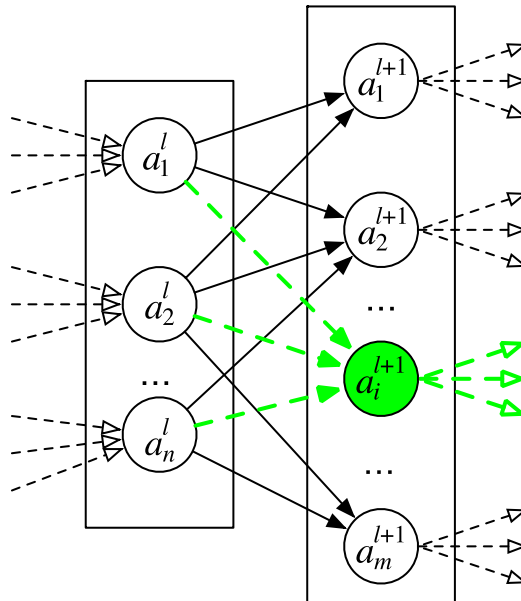


Рис. 4.5. Додавання нейрону

- додавання нового прихованого шару між випадково вибраними сусідніми шарами (рис. 4.6).

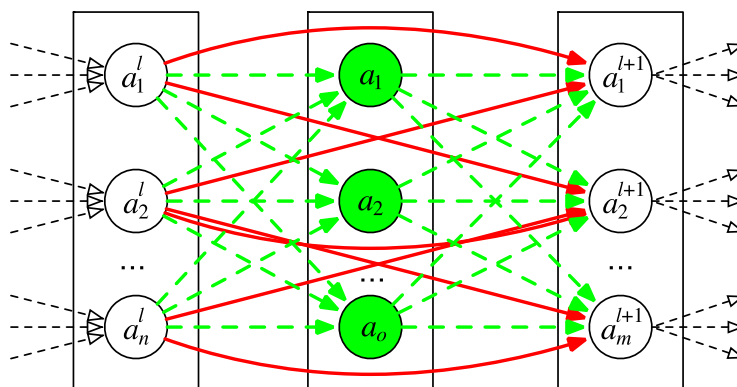


Рис. 4.6. Додавання нового прихованого шару

Під час додавання нового прихованого шару руйнуються всі поточні зв'язки між сусідніми шарами [28].

Алгоритм структурної оптимізації при навчанні. В якості базового алгоритму навчання візьмемо алгоритм зворотного поширення помилки, запропонований в [9, 200].

На початку роботи алгоритму створюється мережа із одним прихованим шаром, заданою кількістю нейронів та функцією активації. Оскільки початкові

значення синаптичних ваг вибираються довільним чином, початкова мережа проходить задану кількість навчальних епох.

У кожній ітерації алгоритму виконується пошук усіх можливих мутацій поточної мережі: видалення кожного синапсу, видалення кожного нейрону усіх прихованих шарів і т. д. Далі, мережі, отримані внаслідок виконання кожної мутації, навчаються протягом заданої кількості епох незалежно одна від іншої.

Після навчання мереж усіх можливих мутацій виконується вибір та схрещення кращих з них для отримання комбінованих мутацій. Як критерій для порівняння мереж використовується значення ціни на тренувальній вибірці. Для створення комбінованих мутацій використовуються лише ті мережі, значення цін яких є меншими за ціну пустої мутації.

Після отримання комбінованих мутацій вони порівнюються та вибирається найкраща. Ця мережа і є результатом роботи ітерації.

Алгоритм припиняє роботу якщо отримана мережа є гіршою (має більше значення ціни) за мережу в попередній ітерації.

Якщо мережа є кращою, відбувається перехід до наступної ітерації.

Імплементація алгоритму. Внутрішньо нейронні мережі представлені у вигляді послідовності числових матриць ваг кожного шару окрім вхідного [16]. На рис. 4.7 зображена послідовність матриць для мережі типу [2-3-2]: матриця 2×3 для прихованого шару та 3×2 для вихідного.

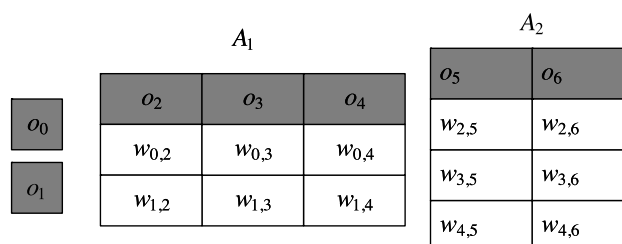


Рис. 4.7. Приклад внутрішньої реалізації для мережі [2-3-2]

Кожен елемент a_{ij} матриці A_k дорівнює значенню ваги між i -тим та j -тим нейронами мережі.

Для реалізації різних видів мутацій використовуються операції над матрицями. При додаванні нових нейронів до шару виконується комбінація із

операцій додавання нового рядка та стовпця матриці. На рис. 4.8, 4.9 та 4.10 зображена реалізація додавання нейрону до вхідного, прихованого та вихідного шарів.

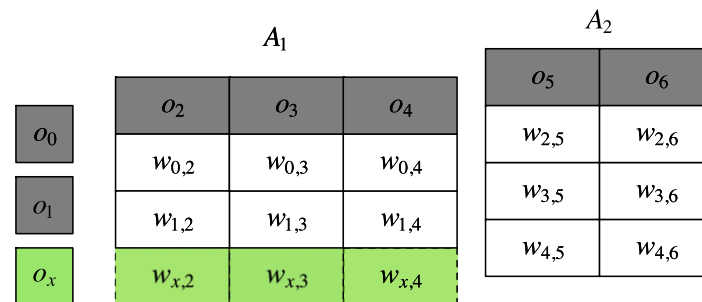


Рис. 4.8. Додавання нейрону до вхідного шару

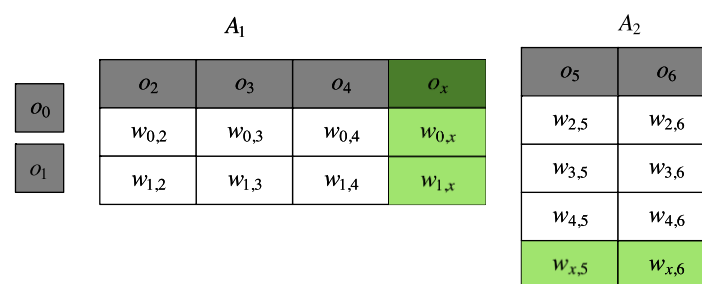


Рис. 4.9. Додавання нейрону до прихованого шару

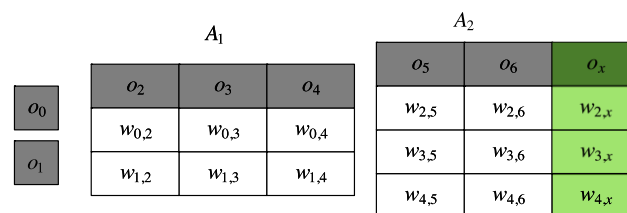


Рис. 4.10. Додавання нейрону до вихідного шару

При видаленні нейронів використовуються протилежні операції. На рис. 4.11 зображена реалізація видалення 2-го нейрону прихованого шару.

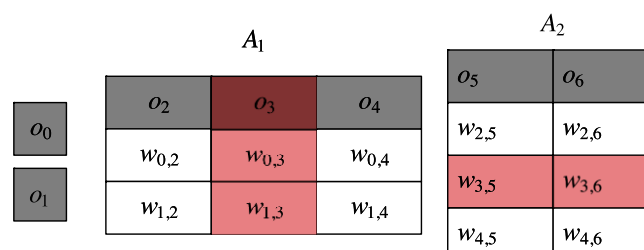


Рис. 4.11. Видалення нейрону прихованого шару

При додаванні нового шару виконується операція вставки нової матриці ваг у послідовність матриць.

Оскільки певні операції змінюють структури матриць, існує певна складність при їх комбінації. Наприклад, при видаленні нейрону o_3 прихованого шару у мережі [2-3-2] у результуючій мережі нейрон o_4 зміститься на 1 позицію та стане нейроном o_3 ; при додаванні нового прихованого шару, що містить 4 нейрони перед існуючим прихованим шаром, наступний шар зміститься на 1 позицію. При комбінації різних мутацій їх покрокове виконання необхідно здійснювати у чіткому порядку, що залежить від виду та параметрів кожної мутації. У лістингу 4.1 нижче наведений фрагмент коду, який правильно виконує комбіновану мутацію. Першим чином виконуються мутації, що не змінюють структури – додавання та видалення зв'язків, згодом виконуються додавання нових нейронів та видалення існуючих; наприкінці виконуються додавання шарів. Мутації, що виконують видалення нейронів, виконуються у порядку зменшення номеру нейрону, аналогічно додавання шарів – у порядку зменшення індексу нового шару.

```
(defmethod mutate ::combined
  [net {:keys [mutations]})
  (let [grouped-ms (group-by :operation mutations)
        {add-node-ms ::add-node del-node-ms ::del-node
         layer-ms ::add-layer} grouped-ms
        safe-ms (mapcat grouped-ms [::identity ::add-edge ::del-edge])
        safe-del-node-ms (reverse
                           (sort-by #(second (:deleted-node %)) del-
node-ms))
        safe-layer-ms (reverse
                       (sort-by :layer-pos layer-ms))
        ms (concat safe-ms add-node-ms safe-del-node-ms safe-layer-ms)]
    (reduce mutate net ms)))
```

Лістинг 0.1. Фрагмент коду, що здійснює комбіновану мутацію

4.1.2 Оцінювання показників якості функціонування елементів критичної інформаційної інфраструктури з використанням методу структурної оптимізації нейронних мереж при навчанні

На рис. 4.12 схематично відображені елементи критичної інформаційної інфраструктури (елементи CAE та AR), їх окремі характеристики та напрямки

впливу одних елементів на інші. Тут CAE_j^i , $i \in [1; m]; j \in [1; N_i]$ AR_j^i , $i \in [1; l]; j \in [1; N_i]$, – (елементи CAE та AR), а стрілками відображений вплив одних елементів на якість параметрів проектування інших. На даному графі можна побачити, як елемент CAE_2^1 впливає на два елементи – AR_1^1 та AR_2^1 .

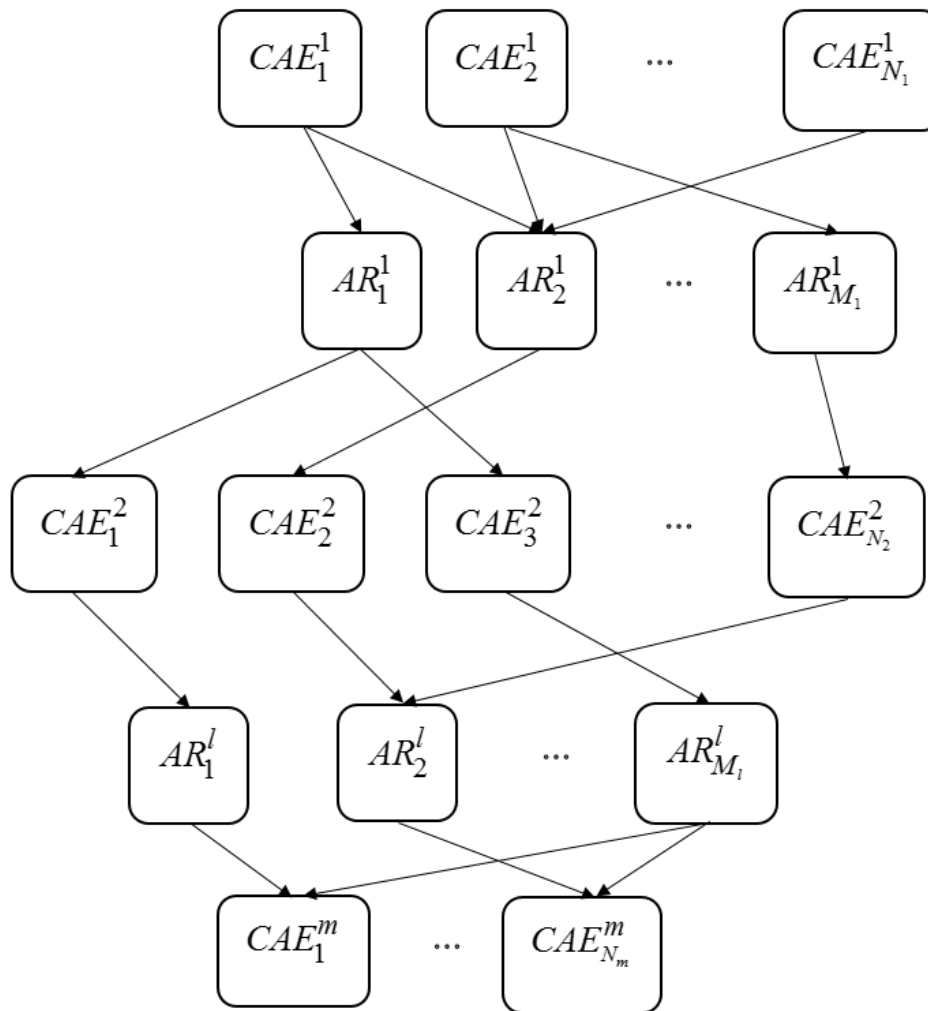


Рис. 4.12. Елементи критичної інформаційної інфраструктури

Такий граф будується архітектором автоматично на базі UML-діаграм проектного рішення, він же і визначає, які характеристики елементів критичної інформаційної інфраструктури необхідно враховувати при розрахунках якісної оцінки вибору включення даних елементів до проектного рішення і їх параметрів.

Якісна оцінка вибору включення даних елементів розраховується на підставі параметрів елементів. Значення таких параметрів, в свою чергу, можуть

визначатися на підставі функціонування, як даного елемента, так і елементів, що впливають на нього.

Позначимо параметри, що впливають на якість елемента, як

$$P_j^i = \{p_{j,1}^i, p_{j,2}^i, \dots\} \quad (4.1).$$

Так як, природа таких параметрів може бути різною, то, для зручності, розіб'ємо їх на дві групи і, в подальшому, будемо розглядати кожну таку групу окремо:

1) параметри, значення яких визначені розробником, завданнями, які необхідно виконати, умовами, в яких елементи виконують свої завдання. Назвемо їх власними параметрами елементів критичної інформаційної інфраструктури;

2) параметри елемента критичної інформаційної інфраструктури, значення яких безпосередньо залежать від значень параметрів інших елементів. Назвемо такі параметри залежними параметрами елементів критичної інформаційної інфраструктури.

Для прикладу, візьмемо такий елемент критичної інформаційної інфраструктури, як сервер застосувань [15]. Нехай, його характеризують такі параметри як:

- 1) максимальна частота процесора;
- 2) максимальна швидкість передачі даних по мережі;
- 3) максимальних обсяг оперативної пам'яті;
- 4) зайнятий обсяг оперативної пам'яті.

Очевидно, що перший і третій параметри визначені ще при виготовленні комплектуючих даного сервера і вони не залежать ні від яких інших чинників. Тому, їх можна віднести до власних параметрів даного елемента. У свою чергу, максимальна швидкість передачі даних по мережі, залежить не тільки від обладнання, встановленого на даному сервері, але і від мережевого обладнання, яке дає доступ до мережі сервера. Таким чином, такий параметр відноситься до групи залежних параметрів елементів критичної інформаційної інфраструктури.

Зайнятий обсяг оперативної пам'яті визначається завданнями, які на даний момент виконує даний сервер. Специфіка таких завдань, як правило, така, що їх виконання не залежить від інших елементів даної критичної інформаційної інфраструктури, відповідно, даний параметр можна віднести до групи своїх властивостей. Але, в разі, коли на зайнятий обсяг оперативної пам'яті впливають будь-які інші чинники, то його необхідно відносити до залежних параметрів. Як і який параметр елемента критичної інформаційної інфраструктури впливає на інші параметри визначається архітектором при побудові проектного рішення [15].

У зв'язку з відсутністю припущень про характер залежностей між елементами і якісними оцінками параметрами елементів, а також можливості існування будь-якого роду відносин між компонентами проекту, доцільно застосовувати наближені експертні оцінки, засновані на особистому досвіді архітектора, адміністраторів, керівників ІТ-підрозділів і ін.

Так як, найчастіше, відповідальним за побудову таких залежностей, важко, а часом і неможливо, налаштувати установки приналежності, а також, побудувати базу правил для подальшої роботи моделі, в пропонованому методі буде використовуватися апарат нейронних мереж. Нейронні мережі в процесі навчання самі визначають базу правил. Таким чином, буде скорочено втручання людей у навчання моделі, яку необхідно побудувати. Для навчання такої нейронної мережі необхідно буде вказати якісну оцінку функціонування конкретних елементів критичної інформаційної інфраструктури, в залежності від заданих значень параметрів.

Кожному параметру $p_{j,m}^i \in P_j^i$, а також самим елементам CAE_j^i та AR_j^i буде відповідати лінгвістична змінна з лінгвістичними значеннями «відмінно», «задовільно», «незадовільно» [31].

Вони будуть характеризувати роботу даного елемента критичної інформаційної інфраструктури. Нейронна мережа, яка розраховує якісну оцінку вибору включення даних елементів, на вхід буде приймати числові значення параметрів проектування, а на виході видавати якісну оцінку, відповідну одному

з лінгвістичних значень. Таким чином, така нейронна мережа буде виконувати роль класифікатора - розбивати простір параметрів проектування на зони, відповідні якісним оцінкам вибору розглянутих елементів. Дані якісні оцінки архітектор може потім використати для прийняття рішень щодо включення окремих елементів проекту до основного проектного рішення, а також для віднесення елементів з менш якісною оцінкою до альтернативних проектних рішень.

Як приклад елементу, якісну оцінку вибору включення якого необхідно розрахувати, візьмемо усереднений сервер застосувань. Взято шість серверів зі схожими технічними характеристиками, які виконують схожі завдання.

Визначено наступні чотири параметри, що впливають на якість його функціонування та на інші елементу проекту [18, 31, 38]:

- p_1 – ступінь завантаженості вінчестеру серверу, що розглядається. Його значення визначається як відношення кількості звернень до вінчестеру за секунду до максимально можливої кількості звернень за секунду;
- p_2 – ступінь завантаженості процесора сервера застосувань. Дана величина містить значення в діапазоні від 0 до 1;
- p_3 – завантаженість мережі сервера застосувань. Дана величина визначається як відношення теперішньої до максимально можливої швидкості передачі даних мережею;
- p_4 – зайнята оперативна пам'ять сервера. Дане значення визначається як відношення об'єму використаної до максимальної доступної оперативної пам'яті на сервері;
- q_1 – якість проектного рішення, яке використовує даний сервер застосувань. Дана величина містить значення в діапазоні від 0 до 1.

Таким чином, як показано на прикладі, формуються значення векторів $\{P', Q'\}$ для кожного елементу проектного рішення, які потрібно оцінити. Сформовані значення векторів використовуються як вхідні значення для алгоритму структурної оптимізації нейронних мереж. В результаті отримаємо класифікатор з оптимальною структурою нейронної мережі, який можна

використовувати для оцінювання елементів проектних рішень з схожими параметрами. Розраховані оцінки зберігаються в об'єкті QNEA відповідних елементів AR.

4.2 Методи та моделі управління ресурсами критичної інформаційної інфраструктури

Задача управління ресурсами є однією з основних задач функціонування критичної інформаційної інфраструктури. Неправильне управління ресурсами може звести до нуля всі переваги найкращого проектного рішення щодо архітектури критичної інформаційної інфраструктури. Для цієї задачі можна побудувати багато різних моделей з різними вихідними умовами та обмеженнями. Розглянемо декілька з них.

4.2.1 Модель управління ресурсами критичної інформаційної інфраструктури на базі нечіткої логіки

Нехай є декілька фізичних серверів $S_i, i=1...n$, на яких під управлінням гіпервізорів працюють віртуальні машини (ВМ) $V_i, i=1...m$.

Введемо необхідні для формування моделі розподілу ресурсів критичної інформаційної інфраструктури позначення [11]:

$Z_1, ..., Z_n$ – комплекс бізнес-процесів, підтримка яких забезпечує ефективне функціонування об'єкта управління;

$S_1, ..., S_m$ – комплекс універсальних сервісів, підтримка яких забезпечує ефективне функціонування об'єкта управління;

$w_1^z, ..., w_n^z$ – коефіцієнти критичності бізнес-процесів $Z_1, ..., Z_n$ відповідно за нечіткою шкалою:

- критичний - $\alpha_1 \leq w_i^z \leq 1$;
- дуже важливий - $\alpha_2 \leq w_i^z < \alpha_1$;

– важливий - $\alpha_3 \leq w_i^z < \alpha_2$;

– не важливий - $w_i^z < \alpha_3$.

$w_i^s, \dots, w_m^s$ – коефіцієнти критичності універсальних сервісів $S_1, \dots, S_m$

відповідно за нечіткою шкалою:

– критичний - $\gamma_1 \leq w_i^s \leq 1$;

– дуже важливий - $\gamma_2 \leq w_i^s < \gamma_1$;

– важливий - $\gamma_3 \leq w_i^s < \gamma_2$;

– не важливий - $w_i^s < \gamma_3$.

$R_1, \dots, R_m$ – ресурси сервера S_i критичної інформаційної інфраструктури у нодах, що необхідні для підтримки бізнес-процесів;

$T_1, \dots, T_m$ – надійність ресурсів сервера S_i критичної інформаційної інфраструктури;

$p_0 = \|\tilde{p}_{0ij}^k\|$ – матриця потреб бізнес-процесів у ресурсах критичної інформаційної інфраструктури на віртуальній машині V_k , яка гарантовано задовольняється, де $\tilde{p}_{0ij}^k$ дорівнює кількості потрібного для бізнес-процесу Z_i ресурсу R_j у вигляді триангулярного нечіткого числа $\tilde{p}_{0ij}^k = (p_{0ij}^{kz}, p_{0ij}^{kc}, p_{0ij}^{kk})$ чи 0, якщо ресурс не потрібен, де p_{0ij}^{kz} – найгірший (максимальний) варіант потреби у ресурсі, p_{0ij}^{kc} – середня потреба у ресурсі, p_{0ij}^{kk} – найкращий (мінімальний) варіант потреби у ресурсі.

$s_0 = \|\tilde{s}_{0ij}^k\|$ – матриця потреб універсальних сервісів у ресурсах критичної інформаційної інфраструктури на віртуальній машині V_k , яка гарантовано задовольняється, де $\tilde{s}_{0ij}^k$ дорівнює кількості потрібного для універсального сервісу S_i ресурсу R_j у вигляді триангулярного нечіткого числа $\tilde{s}_{0ij}^k = (s_{0ij}^{kz}, s_{0ij}^{kc}, s_{0ij}^{kk})$ чи 0, якщо ресурс не потрібен, де s_{0ij}^{kz} – найгірший (максимальний) варіант

потреби у ресурсі, s_{0ij}^{kc} - середня потреба у ресурсі, s_{0ij}^{kk} - найкращий (мінімальний) варіант потреби у ресурсі.

$p = \|\tilde{p}_{ij}^l\|$ – матриця потреб бізнес-процесів у ресурсах критичної інформаційної інфраструктури на віртуальній машині $V_l (l \neq k)$, які додатково бажано зарезервувати, де $\tilde{p}_{ij}^l$ дорівнює кількості потрібного для бізнес-процесу Z_i ресурсу R_j у вигляді триангулярного нечіткого числа $\tilde{p}_{ij}^l = (p_{ij}^{l\epsilon}, p_{ij}^{lc}, p_{ij}^{lk})$ чи 0, якщо ресурс не потрібен, де $p_{ij}^{l\epsilon}$ - найгірший (максимальний) варіант потреби у ресурсі, p_{ij}^{lc} - середня потреба у ресурсі, p_{ij}^{lk} - найкращий (мінімальний) варіант потреби у ресурсі.

$s = \|\tilde{s}_{ij}^l\|$ – матриця потреб універсальних сервісів у ресурсах критичної інформаційної інфраструктури на віртуальній машині $V_l (l \neq k)$, які додатково бажано зарезервувати, де $\tilde{s}_{ij}^l$ дорівнює кількості потрібного для універсального сервісу S_i ресурсу R_j у вигляді триангулярного нечіткого числа $\tilde{s}_{ij}^l = (s_{ij}^{l\epsilon}, s_{ij}^{lc}, s_{ij}^{lk})$ чи 0, якщо ресурс не потрібен, де $s_{ij}^{l\epsilon}$ - найгірший (максимальний) варіант потреби у ресурсі, s_{ij}^{lc} - середня потреба у ресурсі, s_{ij}^{lk} - найкращий (мінімальний) варіант потреби у ресурсі.

$t = \|\tilde{t}_j^i\|$ – матриця надійності ресурсів критичної інформаційної інфраструктури на віртуальній машині V_i , де $\tilde{t}_j$ дорівнює надійності ресурсу R_j у вигляді триангулярного нечіткого числа $\tilde{t}_j^i = (t_j^{i\epsilon}, t_j^{ic}, t_j^{ik})$ чи 0, якщо надійність ресурсу не важлива, де $t_j^{i\epsilon}$ - найгірший (мінімальний) варіант надійності ресурсу, t_j^{ic} - середня надійність ресурсу, t_j^{ik} - найкращий (максимальний) варіант надійності ресурсу.

Середня потреба у ресурсі та надійність визначається за формулами (4.2):

$$\begin{aligned}
p_{0ij}^{kc} &= \frac{p_{0ij}^{kz} + \beta p_{0ij}^{kne} + p_{0ij}^{kk}}{2 + \beta_0}, p_{ij}^{lc} = \frac{p_{ij}^{lz} + \beta p_{ij}^{lne} + p_{ij}^{lk}}{2 + \beta}, \\
s_{0ij}^{lc} &= \frac{s_{0ij}^{lz} + \chi s_{0ij}^{lne} + s_{0ij}^{lk}}{2 + \chi_0}, s_{ij}^{lc} = \frac{s_{ij}^{lz} + \chi s_{ij}^{lne} + s_{ij}^{lk}}{2 + \chi}, \\
t_j^{ic} &= \frac{t_j^{iz} + \tau t_j^{ine} + t_j^{ik}}{2 + \tau}
\end{aligned} \tag{4.2}$$

де $p_{0ij}^{kne}, s_{0ij}^{kne}, p_{ij}^{lne}, s_{ij}^{lne}$ - найімовірніші варіанти потреб у ресурсі для бізнес-процесу та універсального сервісу,

t_j^{ine} - найімовірніше значення надійності ресурсу, $\beta, \chi, \beta_0, \chi_0, \tau$ - зважені параметри усереднення, які визначаються експериментально.

$\tilde{r}_1^k, \dots, \tilde{r}_m^k$ - триангулярні нечіткі числа вигляду $(r_j^{kz}, r_j^{kc}, r_j^{kk})$, що визначають кількість ресурсів $R_1, \dots, R_m$ відповідно на віртуальній машині V_k .

a_{ij} - булева змінна, яка визначає, чи встановлена ВМ V_j на сервері S_i .

Очевидно, що для вказаної моделі розподілу ресурсів має виконуватись вимога (4.3):

$$p_{ij}^l + p_{oij}^k \geq p_{oij}^k, \tag{4.3}$$

У випадку, коли немає можливості для виділення додаткових ресурсів, нерівність (4.3) перетворюється у рівність (4.4):

$$p_{ij}^l + p_{oij}^k = p_{oij}^k, \tag{4.4}$$

Критичний процес або сервіс завжди обслуговується. Інші процеси або сервіси обслуговуються згідно коефіцієнтів критичності.

Введемо коефіцієнти при триангулярних нечітких змінних наступним чином:

$$x_i = \begin{cases} 1, \text{ якщо бізнес-процес } Z_i \text{ є критичним} \\ \quad \text{і обслуговується в першу чергу} \\ \alpha_2, \text{ процес є дуже важливим і обслуговується} \\ \alpha_3, \text{ процес є важливим і обслуговується} \\ 0, \text{ процес не важливий і не обслуговується} \end{cases}$$

$$y_i = \begin{cases} 1, \text{ якщо універсальний процес } S_i \text{ є критичним} \\ \quad \text{і обслуговується в першу чергу} \\ \gamma_2, \text{ сервіс є дуже важливим і обслуговується} \\ \quad \gamma_3, \text{ сервіс є важливим і обслуговується} \\ 0, \text{ сервіс не важливий і не обслуговується} \end{cases}$$

Оскільки кожна ВМ розташована тільки на одному сервері, то має виконуватись наступна умова (4.5):

$$\sum_i^n a_{ij} = 1, j = 1 \dots m, \quad (4.5)$$

Умова критичності сервісів і процесів накладає наступне обмеження (4.6):

$$\sum_{i=1}^n \sum_{j=1}^m a_{ij} p_{0ij}^{kc} \leq r_j^{kz}, k = 1 \dots n, \quad (4.6)$$

Таким чином, управління розподілом ресурсами критичної інформаційної інфраструктури зводиться до пошуку максимуму наступної цільової функції (4.7):

$$\begin{aligned} \tilde{U} &= \sum_i^n \tilde{x}_i \tilde{w}_i^z + \sum_j^m \tilde{y}_j \tilde{w}_j^s \rightarrow \max, \\ T &= \sum_j \tilde{t}_j^i r_j^k \rightarrow \max, \end{aligned} \quad (4.7)$$

за наявності обмежень (4.8) і (4.9):

$$\sum_{i=1}^n \tilde{x}_i \cdot (\tilde{p}_{0ij}^k + \tilde{p}_{ij}^l) + \sum_{i=1}^m \tilde{y}_i \cdot (\tilde{s}_{0ij}^k + \tilde{s}_{ij}^l) \leq \tilde{r}_j^k, , \quad (4.8)$$

$$N_{kp} \rightarrow \max, \quad (4.9)$$

для $j = 1 \dots m$ и $k, l = 1 \dots n$, та врахуванням умов (4.5) і (4.6).

Умова (4.9) визначає той факт, що кількість критичних процесів та універсальних сервісів, які потрібно обслужити є максимальною, тобто всі критичні процеси та сервіси повинні бути забезпечені ресурсами. В протилежному випадку управління розподілом ресурсів є неможливим і потрібно вводити додаткові ресурси.

Коефіцієнти $\tilde{w}_i^z, \tilde{w}_k^s$ цільової функції становлять собою нечіткі числа - $\tilde{w}_i^z \in [w_i^{zL}; w_i^{zR}]$, $\tilde{w}_k^s \in [w_k^{sL}; w_k^{sR}]$, L і R - ліві та праві границі носія нечіткого числа.

Задача (4.5) – (4.9) становить собою задачу нечіткого лінійного програмування, яку можна розв'язати за допомогою методів, які описані в [186].

Приклад використання запропонованої моделі. Розглянемо розподілену та багаторівневу хмару. Зазвичай, така хмара має сотні серверів, розташованих на різних стійках, в різних географічних точках. З'єднання між двома машинами з різних стійок може проходити через один або кілька комутаторів. Багаторівневий розподіл представляє дуже складне завдання надійного, масштабованого, доступного поширення даних. Для такої хмари потрібно розробити політику розташування реплік, яка повинна задовольнити наступним властивостям:

- максимізація надійності;
- максимізація забезпеченості.

Репліки повинні бути розташовані не тільки на різних дисках або різних машинах, але і в різних стійках. Це гарантує, що процес або сервіс буде доступним, навіть якщо ціла стійка пошкоджена або відключена від мережі. При такому розташуванні читання займає час, який приблизно дорівнює пропускну здатності мережі, хоча потік даних при записі повинен пройти через різні стійки.

Одночасно, при створенні нового критичного процесу або сервісу, визначається, де розмістити репліку. При цьому потрібно враховувати наступне:

1. Нова репліка критичного процесу або універсального сервісу розміщується на віртуальну машину сервера з найменшою середньою завантаженістю дисків. У такий спосіб вирівнюється завантаженість дисків на різних серверах та досягається найбільша надійність операції.

2. Число нових створюваних критичних процесів або універсальних сервісів на кожній віртуальній машині серверів є обмеженим. Незважаючи на те, що створення нового процесу є швидкою операцією, вона передбачає подальший запис даних на цю віртуальну машину, що вже є важкою операцією, і яка може привести до розбалансування обсягу трафіку даних на різні частини системи.

3. Як сказано вище, потрібно розподілити віртуальні машини між серверами в різних стійках. Це також дозволяє досягти найбільшої надійності операції.

4. Як тільки число реплік падає нижче встановлюваної користувачем величини, потрібно знову виконати репліку критичного процесу або сервісу. Ця ситуація може статися з декількох причин:

- віртуальна машина стала недоступною;
- сервер став недоступним;
- один з дисків вийшов з ладу;
- збільшене число реплік.

Кожному критичному процесу або універсальному сервісу, для якого потрібно зробити репліку, встановлюється відповідний пріоритет, який теж залежить від декількох факторів. По-перше, пріоритет вище у того критичного процесу або універсального сервісу, який має найменше число реплік. По-друге, щоб збільшити надійність виконання застосувань, збільшується пріоритет у процесів або у сервісів, які блокують прогрес у роботі клієнта. В першу чергу, обслуговуються критичні процеси.

5. Вибирається процес або сервіс з найбільшим пріоритетом і копіюється з однієї з реплік серверу, який є найбільш забезпеченим. Нова репліка розташовується, виходячи з тих же причин, що і при створенні.

6. Створення реплік постійно балансується. Залежно від розподілу реплік в системі, репліки переміщуються для вирівнювання завантаженості дисків і балансування навантаження. Також потрібно постійно вирішувати, яку з реплік має сенс видалити в даний момент. Як правило, видаляється репліка, яка знаходиться на віртуальній машині серверу з найменшим вільним місцем на жорстких дисках і яка належить найбільш забезпеченому і надійному критичному процесу.

Розглянута модель управління ресурсами (відповідно, ресурсами є кількість необхідних серверів, віртуальних машин та жорстких дисків, кількість вільних ресурсів для створення нових процесів та сервісів, реплік для розміщення кожного з наших критичних процесів та універсальних сервісів з метою їх надійного функціонування) в рамках існуючих обмежень (наявність серверів та місця для розміщення реплік на окремих серверах та віртуальних

машинах, кількість наявних ресурсів для створення нових процесів та сервісів) дозволяє розподілити максимальну кількість критичних бізнес-процесів та універсальних сервісів таким чином, щоб не порушувались принципи функціонування розглянутої хмари, гарантувалась максимальна забезпеченість та надійність функціонування такої критичної інформаційної інфраструктури.

4.2.2 Модель розподілу ресурсів критичної інформаційної інфраструктури з чіткими параметрами на основі генетичного алгоритму

Провайдери мережевих інфраструктур намагаються автоматизувати менеджмент віртуальних машин, враховуючи результуючу якість надання послуг. Така проблема може бути виражена у вигляді задачі оптимізації з обмеженнями. Функція, що оптимізується, вибирається таким чином, щоб досягти повного задовільнення вимог до якості сервісу, за допомогою мінімальних витрат на обслуговування. Значення функції являє собою численне представлення ступені задовільнення вимог щодо якості обслуговування [22]. Подібні алгоритми розміщення віртуальних машин складаються з двох модулів:

- модулю локальних рішень: кожен сервіс віртуалізованої ІТ-інфраструктури асоціюється з функцією, значення якої дорівнює ступені задовільнення якості надання ресурсів цьому сервісу.
- модулю глобальних рішень: за допомогою локальних рішень усіх сервісів у віртуалізованій ІТ-інфраструктурі виконується пошук оптимального значення глобальної функції.

При локальній оптимізації, необхідно врахувати декілька обмежень:

- кількість віртуальних машин кожного класу повинна відповідати вимогам фізичної машини;
- кожен сервіс ІТ-інфраструктури встановлює верхню межу на кількість віртуальних машин кожного класу та загальну кількість віртуальних машин.

Під час глобальної оптимізації, необхідний об'єм кожного ресурсу

виділених віртуальних машин не повинна перевищувати максимальний об'єм ресурсу фізичної машини. Основною ціллю є зменшення кількості активних фізичних машин та повне забезпечення ресурсами всіх критичних процесів.

Оскільки задача є NP-складною [201], застосування повного перебору можливе лише для невеликих розмірностей.

Для інших випадків, відома велика кількість евристичних алгоритмів [202, 203], що демонструють хороші результати:

- Best Fit (Найліпший підходящий);
- First Fit (Перший підходящий);
- Best Fit Decreasing (Найліпший підходящий за спаданням);
- First Fit Decreasing (Перший підходящий за спаданням);
- Heaviest First (Перший найбільший);
- Worst Fit (Найгірший підходящий).

Залежно від мети розміщення, алгоритми розміщення можуть бути поділені на 2 категорії:

- алгоритми оптимізації споживання енергії: пошук розміщення що дозволить системі споживати найменшу кількість енергії;
- алгоритми оптимізації якості обслуговування (QoS - Quality of Service): пошук розміщення що дозволить системі задовільнити вимоги користувачів щодо якості обслуговування.

Залежно від необхідності виконувати міграції (переміщення віртуальної машини від однієї фізичної машини до іншої), алгоритми розміщення поділяються на категорії:

- статичне розміщення: алгоритм розміщення не враховує стани віртуальних та фізичних машин
- динамічне розміщення: алгоритм розміщення виконує пошук оптимального рішення, враховуючи поточне розміщення віртуальних машин.

В свою чергу, динамічні алгоритми розміщення поділяються на 2 типа:

– проактивне: розміщення та міграції віртуальних машин виконуються до моменту, коли система набуде певного стану або до виконання певного критерію

– реактивне: розміщення та міграції віртуальних машин виконуються у відповідь на виконання певного критерію або на досягнення певного небажаного стану. Наприклад, зміна розміщення або виконання міграцій можуть бути необхідні під час технічного обслуговування фізичних машин, або після реєстрування погіршення якості надання послуг.

Для впровадження моделі розподілу ресурсів необхідно розглянути декілька ознак, що можуть вплинути на тип самої моделі. Моделі можуть відрізнятися в залежності від методів ведення бізнесу (для власних бізнес-процесів, чи для зовнішніх клієнтів). Також на вибір моделей впливає тип архітектури критичної інформаційної інфраструктури. Залежно від рівня абстракції ресурсів варіюється складність математичної моделі задачі, що розв'язується. Задачі великої розмірності розв'язуються у два етапи: на першому етапі здійснюється розподіл абстрактних ресурсів кожного типу без прив'язки до їх конкретного місцезнаходження з уточненням отриманих результатів на другому етапі.

Дуже істотно впливає на вибір моделей ознака забезпеченості ресурсами, тобто, чи дозволяється часткова підтримка сервісів, чи вони мають підтримуватися у повному обсязі, або не підтримуватися взагалі. У найпростішому випадку дворівневої клієнт-серверної архітектури модель розподілу обмежених ресурсів може подаватися у вигляді декількох бізнес-процесів, що безпосередньо використовують частину одного чи декількох незалежних один від одного ресурсів (рис. 4.13).

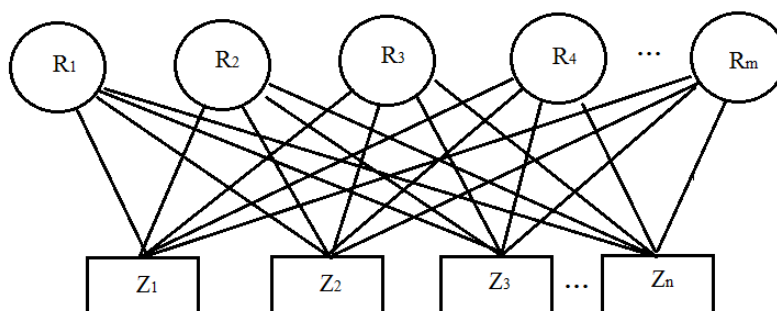


Рис. 4.13. Модель розподілу обмежених ресурсів ІТ-інфраструктури

Введемо необхідні для побудови моделей наступні позначення:

- $Z_1...Z_n$ – бізнес-процеси, підтримку яких забезпечує функціонування КІІ;
- $W=(w_1,...,w_n)$ – коефіцієнти важливості бізнес процесів $Z_1...Z_n$ відповідно;
- $R_1...R_m$ – інтегровані ресурси КІІ, необхідні для підтримки функціонування бізнес-процесів;
- $P=\|p_{ij}\|$ – матриця потреб бізнес-процесів у ресурсах ІТС, де p_{ij} відповідає наведеній кількості потрібного для процесу Z_i ресурсу R_j чи 0, якщо ресурс не потрібен;
- $D=\|d_{ij}\|$ – матриця наявності потреб бізнес-процесів у ресурсах КІІ, де:

$$\begin{cases} d_{ij} = 1, \text{ якщо } p_{ij} > 0 \\ d_{ij} = 0, \text{ якщо } p_{ij} = 0 \end{cases} \quad (4.10)$$

- $R=(r_1,...,r_m)$ – вектор встановлених обмежень на ресурси.

Розглянемо дискретний випадок, коли бізнес-процес або підтримується в повному обсязі, або повністю блокується. Введемо вектор $X=(x_1,...,x_n)$, де:

$$x_i = \begin{cases} 1, \text{ якщо процес } Z_i \text{ обслуговується} \\ 0, \text{ в протилежному випадку} \end{cases} \quad (4.11)$$

Тоді критерій оптимального розподілу ресурсів КІІ можна подати у вигляді:

$$\max \sum_{i=1}^n x_i w_i \quad (4.12)$$

Мають також бути використані обмеження по ресурсам (4.13):

$$\sum_{i=1}^n x_i p_{ij} \leq r_j, j = 1, \dots, m \quad (4.13)$$

Варто зазначити, що наведена вище модель, а також інші моделі, які розглядаються, належать до лінійних або нелінійних, неперервних, булевих та змішаних задач математичного програмування, які до того ж мають стохастичні аналоги. Для детермінованих задач можна використовувати евристичні алгоритми, методи м'яких обчислень та точні методи, використання яких обмежено з огляду на розміри задач. Так, для задач лінійного програмування можна використовувати відомі методи. Задачі булевого програмування можна розв'язувати методами часткового перебору. Точні методи дають змогу знайти найкращий розв'язок, але їх використання можливе лише для задач обмеженої розмірності, оскільки час пошуку рішень суттєво зростає зі збільшенням складності задачі. На відміну від них, евристичні методи та методи штучного інтелекту, насамперед генетичні алгоритми (ГА), дають змогу знайти задовільний розв'язок за доволі короткий час, навіть для задач великої розмірності, до того ж їхню ефективність можна поліпшити за рахунок врахування особливості задач [29].

Загальна схема алгоритму подана на рис. 4.14.



Рис. 4.14. Схема роботи генетичного алгоритму

Використання агентів дає змогу здійснювати управління ресурсами і

навантаженням на двох рівнях. Перший рівень – управління окремими віртуальними серверами. Другий рівень – управління окремими застосуваннями, що встановлені на віртуальних серверах.

Модуль прогнозування призначений для аналізу трендів з метою подальшої реалізації проактивного управління, у цьому випадку – перерозподілу ресурсів між застосуваннями до того, як виникла їх нестача для застосувань, задіяних у підтриманні бізнес-процесів високої важливості. Модуль планування призначений для довгострокового управління ресурсами ІТ-інфраструктури у разі її розвитку чи істотних змін у її функціонуванні. Диспетчер запитів слугує для обмеження клієнтського трафіку заданого типу з метою вивільнення ресурсів, споживання яких залежить від навантаження.

Модель управління віртуальними машинами при серверній віртуалізації. Нехай є декілька фізичних серверів $S_i, i=1, \dots, n$, на яких під управлінням гіпервізорів функціонують віртуальні машини (ВМ) $V_j, j=1, \dots, m$ (рис. 4.15). Кожна з ВМ в залежності від потоку клієнтських запитів використовує певну кількість ресурсів типу $R_k, k=1, \dots, l$ (пам'ять, процесорний час, дисковий простір, пропускна спроможність підсистеми вводу-виводу, зовнішніх каналів зв'язку тощо).

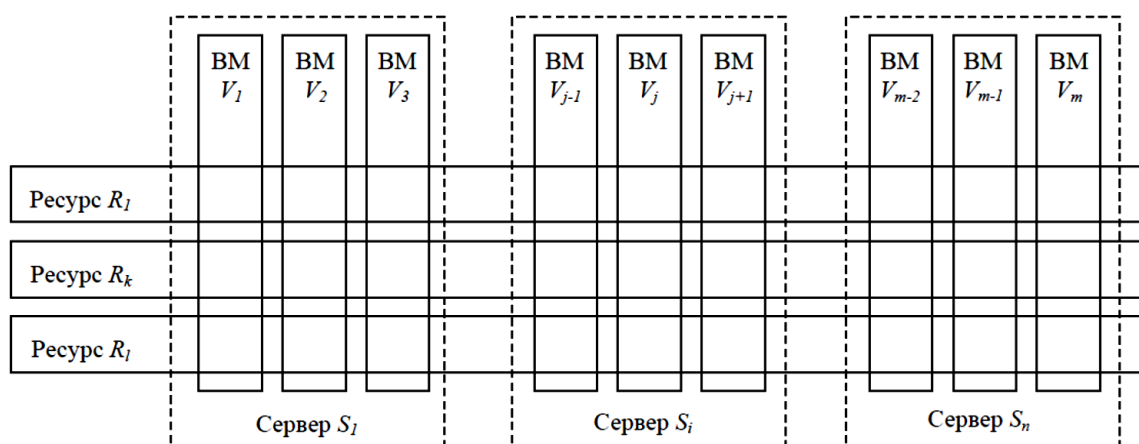


Рис. 4.15. Модель розміщення віртуальних машин на фізичних серверах [111]

Розглянемо процес розподілу ресурсів серверів між віртуальними

машинами. Для цього введемо необхідні для побудови моделей позначення:

- r_{kj} – кількість ресурсу типу R_k , що встановлено на сервері S_i ;
- p_{kj} – потреби ВМ V_j у ресурсах типу R_k для забезпечення заданого рівня SLA;
- x_{ij} – булева змінна, яка визначає, чи встановлена ВМ V_j на сервері S_i .

Оскільки кожна ВМ одночасно розташована не більше ніж на одному сервері, має виконуватись умова описана у формулі (4.14):

$$\sum_{i=1}^n x_{ij} \leq 1, j = 1, \dots, m \quad (4.14)$$

До того ж, для нормального функціонування ВМ повинні бути забезпечені достатнім обсягом ресурсів серверів, на яких вони розташовані (формула 5.15).

$$\sum_{j=1}^m x_{ij} p_{kj} \leq r_{ki}, k = 1, \dots, l; i = 1, \dots, n \quad (4.15)$$

Розглянемо задачу розташування ВМ для випадків нестачі та надлишку ресурсів.

У разі, якщо внаслідок збільшення клієнтських запитів потреба окремих ВМ у ресурсах певного типу збільшилась настільки, що стає неможливим забезпечити усі ВМ необхідною кількістю ресурсів, природним виходом стає задача підтримки найбільш важливих бізнес-процесів шляхом забезпечення ресурсами тих ВМ, на яких працюють пов'язані з ними сервіси, за рахунок менш важливих.

Позначимо через $w_j, j = 1, \dots, m$, важливість застосувань, встановлених на ВМ V_j . Тоді задачу можна сформулювати наступним чином (4.16):

$$\sum_{j=1}^m \sum_{i=1}^n x_{ij} w_j \rightarrow \max \quad (4.16)$$

при обмеженнях (4.14), (4.15).

Ця задача становить собою лінійну задачу булевого програмування. Для її вирішення скористаємося методом гілок та меж. Але спочатку покажемо, як

можна, враховуючи специфіку задачі, зменшити початкову кількість можливих комбінацій, яка складає $2mn$.

По-перше, згідно з обмеженням (4.14), кожна ВМ може бути розташована не більше, ніж на одному сервері, тобто замість перебору $2n$ комбінацій для кожної ВМ V_j , можна обмежитись розглядом лише $n+1$ варіантів: $y_j = 0, \dots, n$, де y_j – номер сервера, на якому встановлена ВМ V_j ($y_j = 0$) у випадку, якщо внаслідок нестачі ресурсів ВМ V_j не встановлена на жодному сервері або ресурси для неї виділяються по залишковому принципу).

По-друге, враховуючи те, що вимоги ВМ у ресурсах p_{ij} є невід’ємними, якщо на якомусь з етапів побудови дерева варіантів розміщення ВМ по серверам одне з обмежень (4.15) перестає виконуватись, продовжувати побудову цієї гілки немає сенсу, оскільки для усіх її вузлів це обмеження також не буде виконуватись.

По-третє, слід врахувати те, що дуже часто при побудові серверних ферм використовують сервери з однаковою конфігурацією. Якщо, наприклад, є два ідентичних сервера, кількість можливих варіантів розміщення ВМ можна скоротити в 2 рази, оскільки перенесення усіх ВМ з першого сервера на другий, а з другого на перший ніяк не вплине на об’єм ресурсів, які використовуються. Тобто перестановка рядків у матриці $X = \|x_{ij}\|$, які відповідають однаковим стовпцям у матриці $X = \|x_{ij}\|$, ніяк не впливає, ані на виконання обмежень (4.14), (4.15), ані на значення критерію (4.16). Іншими словами, для серверів з однаковою конфігурацією важливе не абсолютне розміщення ВМ на цих серверах, а їх розташування одна відносно іншої. При збільшенні кількості серверів з однаковою конфігурацією кількість «зайвих» комбінацій значно зростає від $(n-1)$ для випадку розміщення усіх ВМ на одному сервері до $(n!-1)$ при розміщенні ВМ на різних серверах [22].

Так, наприклад, для випадку трьох ВМ і трьох ідентичних серверів з можливих 27 комбінацій розташування ВМ лише 5 є унікальними, а решта – 22

комбінації – «дзеркальними» до них. Тобто, навіть для таких незначних значень m і n початкову множину варіантів можна скоротити більше, ніж у 5 разів.

Для наочності буквами («А», «В», «С») позначені ВМ, цифрами (1, 2, 3) – номери серверів, комбінацією букви з цифрою – розміщення ВМ на сервері. Алгоритм перебору при побудові дерева розташування ВМ полягає у наступному: на черговому рівні чергової гілки кількість серверів, що розглядаються, для перебору приймається на 1 більше, ніж максимальна для попередніх рівнів цієї ж гілки, але не більше ніж n .

Так, на рівні «А» розглядається лише випадок розташування ВМ «А» на сервері №1, а на серверах №2 та №3 не розглядається, оскільки сервери №1-3 є повністю ідентичними і розташування ВМ «А» на них призведе до тих самих результатів. При розв'язанні задачі в реальних умовах, з метою зменшення кількості міграцій ВМ, при виборі довільного сервера обирається той, на якому ця ВМ вже встановлена. На рівні «В» розглядаються два варіанти «В1» – розміщення ВМ «В» на тому ж сервері, що і ВМ «А», та «В2» – розміщення ВМ «В» на іншому сервері, у цьому випадку №2. Варіант «В3» не розглядається, оскільки він нічим принципово не відрізняється від варіанту «В2» і призведе до тих самих результатів (аналогічно, якщо ВМ «В» уже встановлена на сервері №3, слід використовувати саме його, а не сервер №2).

Якщо у наявності є сервери з іншою конфігурацією, до схеми перебору додаються відповідні вузли на кожному рівні. Сюди також відносяться випадки, коли внаслідок нестачі ресурсів дозволяється низькопріоритетні ВМ не розміщувати на жодному сервері чи виділяти ресурси для них за залишковим принципом. Формально це описується шляхом розміщення таких ВМ на неіснуючому сервері № 0.

Крім визначення процедури розгалуження, метод гілок і меж передбачає визначення процедур оцінки верхньої та нижньої меж за допомогою наближених, але швидких алгоритмів.

При вирішенні задача максимізації для визначення оцінки зверху можна використати оптимістичний прогноз, який полягає в припущенні того, що на

черговому кроці алгоритму усі ВМ, не розподілені між серверами, можуть бути розподілені таким чином, що жодна з них не отримує відмови в обслуговуванні (не буде розміщена на неіснуючому сервері № 0) за умови виконання ресурсних обмежень (4.15).

Щоб одержати оцінку знизу, пропонується використати жадібний алгоритм, упорядкувавши ВМ за зменшенням вимог до того ресурсу, нестача якого стала причиною переходу від старого до нового плану розміщення ВМ.

Процедура відсікання гілок полягає у тому, що гілка, для якої оцінка верхньої межі менша, ніж оцінка нижньої межі хоча б однієї з інших гілок, розглядається як безперспективна та виключається з дерева рішень. Враховуючи те, що вимоги ВМ до ресурсів є невід’ємними, ця процедура може бути доповнена відсіканням тих гілок, для яких припиняють виконуватись обмеження (4.15).

У разі, якщо на черговому кроці буде знайдено рішення, при якому усі ВМ успішно розміщені, тобто жодна ВМ не розміщена на сервері № 0, пошук можна припинити, оскільки глобальний оптимум знайдений і критерій (4.16) набуває максимального значення. Але, за умови наявності часових можливостей, продовження пошуку може допомогти знайти рішення, що забезпечить меншу кількість міграцій ВМ при реалізації нового плану розташування, ніж уже знайдено рішення та його «дзеркальні» рішення.

Проблема розміщення віртуальних машин є багатокритеріальною задачею оптимізації. Наша мета полягає у тому, щоб знайти оптимальне розміщення, що являє собою відношення між віртуальними та фізичними машинами. Для формального опису моделі, введемо наступні поняття:

R - множина ресурсів;

P - множина фізичних машин;

V - множина віртуальних машин;

v_i - віртуальна машина i ;

v_i^r - необхідність віртуальної машини i у ресурсі r ;

P_{all} - множина задіяних (аллокованих) фізичних машин;

p_j - фізична машина j ;

p_j^r - наявність ресурсу r у фізичної машини j ;

V_i - множина віртуальних машин що призначені p_j .

Нашою метою є мінімізація кількості задіяних фізичних машин, тим саме мінімізуючи використання спожитої енергії:

$$P_{all} = \{p_j \in P \mid |V_j| > 0\} \quad (4.17)$$

Опишемо додаткові критерії оптимізації:

Усі віртуальні машини повинні бути розміщені:

$$V = \bigcup_{p_j \in P} V_j \quad (4.18)$$

Кожна віртуальна машина може бути розміщена лише на одній фізичній машині:

$$V_i \cap V_j = \emptyset, i \neq j \quad (4.19)$$

Для кожної фізичної машини, сумарне споживання кожного ресурсу її аллокованих віртуальних машин не повинне перевищувати об'єм доступного ресурсу фізичної машини:

$$p_j^r \geq \sum_{v_i \in V_j} v_i^r, r \in R \quad (4.20)$$

Кожне рішення у популяції характеризується хромосомою, що має структуру N -мірного вектору:

$$C = (p_1, p_2, \dots, p_N) \quad (4.21)$$

N - кількість віртуальних машин,

p_i - фізична машина, що призначена i -тій віртуальній машині.

У запропонованій структурі кожна хромосома містить N генів, кожен з яких визначає позицію кожної віртуальної машини. Така структура хромосоми дає можливість простим чином дотримуватись виконання обмежень (4.18) та (4.19).

Для досягнення оптимального розподілу усіх ресурсів, визначимо функцію пристосованості кожного індивіда популяції (4.22), (4.23) та (4.24):

$$f(i) = -(k+1) \sum_{r \in R} \sum_{p_j \in P_{all}} \frac{w_j^r p_j^r}{p_j^r} \quad (4.22)$$

$$u_j^r = \frac{\sum_{v_i \in V_j} v_i^r}{p_j^r} \quad (4.23)$$

$$w_j^r = 1 - u_j^r \quad (4.24)$$

де u_j^r - використання ресурсу r фізичною машиною j ,

w_j^r - марнування ресурсу r фізичною машиною j ,

k – кількість фізичних машин, для яких споживання ресурсів є надмірним.

Функція пристосованості, що має від'ємне значення, визначає наскільки ефективним є дане розміщення. Неефективне розміщення матиме дуже мале значення функції пристосованості, що дасть змогу вилучати такі розміщення із вибірки. Також, якщо сумарне споживання ресурсу однієї фізичної машини перевищує її запас, ми зменшуємо значення функції пристосованості індивіда у зовнішній сумі (4.22). Завдяки такому штрафу досягається компроміс між недостатнім та надмірним використанням ресурсів.

У той же час марнування ресурсів є більш сприятливим, ніж їх перевикористання. Таким чином, значення штрафу буде меншим завдяки внутрішній сумі. Таким чином, ми досягаємо мінімального марнування ресурсів в усій інфраструктурі.

Варто зауважити, що штрафи не додаються для фізичних машин, що знаходяться у стані простою (для яких не призначено жодної віртуальної машини), оскільки такі фізичні машини можуть бути вимкнені. Нашою метою є максимальна утилізація ресурсів датацентру, і для реалізації цього ми намагаємось збільшити кількість фізичних машин, які можна вимкнути. Чим більша кількість таких фізичних машин, тим більше значення функції пристосованості, що приводить до зменшеного загального використання енергії.

Під час операції вибірки відбувається попарний відбір рішень, над якими згодом виконується операція схрещення. Операція вибірки створює підпопуляцію з рішеннями із загальної популяції, вибрані випадковим чином. Розмір підпопуляції дорівнює 5% розміру загальної популяції. Це значення було підібрано експериментальною та контролює межу між різноманітністю популяції та якістю отриманих рішень.

Операція схрещення визначає, які гени успадкує рішення наступної популяції відносно своїх предків. Операція мутації випадковим чином вибирає ген хромосоми та присвоює йому випадково вибрану фізичні машини. Мутації мають велике значення для методів що базуються на генетичних алгоритмах, оскільки це дає змогу значно збільшити різноманітність популяції. У наших експериментах для збільшення різновиду у популяції, вірогідність успадкування гену від обох предків дорівнює 0.5. Ми не присвоюємо більшу вірогідність успадкування гену предку із більшим значенням функції пристосованості.

Запропонований генетичний алгоритм у більшій мірі схожий на традиційні генетичні алгоритми. На початку роботи алгоритму випадковим чином створюється початкова популяція рішень-індивідів. Далі, на кожній ітерації алгоритму ми обчислюємо значення функції пристосованості кожного індивіду, для кожного індивіда у популяції вибирається пара для генерації індивідів наступної популяції. Після цього ми застосовуємо операцію мутації. Окрім того, ми виконуємо пошук найкращого індивіду нової популяції та порівнюємо його із найкращим індивідом попередньої популяції. Якщо найкращий індивід нової популяції кращий ніж найкращий індивід попередньої популяції, ми зберігаємо новий індивід як останній найкращий. У наступній ітерації алгоритму ми продовжуємо роботу із індивідами нової популяції, повністю відкидаючи попередню популяцію, оскільки індивіди нової популяції містять кращі рішення ніж індивіди попередньої популяції.

Розподіл ресурсів із врахуванням вимог високої доступності критичної інформаційної інфраструктури. До об'єктів критичної інформаційної інфраструктури пред'являють особливі вимоги щодо забезпечення високої

доступності. Введемо наступні поняття:

- відмовостійкість – здатність системи до подальшої роботи після відмови одного із її елементів;
- неперервна доступність – здатність системи до безперервного обслуговування, незалежно від часу відмови вузлів системи;
- високодоступність – здатність системи до подальшої роботи після відмови одного із вузлів, з можливими перервами у роботі.

Як правило, при реалізації неперервної доступності виникає багато труднощів, подолання яких вимагає значних фінансових витрат. Водночас, існують ситуації, коли необхідно забезпечити відмовостійкість системи без жорстких вимог до неперервності доступу.

Розміщення елементів будь-якого високодоступного кластеру завжди повинне виконувати наступну вимогу: основні та резервні елементи критичної інформаційної інфраструктури повинні знаходитись на різних фізичних серверах. Алгоритми, що наведені у попередніх розділах, можуть бути просто модифіковані для уведення додаткових вимог щодо розміщення віртуальних машин шляхом впровадження додаткових “ресурсів”.

У (4.23) ресурс є кумулятивним об’єктом: сумарне споживання ресурсу дорівнює сумі споживань цього ресурсу кожною віртуальною машиною. Для кожного елементу високодоступного кластеру, що повинен знаходитись на окремій фізичній машині, визначимо його “спільний” ресурс за (4.25):

$$u_j^c = \begin{cases} +\infty, & |V_j \cap V^c| > 1 \\ 1, & |V_j \cap V^c| \leq 1 \end{cases} \quad (4.25)$$

де V^c – множина віртуальних машин, що належать кластеру c .

Таким чином, при виявленні на одній фізичній машині більш ніж одного елементу кластеру, у фітнес-функцію вводиться додаткова штрафна величина. Для більш загального випадку, коли необхідно виконати розміщення кластеру, що використовує певну мінімальну кількість фізичних серверів, визначимо ресурс за (4.26) та (4.27):

$$u_j^c = \begin{cases} +\infty, & |V_j \cap V^c| > 1 \\ 1, & |V_j \cap V^c| \leq I \end{cases} \quad (4.26)$$

$$I = C - k + 1 \quad (4.27)$$

де C – кількість елементів високодоступного кластеру;

k – мінімальна кількість різних фізичних машин, що повинні бути задіяні.

Міграції та врахування поточного розміщення. Досі розглядалися методи, що розраховують оптимальне розміщення компонентів критичної інформаційної інфраструктури на етапі запуску інфраструктури. Проте під час експлуатації можуть виникати ситуації, що потребують корегувати поточне розміщення:

- збільшення споживання ресурсів компонентом інфраструктури;
- відмова одного або декількох серверів;
- необхідність планового оновлення обладнання або програмного забезпечення.

У системах з підвищеними вимогами до надійності повинні існувати засоби динамічного балансування навантаження. Для здійснення цієї вимоги необхідно модифікувати фітнес-функцію таким чином, щоб вона враховувала властивості як початкового розміщення, так і нового. Розроблена фітнес-функція представлена у (4.28) та складається із трьох компонентів. На відміну від (4.22), в якій компоненти марнування та перевикористання ресурсів поєднувались через множення безрозмірних величин, у (4.28) усі компоненти приведені до однієї величини – фінансових витрат, що понесе компанія-власник критичної інформаційної інфраструктури. Така реалізація функції оцінки є простішою, більш наочною та краще піддається аналізу:

$$F(A_0, A) = F_w(A) + F_0(A) + F_m(A_0, A) \quad (4.28)$$

де A_0 - стан початкового розміщення;

A - стан нового розміщення;

$F_w(A)$ - витрати марнування ресурсів інфраструктури за нового розміщення;

$F_0(A)$ – витрати, пов'язані із недостатнім забезпеченням сервісів

інфраструктури;

$F_m(A_0, A)$ - витрати на приведення інфраструктури із стану A_0 до стану A

.

Таким чином, мінімальне значення фітнес-функції буде являти собою оптимальне співвідношення між марнуванням ресурсів та забезпеченням сервісів.

Розглянемо кожен її компонент окремо. Для оцінки витрат, пов'язаних з марнуванням ресурсів інфраструктури (4.29), для кожного ресурсу необхідно додатково задати величину середніх витрат одиниці ресурсу, що несе власник інфраструктури. Для прикладу, для розрахунку таких витрат можна використати амортизаційну вартість обладнання, що підтримує ресурс у наявності, об'єм електроенергії, що споживається обладнанням або об'єм упущеної вигоди, що могла б бути отримана. Очевидно, що коли певний ресурс використовується в повному або надмірному обсязі, це значення дорівнює нулю.

$$F_w(A) = \sum_{p \in P} \sum_{r \in R} w_r^p \quad (4.29)$$

$$w_r^p = \begin{cases} 0, & \text{якщо } v^r \geq p^r \\ (p^r - v^r) \cdot C^r, & \text{інакше} \end{cases} \quad (4.30)$$

де w_r^p - витрати марнування ресурсу;

v^r - об'єм використаного ресурсу;

p^r - об'єм наявного ресурсу;

C^r - ціна витрат, пов'язаних з марнуванням однієї одиниці ресурсу.

Для гарантії забезпечення сервісів необхідним об'ємом ресурсів, компонент фітнес-функції (4.31) визначає штраф, що стягується при порушенні такої умови. Очевидно, що коли сервіс отримує необхідний об'єм ресурсів, це значення дорівнює нулю. Величина штрафу є нелінійною та залежить від об'єму фактично наданих ресурсів. Приклад зображений на рис. 4.16.

$$F_0(A) = \sum_{s \in S} \sum_{r \in R} Y_s^r \quad (4.31)$$

$$Y_s^r = \begin{cases} 0, & \text{якщо } v_{rec}^r \geq v_{req}^r \\ K_s \left(\frac{v_{req}^r}{v_{rec}^r} - 1 \right), & \text{інакше} \end{cases} \quad (4.32)$$

де v_{req}^r - об'єм ресурсу, що вимагає сервіс для своєї роботи;

v_{rec}^r - об'єм наданого ресурсу для використання сервісом;

K_s - штраф внаслідок падіння швидкодії сервісу, що зумовлена двократною нестачею ресурсу.

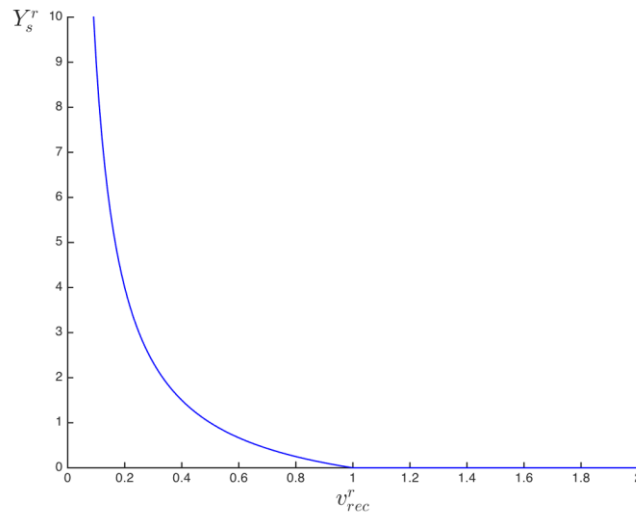


Рис. 4.16. Залежність величини штрафу від об'єму наданих ресурсів при

$$K_s = 1, v_{req}^r = 1$$

Останній компонент фітнес-функції (4.33) відповідає за мінімізацію витрат, пов'язаних із приведенням системи з одного стану до іншого: міграцією сервісів. Для урахування вартості здійснених міграцій необхідно визначити функцію вартості переміщення сервісу, що оцінює складність міграції сервісу з початкового стану у кінцевий. Ця функція повинна задовольняти вимогам (4.34), (4.35) та залежить від особливостей реалізації. Як правило, вартість переміщення тим більша, чим більша його складність. Для прикладу, вартість переміщення сервісу у інший ДЦ повинна бути більшою, ніж вартість переміщення у межах

одного ДЦ [22].

$$F_m(A_0, A) = \sum_{s \in S} F_m^*(s, A_0, A) \quad (4.33)$$

$$F_m^*(s, A_k, A_k) = 0 \quad (4.34)$$

$$F_m^*(s, A_k, A_m) \geq 0 \quad (4.35)$$

де A_0 - стан початкового розміщення;

A - стан нового розміщення;

F_m^* - функція вартості переміщення сервісу.

Результати моделювання представлені в розділі 5.

4.2.3 Модель розподілу ресурсів критичної інформаційної інфраструктури з чіткими параметрами на базі методу рою часток

Кожен індивід популяції, який називається «часткою», являє собою рішення у просторі пошуку. Популяція, що називається “роєм”, складається з множини потенційних розв’язків задачі оптимізації. Основна ідея МРЧ — взаємодія та комунікація часток між собою. МРЧ дозволяє знаходити близьке до оптимального рішення за прийнятний час.

Кожна частка описується наступною групою векторів [23]:

$$(\vec{x}_i, \vec{v}_i, \vec{p}_i) \quad (4.36)$$

$$\vec{x}_i = (x_{i1}, x_{i2}, \dots, x_{iD}) \quad (4.37)$$

$$\vec{v}_i = (v_{i1}, v_{i2}, \dots, v_{iD}) \quad (4.38)$$

$$\vec{p}_i = (p_{i1}, p_{i2}, \dots, p_{iD}) \quad (4.39)$$

Кожна частка має позицію (4.37), швидкість руху (4.38) та найкраще рішення, що було досягнуто часткою в просторі рішень (4.39). На початку роботи алгоритму позиції часток генеруються випадково. Розмірність векторів визначається розмірністю задачі оптимізації. Серед усіх часток найкраще рішення популяції має вигляд:

$$\vec{g}_i = (g_{i1}, g_{i2}, \dots, g_{iD}).$$

На кожній ітерації вектори позиції та швидкості кожної частки змінюються за наступними формулами:

$$v_{id}[t+1] = wv_{id}[t] + c_1r_1(p_{id} - x_{id}[t]) + c_2r_2(p_{gd} - x_{id}[t])$$

$$x_{id}[t+1] = x_{id}[t] + v_{id}[t+1]$$

де $i=1,2,\dots,m$ – кількість часток; p_i – позиція i -ї частки; p_g – найкраще рішення популяції; $d=1,2,\dots,D$ – розмірність часток; r_1, r_2 – випадкові числа; c_1 – індивідуальна швидкість навчання; c_2 – соціальна швидкість навчання; w – коефіцієнт інерції.

Параметр c_1 визначає власну здатність індивіда до пошуку рішення, тоді як параметр c_2 визначає соціальну комунікацію, тобто вплив соціального середовища на рух частки. Визначення оптимальних значень обох параметрів є предметом багатьох досліджень. За допомогою МРЧ виконується пошук оптимального рішення, оновлюючи популяцію часток у кожній ітерації. Критерієм зупинки може бути досягнення максимальної кількості ітерацій або бажаного мінімального значення помилки.

Недоліком МРЧ є тенденція часток до досягнення локального оптимуму. Рух рою зупиняється, коли кожна частка досягає свого локального оптимуму. Для вирішення цієї проблеми часто використовують метод Local Improvement Procedure [204], що дає змогу кожній частці продовжити рух після досягнення локального оптимуму. Зрештою, це можна вирішити підібравши значення параметру інерції: велике значення інерції сприяє пошуку глобального рішення, а мале значення – локальних. У роботі [205] зазначено, що МРЧ дає кращі результати при використанні не фіксованого значення, а такого, яке змінюється лінійно від одиниці до 0,4 з плином ітерацій. МРЧ має здатність швидко сходиться та якість отриманих рішень не сильно залежить від розміру популяції [206].

Для роботи МРЧ у динамічному середовищі часто використовують наступні методи:

1. Випадковий коефіцієнт інерції — найпростіший спосіб модифікації алгоритму для роботи в динамічному середовищі. У роботі [207] використано формулу $w = 0.5 + rnd / 2.0$ для зміни коефіцієнту інерції. Це дозволяє відслідковувати оптимум, що змінюється в динамічній системі.

2. Рерандомізація — метод описує ефект рерандомізації для знаходження оптимуму в динамічному середовищі. У роботах [206, 207] запропоновано два методи: «changed-gBest-value» та «fixed-gBest-value». Метод «changed-gBest-value» базується на припущенні, що оптимальне значення кожної частки змінюється тоді, коли змінюється оптимальне значення системи. При виявленні такої ситуації відбувається скидання пам'яті часток. Незважаючи на те, що метод швидкий, він потребує витрат на переобчислення функції пристосування. Метод «fixed-gBest-value» полягає у відслідковуванні найкращого глобального значення та найкращого глобального значення для фіксованої кількості ітерацій. Якщо протягом ітерацій ці значення не змінюються, може відбутись зміна оптимуму системи. В такій ситуації метод рерандомізує набір часток для відслідковування зміни оптимуму. Рерандомізацію також можна розглядати як заміну старих часток новими. У більшості випадків радять використовувати значення параметру, що дорівнює 1%.

3. Динамічний МРЧ з використанням опорних часток. Цей метод [208] базується на використанні фіксованої кількості стаціонарних опорних часток, які використовуються для виявлення змін стану динамічного середовища. При виявленні зміни середовища виконується очистка пам'яті, а саме кожна частка замінює значення своєї найкращої позиції значенням поточної позиції. Недоліком методу є те, що опорні частки можуть виявляти лише локальні зміни, тобто зміни в місцях, де вони розташовані. У такому випадку класична децентралізована модель обчислень МРЧ перетворюється у централізовану, що створює деякі труднощі при використанні у децентралізованому середовищі. У роботі [209] запропоновано метод Multi Ensemble Particle Swarm Optimization (мультиансамблевий МРЧ) для роботи у динамічному середовищі. Популяція

часток поділяється на дві групи. Частки першої групи виконують пошук глобального оптимуму у просторі рішень в поточному середовищі. Це гарантує хороше сходження алгоритму. Для підвищення ефективності пошуку часток першої групи використано гаусів локальний пошук. Натомість, частки другої групи рухаються на межі простору пошуку часток першої групи, таким чином виявляючи зміни в глобальному оптимумі.

4. Децентралізований МРЧ без опорних часток. У роботі [208] описано динамічний адаптивний МРЧ (ДАМРЧ) для моніторингу та реагування на зміни в середовищі. Цей метод розроблено для пошуку рухомого глобального оптимуму. Головною характеристикою алгоритму є можливість оновлення пам'яті часток без централізованого управління. У ДАМРЧ немає спеціальних часток, які відслідковують зміни, або додаткової функції пристосованості. Натомість, кожна частка порівнює значення функції пристосованості поточної позиції до значення її найкращої позиції. Якщо значення поточної позиції не поліпшується, частка почне змінювати значення пристосованості своєї найкращої позиції за таким законом:

$$P_i(t+1) = \begin{cases} P_i(t) * T, & f(X_i(t+1)) \leq P_i(t) * T \\ f(X_i(t+1)), & f(X_i(t+1)) > P_i(t) * T \end{cases}$$

де T – фактор випаровування у межах $[0, 1]$.

Найважливішим кроком проектування успішного МРЧ є вибір правильного подання рішення, яке точно описує відношення між частками МРЧ та задачею, що вирішується [210]. При розміщенні N віртуальних машин кожна частка буде представлена у вигляді N -мірного вектору. Вектори позиції та швидкості визначаємо так:

$$x_i^k = (x_{i1}^k, x_{i2}^k, \dots, x_{in}^k), \quad v_i^k = (v_{i1}^k, v_{i2}^k, \dots, v_{in}^k)$$

де x_i^k – позиція j -ї віртуальної машини i -ї частки у k -й ітерації; v_i^k – швидкість j -ї віртуальної машини i -ї частки у k -й ітерації.

Оригінальний МРЧ використовується для задач оптимізації, в яких елементами простору рішень є неперервні дійсні числа. Саме неперервна

природа алгоритму є складною перешкодою для використання алгоритму у комбінаторних задачах. Існує багато методів виправлення цього недоліку, одним із яких є метод найменшого значення позиції (НЗП) [211]. За допомогою цього методу, неперервні значення позиції перетворюються у таку послідовність розміщень:

$$s_i^k = (s_{i1}^k, s_{i2}^k, \dots, s_{in}^k) \quad (4.40)$$

$$r_i^k = (r_{i1}^k, r_{i2}^k, \dots, r_{in}^k) \quad (4.41)$$

$$r_{ij}^k = s_{ij}^k \bmod M \quad (4.42)$$

Остаточний вектор ідентифікаторів фізичних машин (ФМ) (4.41) вираховується за формулою (4.42).

Наприклад, нехай необхідно розмістити 10 віртуальних машин (ВМ) на чотирьох ФМ [23]. Приклад кодування вектору позиції i -ї частки у k -й ітерації наведений у табл. 5.1. Згідно правила НЗП, найменше значення позиції i -ї частки дорівнює $x_{i3}^k = -0.39$, отже ВМ під номером 3 призначається першій ФМ. Друге найменше значення позиції дорівнює $x_{i2}^k = -0.21$, отже ФМ під номером 2 розміщується на наступній ФМ.

Табл. 4.1. Приклад кодування розміщення 10 ВМ на чотирьох ФМ

Номер ВМ	x_{ij}^k	s_{ij}^k	r_{ij}^k
0	1,13	3	4
1	3,91	9	2
2	-0,21	1	2
3	-0,39	0	1
4	3,25	7	4
5	3,51	8	1
6	2,74	5	2
7	3,10	6	3
8	0,99	2	3
9	1,62	4	1

4.3 Метод верифікації даних в інформаційних ресурсах критичної інформаційної інфраструктури

Однією з основних характеристик при проектуванні критичних інформаційних інфраструктур є належна якість даних, що циркулює в її інформаційних ресурсах. Відсутність верифікованих даних є суттєвою проблемою, що також може призвести до порушення роботи КІІ.

Створювана технологія наскрізної ідентифікації передбачає трансформування комбінації схем полів з даними в хеш-коди за допомогою одностороннього алгоритму хешування. Декілька полів з даними можуть об'єднуватися в різні шаблони з метою полегшення пошуку в різних інформаційних масивах.

4.3.1 Первинне формування зведеного масиву

Первинне формування записів зведеного масиву даних відбувається з використанням N полів даних для ідентифікації об'єкту, включаючи M обов'язкових полів БД і L опціональних полів (див. табл. 4.2). Як правило, вони є унікальними для об'єкту і не змінюються протягом життєвого циклу. Кожне поле даних має свою апроксимовану ймовірність ідентифікування двох різних об'єктів. Це означає, що 2 різних об'єкти можуть бути випадковим чином ідентифіковані в межах вибірки, і які мають такі ж значення для цього поля даних.

Табл. 4.2. Поля даних

Тип	Назва	Значення
Обов'язковий	X_1	Обов'язкове поле 1
	...	...
	X_M	Обов'язкове поле M
Опціональний	Z_1	Опціональне поле 1
	...	...

Тип	Назва	Значення
	Z_T	Опціональне поле T

Кожне поле даних програмно нормалізується з метою їх приведення до уніфікованого формату (тільки великі літери і цифри, без пробілів, знаків пунктуації). Всі поля нормалізованої база даних хешуються за допомогою алгоритму SHA-512. Саме цей тип хешування обрано через його поширеність та наявність підтримки у сучасних СКБД. Саме за такою процедурою пропонується обробити всі поля даних з метою їх подальшої використання для первинного його наповнення у разі застосування хешованих ПД. Наступним кроком передбачається поступове внесення всіх записів шляхом додаткової обробки кожного запису. Така обробка передбачає створення додаткової таблиці БД, що буде містити записи для кожного запису даних у вигляді додаткових полів з хеш-кодами від шаблонів комбінацій полів даних, вже попередньо оброблених та представлених у вигляді хеш-кодів (див. табл. 4.3).

Табл. 4.3. Шаблони комбінацій даних

Хеш-код	Шаблони комбінацій
1	$X_1 + \dots + X_M + \text{choice}_U^L (Z)$
...	$X_1 + \dots + X_M + \text{choice}_U^L (Z)$
Q	$X_1 + \dots + X_M + \text{choice}_U^L (Z)$

Кожен шаблон комбінації перетворюється в 64-байтовий хеш-код за допомогою одностороннього алгоритму хешування SHA-512. До кожного результуючого коду додається додатковий байт, який вказує кількість відсутніх полів даних для хеш-коду. Кожна комбінація є достатньою для того, щоб впевнено розрізняти об'єкт. Далі, генерується випадковий унікальний код UID і зв'язується з хеш-кодами комбінацій. UID і його зв'язані хеш-коди зберігаються на сервері і використовуються для визначення об'єкту.

Правило співставлення хеш-кодів та об'єктів. Кожний хеш-код складається з 64-байтового хеш-значення, яке обчислюється з шаблону

комбінації ПД за допомогою одностороннього алгоритму хешування та додаванням 1 додаткового байту, яке вміщує кількість пропущених полів даних у хеш-кодi. Таким чином, будь-яка помилка в полі даних, що використовуються в комбінації, призведе до помилки співставлення хеш-коду.

Пропонується використовувати 3 типи хеш-кодів: ідеальний, хороший і поганий. Для кожного хеш-коду використовуються 2 параметри для визначення його типу: нижній поріг (L) і верхній поріг (U) (див. табл. 4.4). Для ідеального хеш-коду вимагається, щоб кількість пропущених полів даних дорівнювала або була меншою L. Кількість відсутніх полів даних для генерації хорошого хеш-коду обмежується інтервалом (L, U). Якщо кількість пропущених полів даних більше U, то хеш-код буде визначений як поганий. Співпадіння k ідеальних хеш-кодів для двох записів БД свідчить про те, що вони ймовірно належать одному і тому ж об'єкту, а співпадіння між двома хорошими хеш-кодами для двох записів визначають ці хеш-коди ймовірними кандидатами, що належать одному і тому ж об'єкту.

Табл. 4.4. Пороги відсутніх полів для визначення типу хеш-коду

Параметри	Шаблон 1	...	Шаблон Q
Нижній поріг	l_1	...	l_Q
Верхній поріг	u_1	...	u_Q

При введенні нових записів до інформаційного масиву автоматично підраховується кількість ідеальних та хороших співпадінь. Це надасть можливість з'ясувати, чи є вже даний об'єкт у БД. Для цього використовуються 3 параметри для визначення відповідності об'єкту: поріг для ідеального співпадіння (P), поріг для хорошого співпадіння (G) і поріг для змішаного співпадіння (X). Два записи збігаються один з одним, якщо кількість ідеальних співпадінь більша або рівна P, або кількість хороших співпадінь більша або рівна G, або сума ідеальних і хороших співпадінь більша або рівна X.

Імовірні шаблони комбінації даних для ідеальних або хороших хеш-кодів.

Хеш-коди генеруються з комбінацій полів даних в зведеному масиві, тому кожен з них може розглядатися як набір перетворених полів даних. Крім того, у різних хеш-кодах є поля, що перекриваються. Завдяки цьому можливе використання теорії множин для систематичної перевірки сумнівних полів даних. Доки хеш-код узгоджується, його відповідні поля з даними можуть бути виключені операціями над множинами з пошуку сумнівних полів даних. Оскільки дозволено мати відсутні значення опціональних полів даних, то спочатку необхідно проаналізувати всі ймовірні схеми поєднання полів даних для ідеальних та хороших хеш-кодів, після чого можна розробити алгоритм перевірки сумнівних даних.

Кожен хеш-код генерується з різних шаблонів комбінацій полів даних, які є опціональними або обов'язковими. На основі розроблених шаблонів комбінацій даних можуть бути проаналізовані та ідентифіковані всі ймовірні ідеальні або хороші хеш-коди, визначені правила співставлення хеш-кодів (див. табл. 4.5).

Табл. 4.5. Імовірні шаблони комбінації ПД для ідеальних або хороших хеш-кодів

Індекс	Хеш-код	Комбінації полів з ПД						Відсутні поля	Тип хеш-коду
1	1	X_1	...	X_M	Z_1	...	Z_T	...	Ід
...	...	X_1	...	X_M	Z_1	...	Z_T	...	
...	...	X_1	...	X_M	Z_1	...	Z_T	...	
R	...	X_1	...	X_M	Z_1	...	Z_T	...	Хор

Оскільки хеш-код формується з комбінації полів даних, він пов'язаний з набором цих полів. Як тільки він збігається з одним з хеш-кодів ідентифікованого об'єкту, відповідний набір полів даних також є таким, що збігається, і ці поля даних вважатимуться перевіреними.

4.3.2 Уточнення даних щодо сумнівних записів

Відповідно до етапу первинного формування зведеного масиву, кожний запис реєстру може знаходитися в одному з чотирьох станів:

- а) “verified” – запис верифікований з еталонним реєстром;
- б) “approved” – запис визнано верифікованим через зіставлення даних з різних реєстрів, але для запису відсутній аналогічний запис в еталонному реєстрі;
- в) “unapproved” – запис визнано не верифікованим через брак ідентичних даних в різних реєстрах;
- г) “unknown” – запис щодо об’єкту, який створено вперше і відсутні інші джерела для порівняння.

Після зіставлення реєстрів для кожного запису з статусами “approved” і “verified” генерується унікальний ідентифікатор.

Для записів “unapproved” потрібно провести ручну верифікацію.

Приклад використання запропонованого методу представлено в розділі 5.7.

4.4 Висновки до четвертого розділу

У даному розділі представлений детальний опис методу аналізу стану і оцінювання елементів проектних рішень критичної інформаційної інфраструктури, побудованого за допомогою методу структурної оптимізації нейронних мереж, наведено основні базові елементи та операції методу. Запропонований метод дозволяє побудувати класифікатори на базі нейронних мереж з оптимальною структурою та використати їх для оцінювання елементів проектних рішень критичної ІТ-архітектури та їх впливу на інші елементи та архітектуру в цілому. Далі наведений опис моделей управління ресурсами критичної інформаційної інфраструктури, побудованих з використанням

нечіткої логіки, генетичних алгоритмів та методу рою часток. І насамкінець, наведений детальний опис методу верифікації даних в інформаційних ресурсах критичної інформаційної інфраструктури на прикладі одного з сервісів ЄІС МВС, який дозволяє підвищити якість даних шляхом усунення дублювання, неправильних значень тощо.

РОЗДІЛ 5

ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ЗАПРОПОНОВАНИХ В РОБОТІ МОДЕЛЕЙ ТА МЕТОДІВ

5.1 Єдина інформаційна система МВС України

Єдина інформаційна система МВС України (далі – ЄІС МВС) – критична інформаційна інфраструктура (включена до державного реєстру об’єктів критичної інформаційної інфраструктури), яка є основним елементом побудови спільного інформаційного простору органів системи МВС. Згідно з [212] ЄІС МВС складається з постійно діючих:

- центральної підсистеми;
- функціональних підсистем;
- транспортної мережі передачі даних;
- центрів обробки даних, телекомунікаційних мереж суб’єктів єдиної інформаційної системи МВС;
- комплексних систем захисту інформації підсистем єдиної інформаційної системи МВС з підтвердженою в установленому законодавством порядку відповідністю.

Центральна підсистема ЄІС МВС (далі – ЦП ЄІС МВС) – це програмний комплекс, що забезпечує функціонування автоматизованих сервісів та сукупності взаємопов’язаних функціональних підсистем, програмно-інформаційних комплексів, програмно-технічних та технічних засобів телекомунікації, які забезпечують логічне поєднання визначених інформаційних ресурсів, обробку та захист інформації, внутрішню та зовнішню інформаційну взаємодію, безперервність обслуговування в інтерактивному режимі, наближеному до реального часу та продуктивність.

Призначення ЦП ЄІС МВС:

- оптимізація та автоматизація ділових процесів діяльності суб’єктів ЄІС МВС під час виконання завдань та функцій, покладених на них

законодавством, з метою підвищення ефективності такої діяльності для забезпечення інформаційних потреб цих суб'єктів;

- створення єдиного інформаційного простору як основи для подальшого розвитку електронних послуг для громадян;
- інтеграція інформаційних ресурсів ЄІС МВС;
- забезпечення електронної інформаційної взаємодії між суб'єктами ЄІС МВС;
- розмежування прав доступу та надання контрольованого доступу користувачам ЄІС МВС до інформаційних ресурсів ЄІС МВС.

5.1.1 Мета створення і завдання ЦП ЄІС МВС

Метою створення ЦП ЄІС МВС є побудова базової програмної інфраструктури для об'єднання інформаційних ресурсів ЄІС МВС, забезпечення їх використання для підтримки типових видів діяльності МВС на основі єдиних системних і технологічних рішень.

Для досягнення мети ЦП ЄІС МВС повинна надати можливість здійснення інформаційної взаємодії суб'єктів ЄІС МВС на рівні загальнодержавних інформаційних ресурсів в інтересах національної безпеки, захисту прав та законних інтересів громадян, суспільства і держави у сферах:

- забезпечення охорони прав і свобод людини, інтересів суспільства і держави, протидії злочинності, підтримання публічної безпеки і порядку;
- цивільного захисту, захисту населення і територій від надзвичайних ситуацій та запобігання їх виникненню, ліквідації наслідків надзвичайних ситуацій, рятувальної справи, гасіння пожеж, пожежної та техногенної безпеки, діяльності аварійно-рятувальних служб, а також гідрометеорологічної діяльності;
- захисту персональних даних користувачів під час їх обробки та зберігання, у відповідності з діючим законодавством та законодавством країни суб'єкта обробки персональних даних;

- міграції (імміграції та еміграції), у тому числі протидії нелегальній (незаконній) міграції, громадянства, реєстрації фізичних осіб, біженців та інших визначених законодавством категорій мігрантів.

Завданнями ЦП ЄІС МВС є:

- створення єдиного інформаційного простору системи МВС та центральних органів виконавчої влади, діяльність яких спрямовується і координується Кабінетом Міністрів України через Міністра внутрішніх справ шляхом логічного об'єднання їх інформаційних ресурсів, оптимізація процесів спільного використання технічних та програмних ресурсів;
- інформаційна підтримка діяльності суб'єктів єдиної інформаційної системи МВС під час виконання завдань та функцій, покладених на них законодавством, з метою підвищення її ефективності;
- створення умов для електронної взаємодії суб'єктів єдиної інформаційної системи МВС з метою оперативного виконання завдань, покладених на них законодавством, зменшення часових та фінансових витрат на адміністративно-управлінські, інформаційно-пошукові, розрахункові та аналітичні роботи, формування звітності;
- взаємодії з державними органами, органами місцевого самоврядування, міжнародними організаціями, суб'єктами господарювання та правоохоронними органами інших держав, у тому числі з використанням загальнодержавних інформаційно-телекомунікаційних систем, системи електронної взаємодії державних електронних інформаційних ресурсів “Трембіта” та інтегрованої системи електронної ідентифікації.

5.1.2 Концепція ЦП ЄІС МВС

Єдина інформаційна система МВС – багатофункціональна інтегрована автоматизована система, яка в першу чергу повинна забезпечувати внутрішню та зовнішню взаємодію між суб'єктами в задачах отримання необхідної інформації з інформаційних ресурсів відповідно до системи прав доступу.

Суб'єктами ЄІС МВС – джерелами інформації - виступають:

- апарат МВС та його територіальні органи з надання сервісних послуг МВС;
- Національна гвардія, заклади, установи і підприємства, що належать до сфери управління МВС;
- центральні органи виконавчої влади, діяльність яких спрямовується і координується Кабінетом Міністрів України через Міністра внутрішніх справ,
- інші державні органи, які обробляють інформацію в єдиній інформаційній системі МВС.

Об'єктом ЄІС МВС виступає інформаційний ресурс – групи взаємозв'язаних задокументованих одиниць інформації, які формуються і об'єднуються в автоматизованих інформаційних системах суб'єктів єдиної інформаційної системи МВС за певними ознаками.

Компоненти ЄІС МВС представлені на рис. 5.1.

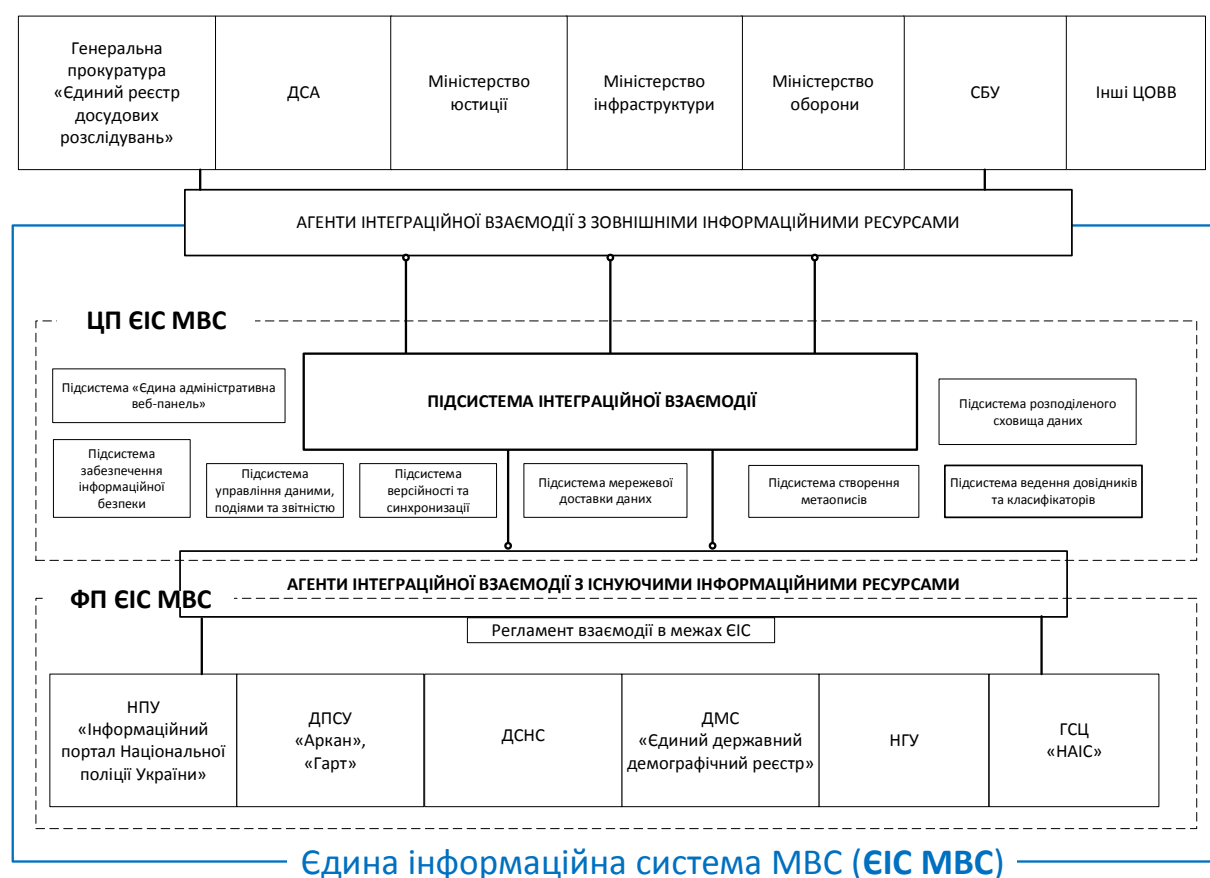


Рис. 5.1. Складові ЄІС МВС

ЦП ЄІС МВС – центральна підсистема ЄІС МВС – до складу якої також входять агенти інтеграційної взаємодії з внутрішніми та зовнішніми інформаційними ресурсами.

ФП ЄІС МВС – функціональні підсистеми ЄІС МВС – це сукупність технічних засобів та програмних комплексів, які автоматизують службові процеси суб'єктів єдиної інформаційної системи МВС до рівня стандартів операційних процедур та автоматизованого робочого місця користувача, забезпечують формування, зберігання, спільне використання і верифікацію інформаційних ресурсів єдиної інформаційної системи МВС.

Користувачами інформації в межах ЄІС МВС виступають:

- уповноважені співробітники;
- зовнішні користувачі, для перегляду власної інформації фізичною особою;
- функціональні підсистеми, які входять до складу ЄІС МВС;
- зовнішні функціональні підсистеми (державні інформаційні ресурси).

ЄІС МВС повинна забезпечити отримання необхідної для користувача інформації, як логічного об'єднання даних з різних інформаційних ресурсів (далі, інформаційних об'єктів) одним запитом. При цьому доступ до інформаційних об'єктів повинен відбуватися у суворій відповідності до повноважень. Керування доступом повинно здійснюватися ЦП ЄІС МВС.

Інформаційний об'єкт належить як ЄІС МВС, так і зовнішнім підсистемам. Обмін даними з зовнішніми підсистемами здійснюється через систему електронної взаємодії «Trembita» або аналогічною за властивостями системою.

ЦП ЄІС МВС повинна вести журнал кожного факту запиту до інформаційних об'єктів з використанням сервісу логування, який контролює і робить неможливим будь-яке внесення змін, в тому числі і адміністраторами системи, фіксація подій і контроль цілісності відбувається шляхом побудови взаємопов'язаних блоків з використанням криптографічних функцій та/або технології блокчейн. Для визначених видів даних повинна бути передбачена

можливість використання ролевої моделі для визначення обґрунтування причини запиту до інформаційних об'єктів, які відповідають отриманій інформації. Рольова модель ЦП ЄІС МВС визначається відповідно до розпорядчих документів.

Загальна структура взаємодії наведена на рис. 5.2.

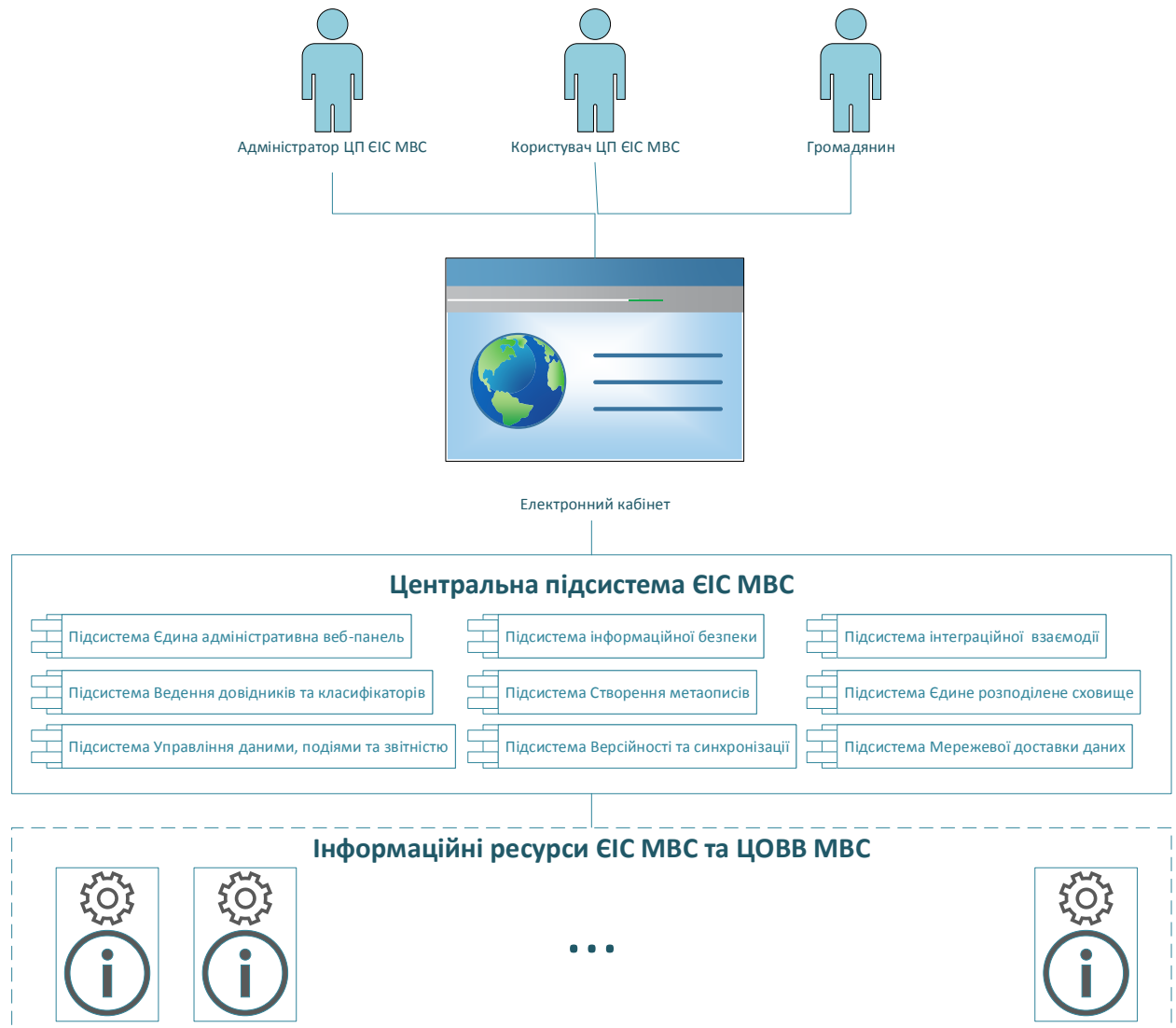


Рис. 5.2. Структурна схема ЦП ЄІС МВС

Для забезпечення інтероперабельності, проте з урахуванням наданих повноважень, існуючі системи або майбутні функціональні підсистеми суб'єктів ЄІС МВС необхідно розробити відповідні програмні інтерфейси (API) ЦП ЄІС МВС або SDK, що спростить підключення для систем споживачів інформації. Наявність таких API ЦП ЄІС МВС звільнить від необхідності написання

підсистем, що направлені на можливість взаємодії з кожним конкретним інформаційним ресурсом. Проте, в свою чергу, це призведе до необхідності доопрацьовувати кожну існуючу систему для налаштування взаємодії з ЦП ЄІС МВС.

Таким чином, завдяки впровадженню ЦП ЄІС МВС будь-які інформаційні ресурси, як внутрішні, так і зовнішні, стають прозорими для всіх функціональних підсистем ЄІС МВС, тому що їм не потрібно «знати» до якого саме інформаційного ресурсу необхідно звертатися для отримання необхідних даних, цю інформацію визначає сервіс ЦП ЄІС МВС і запит повинен здійснюватися лише через ЦП ЄІС МВС.

5.1.3 Функціональна структура ЦП ЄІС МВС

Загальна функціональна структура ЦП ЄІС МВС наведена на рис. А.6.

Далі наведено опис вимог до функціональності складових ЦП ЄІС МВС.

5.1.3.1 Функціональні задачі підсистеми інформаційної безпеки

Підсистема має гарантувати стан захищеності систем обробки і зберігання даних ЦП ЄІС МВС, при якому має бути забезпечено конфіденційність, доступність і цілісність інформації, та включає:

- визначення конфігурації організаційної структури ЄІС МВС, включаючи суб'єктів МВС, їх підрозділи та функціональні підсистеми (створення, редагування та ін.);
- управління користувачами ЦП ЄІС МВС та їх ролями (створення, редагування та ін.), включаючи користувачів - відповідальних осіб суб'єктів МВС, зовнішніх користувачів, підсистеми та сервіси, які можуть виступати в ролі користувачів;
- адміністрування прав доступу користувачів до інформаційних ресурсів на основі рольової моделі;

- криптографічну ідентифікацію, аутентифікацію та авторизацію користувачів та сервісів функціональних систем суб'єктів МВС при автоматизованому доступі до інформаційних об'єктів ЦП ЄІС МВС;
- контроль відповідності форматів, складу та цілісності даних в процесах обміну інформацією;
- використання захищених засобів доступу до платформу, у тому числі і мобільних;
- криптографічний захист інформації;
- апаратно-програмний захист інформації;
- апаратно програмний захист ключових даних;
- апаратно-програмне шифрування даних;
- захист від загроз несанкціонованого вторгнення та злому.

5.1.3.2 Функціональні задачі підсистеми створення метаописів

Підсистема передбачає:

- Створення, реєстрацію існуючих, модифікацію та конфігурацію метаданих інформаційних об'єктів ЦП ЄІС МВС, а саме:
 - формати структур даних інформаційних об'єктів (довідників, класифікаторів, реєстрів та ін.);
 - метадані сервісів та їх функцій, включаючи формати запитів та результатів, що повертаються;
 - зв'язки між сервісами та відповідними інформаційними об'єктами, включаючи правила отримання, перетворення та завантаження даних (ETL-процеси) у разі необхідності.
- Створення, реєстрацію, модифікацію та конфігурацію функціональних (ділових) процесів, їх зв'язок з інформаційними об'єктами, інтерфейсами користувачів та ін. При цьому можуть використовуватись як вбудовані, так і зовнішні інструменти моделювання процесів та/або програмування процесів.

- Публікація метаописів інформаційних об’єктів та процесів для доступу та інтеграції з іншими підсистемами ЦП ЄІС МВС та, при необхідності, інших функціональних систем суб’єктів МВС з використанням JSON-схем, XML-схем та ін.

5.1.3.3 Функціональні задачі підсистеми інтеграційної взаємодії

Підсистема передбачає:

- Надання стандартизованих програмних інтерфейсів (API) та /або SDK іншим підсистемам ЦП ЄІС МВС та функціональним підсистемам суб’єктів МВС для взаємодії за стандартизованими протоколами (REST, RPC).
- Надання API для взаємодії з інформаційними ресурсами інших органів державної влади шляхом інтеграції з системами взаємодії, у тому числі “Трембіта”.
- Передача службових та інформаційних даних запитів в підсистему інформаційної безпеки для виконання ідентифікації, аутентифікації та авторизації користувачів та надання доступу до відповідних інформаційних ресурсів за результатами авторизації.
- Маршрутизація викликів методів сервісів інших підсистем ЄІС МВС, функціональних підсистемам суб’єктів МВС та інших доступних інформаційних ресурсів за єдиними визначеними правилами.
- Виконання заданих функціональних процесів та трансформації даних відповідно до метаданих, заданих в підсистемі створення метаописів.
- Забезпечення безпечного, шифрованого обміну інформацією між підсистемами та системами.
- Забезпечення відправки повідомлень користувачам доступними каналами (e-mail, SMS, push-повідомлення та ін.).
- Передача даних в підсистему управління даними, подіями та звітністю з метою логування процесів обміну, їх моніторингу та контролю, побудови оперативної та статистичної звітності.

5.1.3.4 Функціональні задачі підсистеми управління даними, подіями та звітністю

Підсистема має забезпечити:

- Логування процесів аутентифікації та авторизації користувачів.
- Логування дій користувачів за визначеними адміністратором критеріями.
- Логування процесів обміну даними та повідомленнями між функціональними підсистемами суб'єктів МВС та функціональними підсистемами суб'єктів МВС з іншими підсистемами ЄІС МВС.
- Логування системних подій, збоїв та проблем в роботі ЦП ЄІС МВС.
- Резервне копіювання (включаючи інкрементне) за розкладом та/або за командою адміністратора, зберігання та відновлення даних з використанням резервних копій у разі необхідності.
- Перенесення даних до архівних сховищ для довгострокового зберігання.
- Забезпечення процесів моніторингу та контролю шляхом візуалізації процесів логування та відправки повідомлень адміністратору ЦП ЄІС МВС, адміністраторам функціональних підсистем ЄІС МВС та іншим користувачам про визначені події згідно налаштуванням.
- Підготовка та формування оперативної та статистичної звітності.

5.1.3.5 Функціональні задачі підсистеми мережевої доставки даних

Підсистема має забезпечити гарантовану доставку даних між підсистемами ЄІС МВС та іншими користувачами ЄІС МВС, при цьому можуть використовуватись:

- канали з апаратним шифруванням трафіку;
- внутрішні канали МВС без шифрування трафіку, але такі, що відповідають вимогам КСЗІ;

- зовнішні інтернет-канали.

5.1.3.6 Функціональні задачі підсистеми ведення довідників та класифікаторів

Основна задача підсистеми - ведення централізованої нормативно-довідкової інформації (класифікаторів та довідників) з метою:

- забезпечення однозначної ідентифікації інформації при введенні і подальшому пошуку;
- забезпечення актуальності нормативно-довідкової інформації.
- Підсистема повинна включати ведення наступних типів довідників:
- загальнодержавні та міжнародні довідники та класифікатори (КОАТУУ, КВЕД, КОПФГ та ін.);
- відомчі довідники суб'єктів МВС, що можуть використовуватись декількома суб'єктами МВС та/або зовнішніми користувачами (напр. “Умовні позначення областей України на номерних знаках автомобілів”).

При цьому має бути забезпечено:

- ведення загальнодержавних довідників та класифікаторів в підсистемі “Єдине розподілене сховище” шляхом моніторингу та коригування змін у відповідних довідниках відповідальними особами ЄІС МВС та/або їх автоматична синхронізація з відповідними зовнішніми джерелами у разі можливості;
- ведення відомчих довідників суб'єктів МВС у відповідних функціональних підсистемах суб'єктів-власників та їх публікація в підсистемі шляхом надання відповідного сервісу та/або їх автоматизована синхронізація з відповідними об'єктами в підсистемі “Єдине розподілене сховище” з метою кешування та прискорення доступу у разі відсутності обмежень з боку вимог конфіденційності. Права доступу користувачів до таких довідників встановлюються адміністратором відповідного суб'єкту.

5.1.3.7 Функціональні задачі підсистеми версійності та синхронізації

Підсистема має забезпечити безперервне функціонування всіх підсистем ЦП ЄІС МВС та функціональних підсистем суб'єктів ЄІС МВС у разі зміни метаданих інформаційних об'єктів, функціональних процесів та інших інформаційних об'єктів.

При цьому:

- у разі внесення змін у відповідні метадані автоматично створюється нова версія метаданих, яка стає доступна після її публікації;
- стара версія залишається функціональною та доступною для використання іншими підсистемами;
- стара версія передбачає видалення (з перенесенням в архів) тільки після переходу всіх підсистем, що використовують даний інформаційний об'єкт, на нову версію;
- при публікації нових версій має бути забезпечене фіксування події в сервісі логування та інформування адміністраторів відповідних функціональних підсистем.

На рівні окремих інформаційних об'єктів, включаючи окремі записи передбачено забезпечення того, що інформація, яка внесена і зберігається у ЦП ЄІС МВС та зазнала будь яких змін, зберігається в новій версії відповідного інформаційного об'єкта. Зберігається також інформація про всі внесені зміни.

5.1.3.8 Функціональні задачі підсистеми Єдина адміністративна веб-панель

Підсистема має забезпечити типовий користувацький веб-інтерфейс до підсистеми інформаційної безпеки та реалізувати можливість:

- конфігурації організаційної структури ЄІС МВС;
- управління користувачами ЦП ЄІС МВС та їх ролями;
- адміністрування прав доступу користувачів до інформаційних

ресурсів;

- надання інтерфейсу для управління функціями інших підсистем ЄІС МВС в залежності від прав користувача (адміністратора відповідного рівня).

При цьому доступ до функцій підсистеми має бути забезпечено після аутентифікації та авторизації користувача, як адміністратора ЦП ЄІС МВС або адміністратора суб'єкта МВС.

Для аутентифікації має бути забезпечено вхід користувача по КЕП.

В залежності від ролі користувачів підсистема умовно розділена на:

- Кабінет адміністратора ЦП ЄІС МВС (Адміністратор ЦП ЄІС МВС призначається Власником ЦП ЄІС МВС) доступ до якого відбувається в тому числі і за допомогою захищених мобільних пристроїв, в якому доступні функції:

- управління загальними налаштуваннями ЦП ЄІС МВС;
- управління суб'єктами ЄІС МВС;
- управління загально-системними та зовнішніми користувачами та ролями ЄІС МВС;
- надання прав доступу до ресурсів власником яких є ЦП ЄІС МВС (виступає також в ролі одного з суб'єктів ЄІС МВС та передбачає власні ресурси, наприклад різні класифікатори та загальні довідники);
- делегування повноважень Адміністратор ЦП ЄІС МВС іншій відповідальній особі;
- надання ролі Адміністратора суб'єкту ЄІС МВС відповідальній особі відповідного суб'єкту.

- Кабінет адміністратора суб'єкту ЄІС МВС (Адміністратор суб'єкту ЄІС МВС призначається Адміністратором ФП ЄІС МВС у прив'язці до його суб'єкту ЄІС МВС), в якій доступні функції:

- реєстрація користувача – відповідальної особи свого суб'єкту ЄІС;
- надання прав доступу іншим користувачам та ролям, власниками яких є інші суб'єкти ЄІС МВС, до ресурсів свого суб'єкту ЄІС МВС.

Додатково передбачена Панель моніторингу та контролю, яка забезпечує веб-інтерфейс до підсистеми управління даними, подіями та звітністю, підсистеми версійності та синхронізації з обмеженням доступу в залежності від ролі адміністратора.

5.1.3.9 Функціональні задачі підсистеми Єдине розподілене сховище

Єдине розподілене сховище призначене для зберігання даних, що потрібні для реалізації функцій інших функціональних підсистем ЄІС МВС та передбачає реалізацію як декілька логічно взаємозалежних баз даних та файлових сховищ, розподілених у комп'ютерній мережі та надає такі механізми доступу до них, що їх застосування дає можливість функціональними підсистемам ЄІС МВС працювати з ним як з однієї цілісної базою даних та/або використовувати однією підсистемою даних, основна обробка яких ведеться в іншій підсистемі.

Дані між різними базами даних та файловими сховищами можуть бути розподілені:

- по функціональним підсистемам ЄІС МВС та/або їх функціональним блокам (наприклад дані користувачів для аутентифікації та авторизації можуть зберігатись в одній БД, а логування дій користувачів в іншій БД);
- відповідно вимогам надійності та доступності (наприклад основні, резервні, дзеркальні);
- однотипні структури по окремим критеріям (наприклад логи однакової структури, але для різних підсистем);

Інші методи розділення, наприклад кластеризація, що будуть забезпечувати надійність, доступність та високу продуктивність роботи ЦП ЄІС МВС в цілому.

В єдиному розподіленому сховищі передбачена наскрізна ідентифікація інформаційних об'єктів. Унікальний наскрізний ідентифікатор є уніфікованим єдиним засобом ідентифікації для всіх інформаційних об'єктів, що будуть включені до ЦП ЄІС МВС чи інтегровані з нею, що дозволить:

- створення фундаменту для розбудови якісно нової системи державних інформаційних ресурсів;
- інтеграцію різних баз персональних даних, а також організацію до них мережевого розподіленого доступу;
- формування єдиних методів реєстрації, ідентифікації і аутентифікації користувачів у всіх функціональних системах ЄІС МВС;
- реалізацію санкціонованого доступу учасників інформаційної взаємодії до інформації, що міститься в державних інформаційних системах.

5.2 Дослідження моделі оцінки впливу забезпечуючих систем на архітектуру критичної інформаційної інфраструктури

Для моделювання побудовані спрощені моделі деяких компонент критичної інформаційної інфраструктури та досліджено за їх допомогою можливість використовувати запропонований підхід. Цей підхід дозволяє створювати композиції моделей на основі взаємозалежностей, які існують між ними, об'єднуючи їх у більш складні моделі, і таким чином, утворювати наступні рівні абстракції моделей. На простих моделях перевірена працездатність запропонованого підходу – створені декілька моделей в пакеті Matlab, досліджена їх робота, отримані очікувані результати

Бібліотека моделей РТС компонент та систем критичної інформаційної інфраструктури для Simulink/Stateflow надасть можливість дослідникам проводити моделювання будь-яких взаємозв'язків між системами, компонентами критичної інформаційної інфраструктури [7].

РВГА ІТС складається з композиції двох елементів:

- центру мережевих операцій (network operation center);
- мережі.

Опишемо модель центру мережевих операцій у термінах РВГА. Модель містить 5 дискретних станів:

- «Звичайний» – стан звичайної роботи центру мережних операцій (ЦМО);
- «Безперебійне живлення» – стан роботи ЦМО на пристроях безперебійного живлення у випадку відсутності струму в електричній мережі;
- «Помилки охолодження» – стан роботи ЦМО у випадку аварій в системі охолодження обладнання;
- «Критичний» – стан роботи ЦМО у випадку одночасної відмови систем живлення та охолодження;
- «Аварія» – стан ЦМО, у якому подальша робота не можлива через відмови, що сталися.

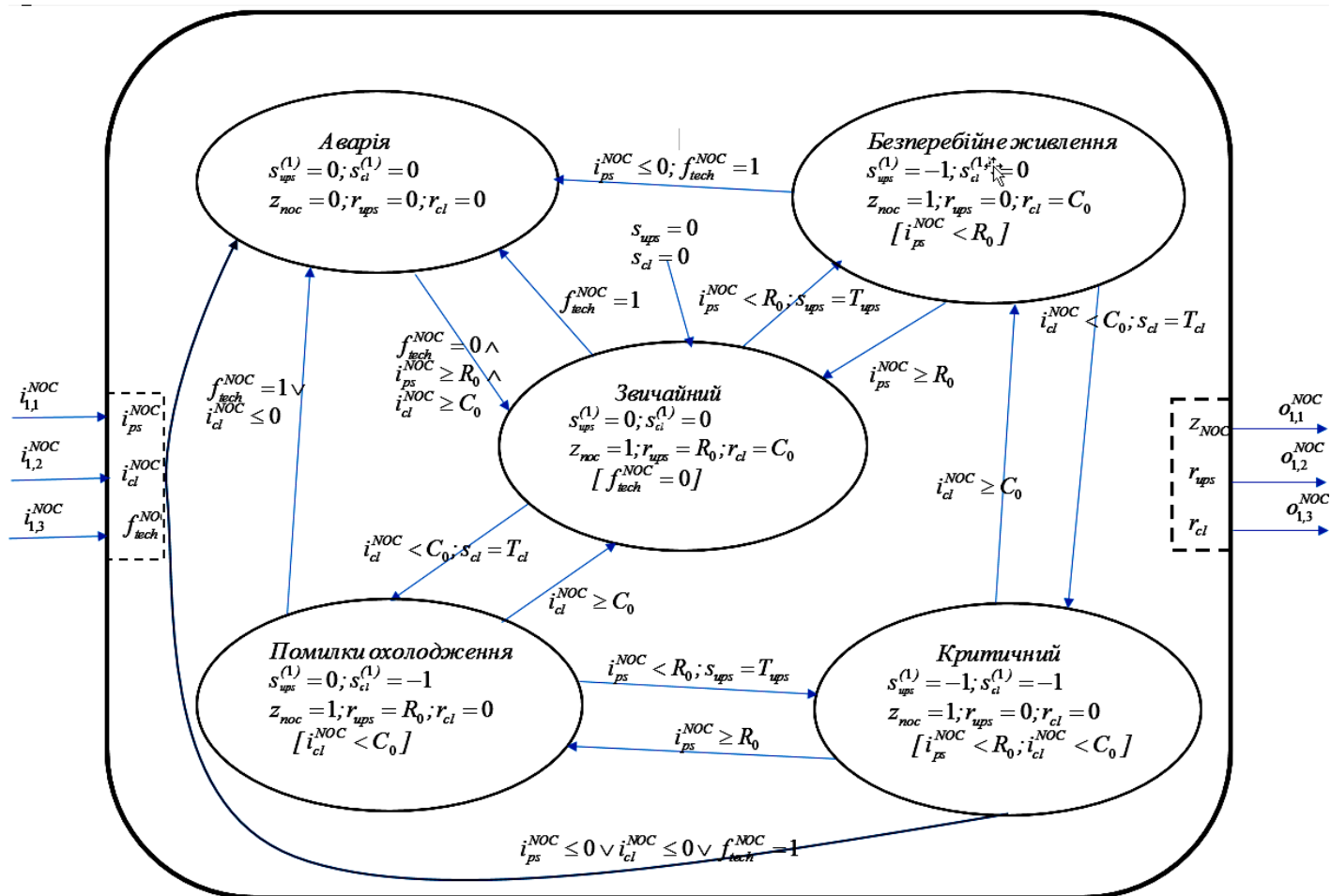


Рис. 5.3. Модель роботи центру мережних операцій

Модель має наступні входи (рис. 5.3):

- i_{ps}^{NOC} – живлення ЦМО;

- i_d^{NOC} – охолодження ЦМО;
- f_{tech}^{NOC} – наявність технічних відмов у ЦМО. Виникнення технічних

відмов контролюється розподілом F або самою системою.

Перехід зі стану «Звичайний» в інші стани відбувається при наступних умовах:

- якщо виникла технічна відмова $f_{tech}^{NOC} = 1$;
- якщо рівень постачання живлення впав нижче рівня потреби ЦМО $i_{ps}^{NOC} < R_0$;
- якщо рівень постачання хладогену впав нижче рівня потреби ЦМО $i_d^{NOC} < R_0$.

У якості диспетчерів часу в системі виступають значення безперервних станів s_{ups} та s_d . В разі відмови мережі електропостачання система безперебійного живлення може підтримувати роботу ЦМО $s_{ups} = T_{ups}$.

В разі відмови системи охолодження робота ЦМО підтримується ще деякий час, що дорівнює $s_d = T_d$.

Модель також має три виходи:

- статус роботи ЦМО z_{NOC} (z_{NOC} приймає 2 значення: 1 – ЦМО виконує свої функції, 0 – на ЦМО аварія);
- потреба ЦМО в електроживленні r_{ups} ;
- потреба ЦМО в охолодженні r_d .

Розглянемо наступну модель роботи мережі, що є варіантом моделі з [213].

Модель має три дискретні стани роботи (рис. 5.4):

- «Звичайний» – мережа працює у штатному режимі;
- «Відмова каналу передачі даних» – відмова одного або декілька каналів передачі даних;
- «Аварія» – мережа повністю відмовила.

Модель мережі має наступні входи:

- i_l^{NET} – мережа працює в штатному режимі, якщо немає відмов каналів передачі даних, тобто $i_l^{NET} = 0$;
- i_{noc}^{NET} – мережа працює у штатному режимі, якщо ЦМО також працює в штатному режимі, тобто $i_{noc}^{NET} = 1$;
- i_{ps}^{NET} – вхід, що отримує дані від системи електропостачання;
- i_{cl}^{NET} – вхід, що отримує дані від системи охолодження;
- i_{pct}^{NET} – вхід, що отримує дані про кількість вхідних пакетів, що потрапляють у мережу.

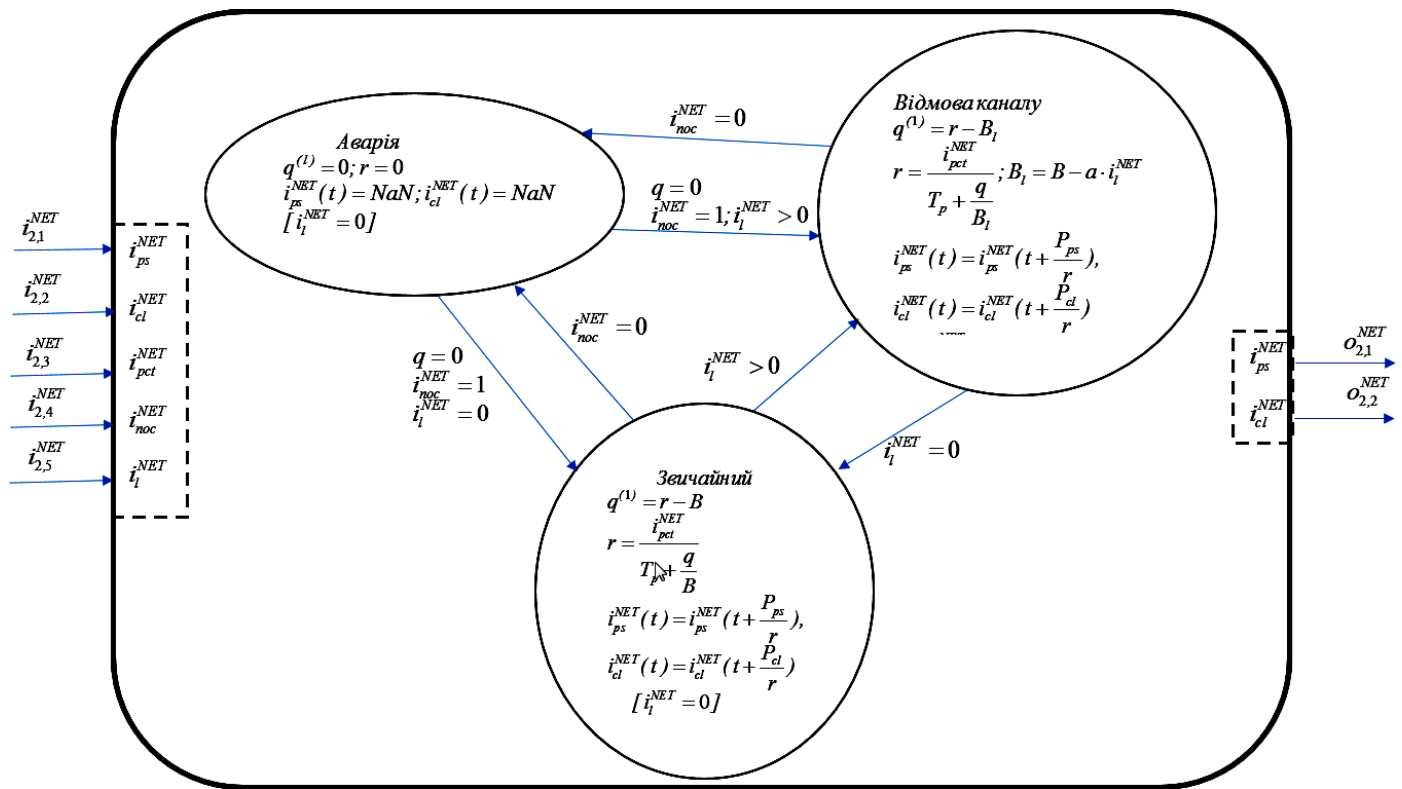


Рис. 5.4. Модель роботи мережі

Доки мережа працює у штатному режимі, швидкість передачі даних розраховується за формулою:

$$r = \frac{i_{pct}^{NET}}{T_p + \frac{q}{B}}, \quad (5.1)$$

де i_{ps}^{NET} – кількість пакетів, що входять в мережу; T_p – затримка розповсюдження; q – розмір черги; B – пропускна здатність мережі в пак/с.

Згідно з [10] розмір черги q описується наступним рівнянням динаміки:

$$q^{(1)} = r - B. \quad (5.2)$$

Дані від систем електропостачання та охолодження подаються в систему з затримкою:

$$\begin{aligned} i_{ps}^{NET}(t) &= i_{ps}^{NET} \left(t + \frac{P_{ps}}{r} \right), \\ i_d^{NET}(t) &= i_d^{NET} \left(t + \frac{P_d}{r} \right), \end{aligned} \quad (5.3)$$

де P_{ps}, P_d – кількість пакетів, що приходять від конкретної системи.

Таким чином, чим більше трафік в мережі, тим більші затримки розповсюдження. У випадку відмови каналів передачі даних, пропускна здатність мережі зменшується за формулою:

$$B_i = B - a \cdot i_i^{NET}, \quad (5.4)$$

де a – кількість каналів передачі даних, що відмовили.

У випадку, коли ЦМО в неробочому стані, тобто $i_{noc}^{NET} = 0$, відбувається перехід до стану «Аварія» і на всіх виходах системи маємо значення NaN.

Результати дослідження моделі. Для моделювання ЦМО побудована модель в пакеті Simulink/Stateflow (рис. 5.5). Вказана модель є дуже спрощеною, але достатньою для демонстрації запропонованого підходу для проектування та дослідження складних взаємозалежних систем за допомогою РВГА.

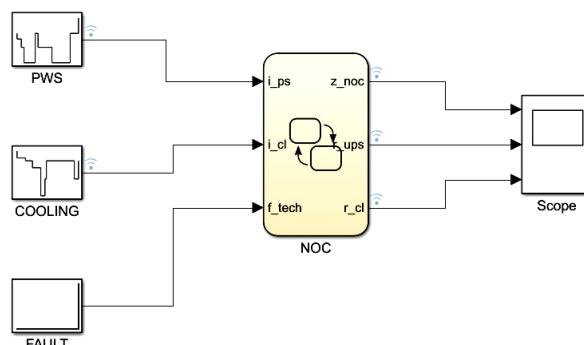


Рис. 5.5. Модель центру мережевих операцій в Simulink

Діаграма станів моделі ЦМО у StateFlow виглядає наступним чином (рис. 5.6).

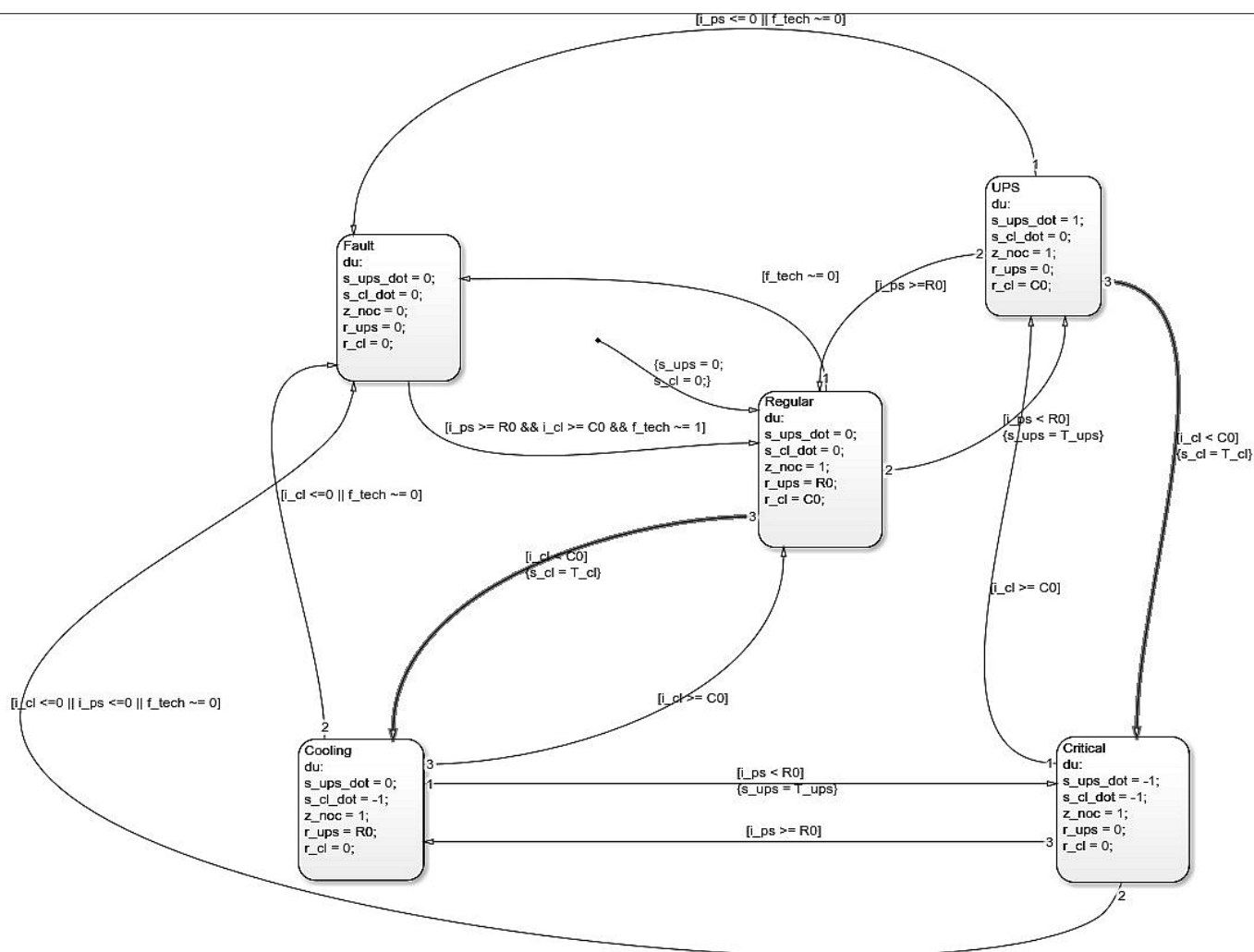


Рис. 5.6. Модель центру мережевих операцій в Stateflow

Для моделі встановлені наступні значення параметрів:

- $T_{ups} = 20$ с – час, який ЦМО може працювати на джерелах безперебійного живлення;
- $T_d = 10$ с – час, який ЦМО може працювати без системи охолодження;
- $R_0 = 3000$ Вт*час – мінімальний рівень електроенергії, потрібний для роботи ЦМО;
- $C_0 = 2000$ ВТУ/час – мінімальний рівень потужності охолодження, потрібний для роботи ЦМО.

На часових діаграмах представлені значення параметрів, що подаються на входи побудованої моделі (рис. 5.7–5.8).

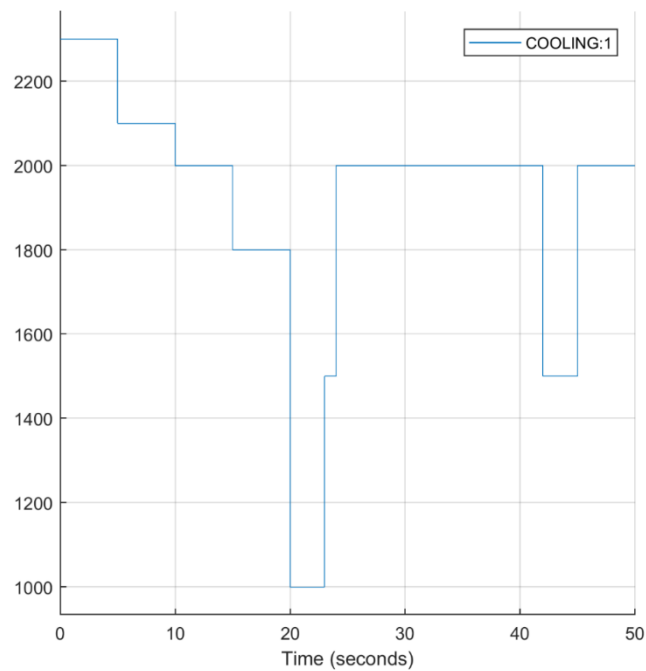


Рис. 5.7. Діаграма потужності охолодження на вході центру мережевих операцій

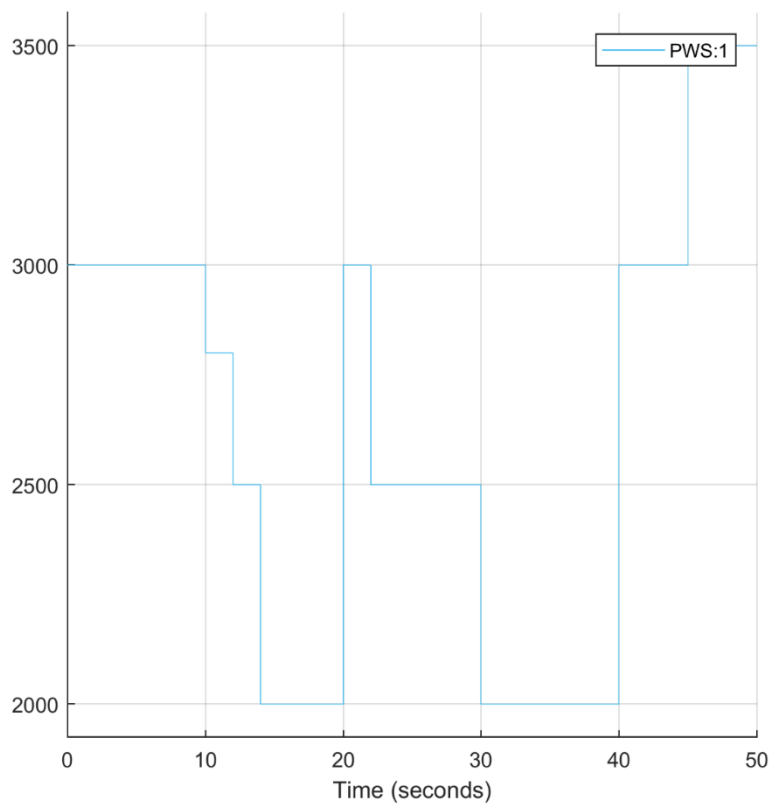


Рис. 5.8. Діаграма наявності електроенергії на вході центру мережевих операцій

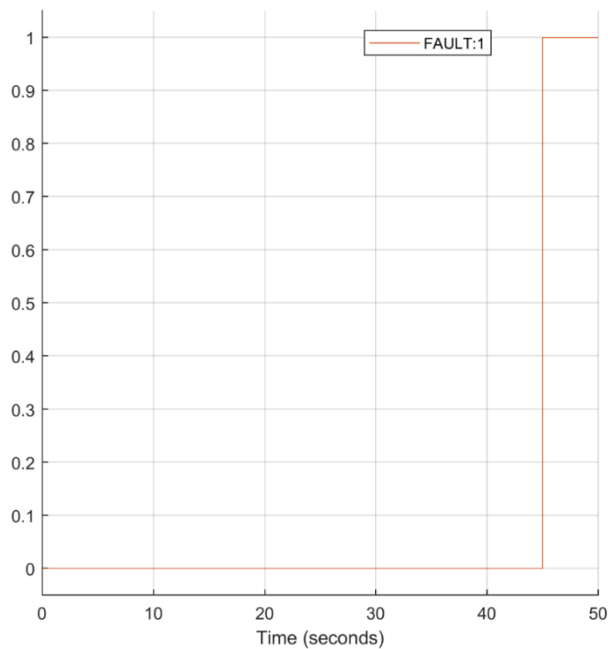


Рис. 5.9. Діаграма розподілу виникнення технічних відмов на вході центру мережеских операцій

Часові діаграми виходів моделі ЦМО представлені на рис. 5.10.

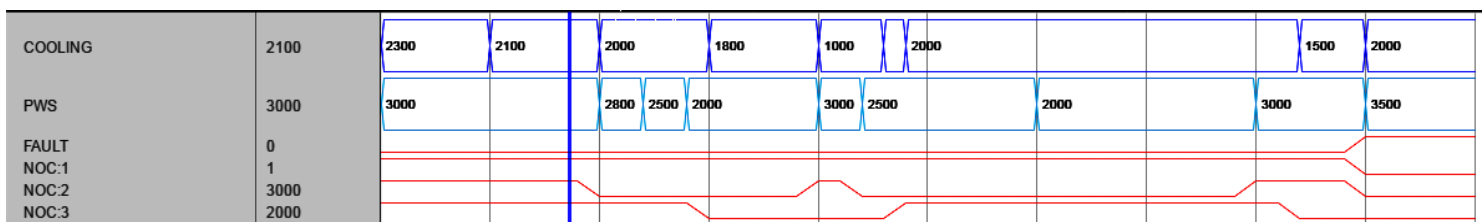


Рис. 5.10. Часові діаграми виходів

Діаграма на рис. 5.10 вказує на те, що ЦМО працює у заданому режимі, проходить відповідні стани в залежності від вхідних параметрів, які надходять від інших підсистем моделі.

Даний метод досліджень використаний при проектуванні ЦОД ЄІС МВС, що дозволило врахувати взаємозалежність систем охолодження та електроживлення на всю роботу ЦОД, і таким чином, якісно продумати резервування відповідних систем.

5.3 Дослідження методу оцінки та вибору оптимальної конфігурації компонент критичної інформаційної інфраструктури за допомогою Марківського процесу прийняття рішень

Для хмарних обчислень необхідно забезпечити керування застосуваннями, реалізувати взаємодію з платформами віртуалізації і мережною інфраструктурою під загальним сценарієм для всієї системи. Система в цілому має підтримувати велику кількість компонентів і надавати загальні інструменти керування, які можуть гарантувати надійне, безпечне і якісне надання послуг клієнтам. Мережна, апаратна і програмна інфраструктури хмарної системи мають відповідати існуючим і новим мережним стандартам:

- ISO/IEC 17789 (ITU-T Y.3502) – Інформаційні технології. Хмарні обчислення. Еталонна архітектура [214];
- NIST SP 500-291 – Стандарти хмарних обчислень [215];
- NIST SP 500-292 – Базова архітектура хмарних обчислень [216];
- ISO/IEC Committee Draft 27017 – Основи управління інформаційною безпекою для хмарних обчислень на базі ISO/IEC 27002 [217];
- ISO/IEC Draft International Standard 27018 – Основи захисту даних для публічних хмарних сервісів [218];
- ISO/IEC Working Draft 27036-4 – Інформаційна безпека відносин з постачальниками – Частина 4: Керівні принципи для забезпечення безпеки хмарних сервісів [219];
- ISO/IEC Draft International Standard 27040. Безпека систем зберігання даних IEEE P2301 – Профілі сумісності і переносимості (CPIP) [220];
- IEEE P2302 – Взаємодія хмарних систем (SIIF) [221];
- ANSI/TIA-942 [222], EN 50173-5 [223], ISO/IEC 24764 [224] – стандарти проектування хмарних дата-центрів.

Національний інститут стандартів і технологій США (National Institute of Standards and Technology – NIST) у документі [214] подав огляд еталонної

архітектури хмарних обчислень, який ідентифікує основних суб'єктів, їх діяльність і функції у хмарних обчисленнях (рис. 5.11).

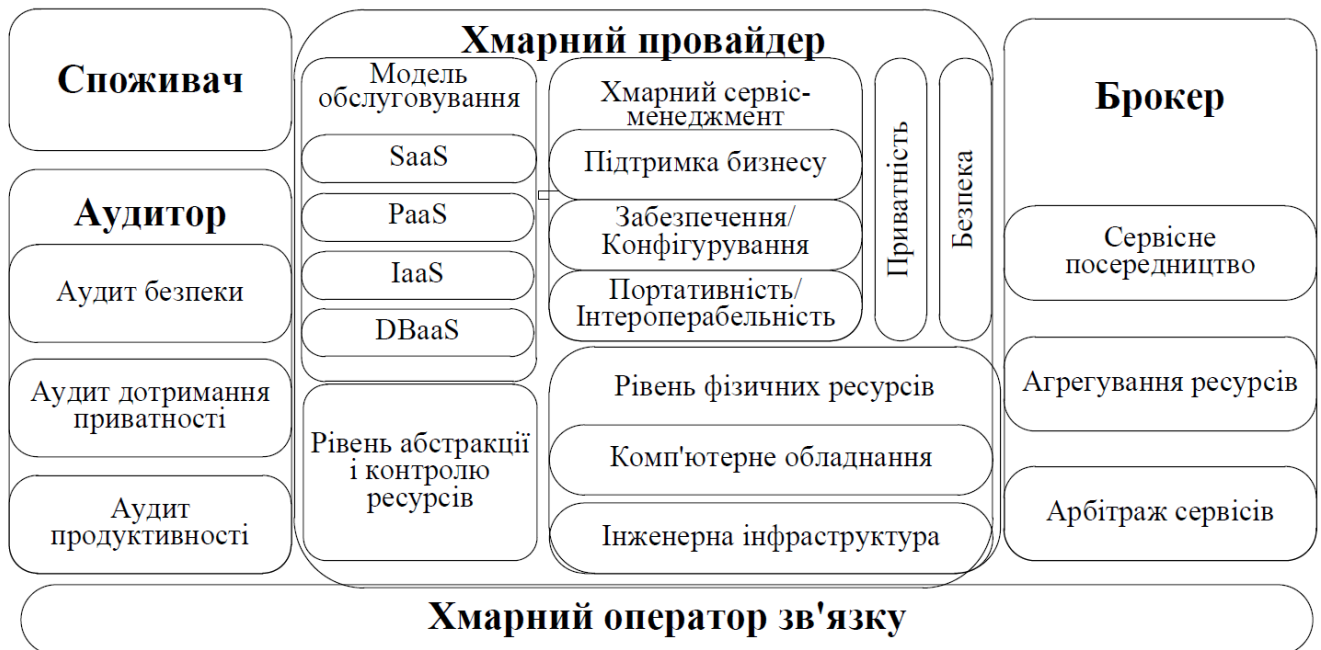


Рис. 5.11. Еталонна архітектура хмарних обчислень Національного інституту стандартів і технологій США

Основними суб'єктами еталонної архітектури є:

- споживачі – особи, як фізичні, так і юридичні, які користуються послугами провайдерів хмарних обчислень;
- провайдери – особи, як фізичні, так і юридичні, які відповідають за надання хмарних обчислень;
- аудитори – фізичні або юридичні особи або організація, які виконують незалежне оцінювання хмарних обчислень;
- брокери – особи, як фізичні, так і юридичні, які є ланкою між провайдером і споживачем хмарних обчислень (можливі варіанти без участі брокера, тобто хмарні послуги поставляються від провайдера до споживача безпосередньо);
- оператори зв'язку – особи, як фізичні, так і юридичні, або компанія, яка надала послуги підключення та доставки хмарних послуг від провайдера до споживача.

Основними моделями обслуговування є:

- IaaS – інфраструктура як послуга.
- PaaS – платформа як послуга.
- SaaS – програмне забезпечення як послуга.

Крім основних моделей обслуговування існують такі моделі:

- HaaS – апаратне забезпечення як послуга.
- SecaaS – безпека як послуга.
- BPaaS – бізнес-процес як послуга.
- DBaaS – база даних як послуга.
- TaaS – довіра як послуга.
- SDPaaS – хмарне середовище розроблення як послуга та інші [225].

Крім того, розрізняють чотири основні моделі розгортання:

- Приватна (Private cloud).
- Модель спільноти (Community cloud).
- Публічна (Public cloud).
- Гібридна (Hybrid cloud).

На основі аналізу розглянутих вище стандартів установлено, що типова хмарна архітектура є клієнт-сервальною з підтримкою технології віртуалізації, побудованою на базі ЦОД, і має ієрархічну структуру. На рис. 5.12 зображено типову мережну архітектуру хмарної системи.

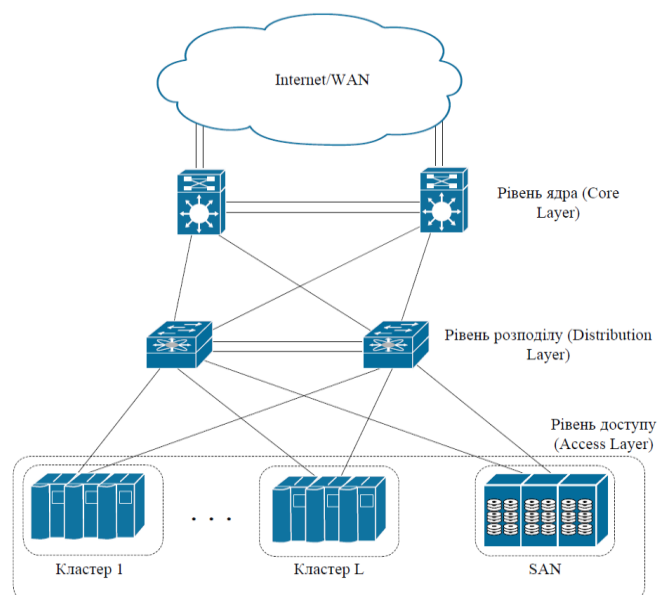


Рис. 5.12 Типова мережева архітектура хмарних систем

Мережева структура складається з трьох основних рівнів:

- Рівень ядра (Core layer). На цьому рівні функціонують маршрутизатори або комутатори третього рівня моделі OSI, які складають основу всієї мережі дата-центру з високошвидкісним и портами (10/40/100 GbE) для маршрутизації потоків між WAN і мережею дата-центру.
- Рівень агрегації або розподілу (Aggregation/Distribution layer). На цьому рівні також працюють комутатори третього рівня моделі OSI, основне призначення яких полягає у розподілі навантаження між локальними мережами дата-центру.
- Рівень доступу (Access layer). На даному рівні розташовуються кінцеві точки (сервера) та мережеве обладнання, що пов'язує кінцеві точки з рівнем агрегації. На рівні доступу функціонують кластери дата-центрів, що складаються з великої кількості фізичних серверів і віртуальних машин, які функціонують на кожному з них. На цьому ж рівні розташовується загальна мережа зберігання даних SAN (Storage Area Network). Група, що складається зі взаємозв'язаних компонентів зберігання даних, обчислювальних і мережних ресурсів, які працюють спільно на рівні доступу з метою надання доступу до сервісів або застосувань клієнтам, називається точкою доставки або POD (Point of delivery).

На сьогодні поширені дві основні топології з'єднання серверів у кластерах: Top of Rack (ToR) і End of Row (EoR) [226]. ToR топологія передбачає розміщення окремого комутатора (або двох для забезпечення надлишковості) на вершині кожної стійки серверів. Комутатори рівня доступу з'єднуються з комутаторами верхнього рівня (рівня агрегації) волоконно-оптичними кабелями (рис. 5.13).

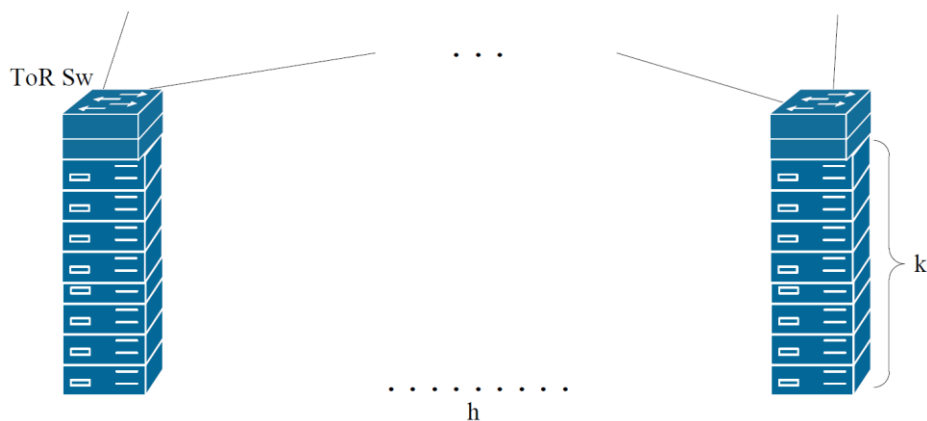


Рис. 5.13 Топологія ToR (Top of Rack)

Позначення на рис. 3: m – загальна кількість серверів у кластері і k – кількість серверів у кожній стійці, з'єднаних з комутаторами рівня доступу двома патчкордами для забезпечення надлишковості. Інша топологія (рис. 5.14) передбачає використання одного (двох для надлишковості) комутатора в кінці (EoR) або в середині (MoR – Middle of Row) масиву стійок з серверами [227].

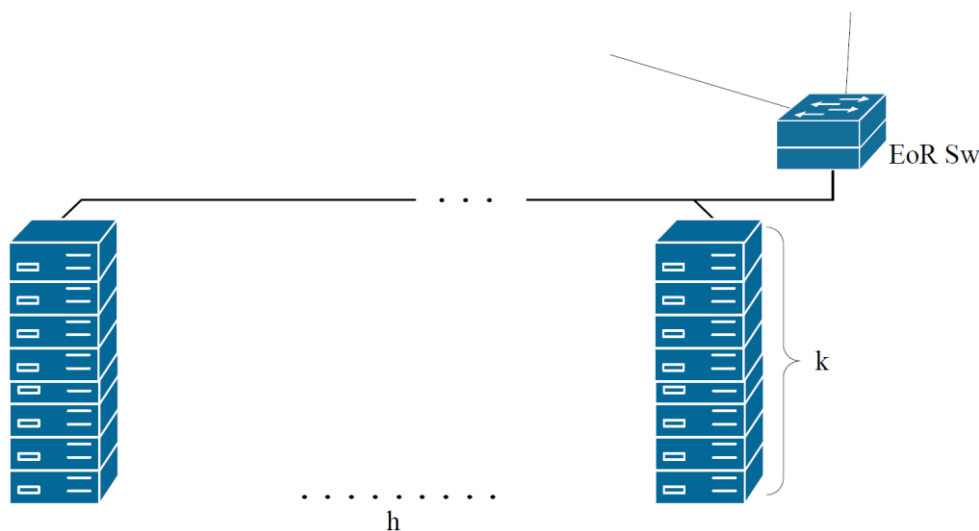


Рис. 5.14. Топологія кластера EoR/MoR (End of Row/Middle of Row)

У такому виконанні мережеве навантаження повністю доводиться на EoR-комутатор.

З точки зору зв'язності мережі архітектури ToR і EoR/MoR являють собою протилежні рішення. В першому випадку зв'язаність мережі максимальна. Але потрібні великі витрати на обладнання і більшу кількість портів на комутаторі рівня агрегації, до якого безпосередньо підключені ToR-комутатори рівня доступу. В другому випадку потрібна менша кількість портів на комутаторі рівня

агрегації і менші витрати на комутатор рівня доступу, але зв'язність мережі при цьому буде мінімальною.

При виборі топології необхідно враховувати накладні витрати на розгортання, кількість обладнання та інші критерії [8]. Тому третій варіант – гібридна архітектура мережі рівня доступу кластера ЦОД між топологіями ToR і EoR. Даний варіант формується на базі комбінації двох концепцій з урахуванням накладних витрат та економії портів. Як результат, варіант забезпечує мінімально прийнятний рівень критичності відмов комутаторів рівня доступу, не знижуючи при цьому показники надійності, порівняно з топологією ToR. Модель такої архітектури показано на рис. 5.15.

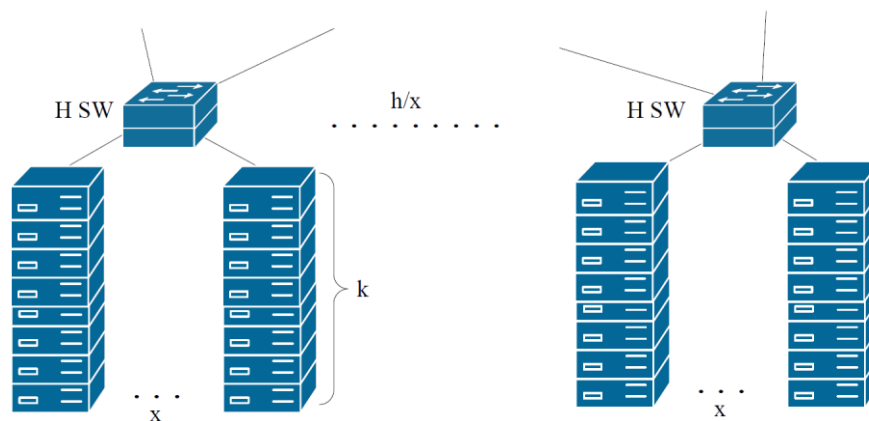


Рис. 5.15. Гібридна топологія кластеру

Гібридна топологія передбачає підключення декількох стійок серверів до окремих комутаторів рівня доступу, які можуть бути розташовані в середині ряду.

На базі представлених топологій кластерів формується архітектура мережі ЦОД. Є два основних варіанти реалізації мережі ЦОД. Перший варіант побудови – архітектуру мережі ЦОД з одним POD [228] подано на рис. 5.16.

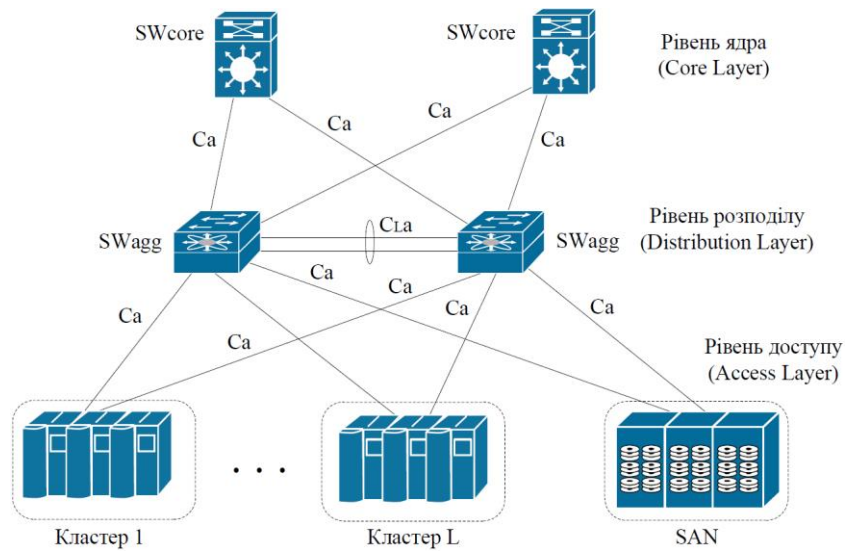


Рис. 5.16. Архітектура мережі центру обробки даних (ЦОД) з одною точкою доставки (POD)

Система є працездатною, якщо складається з мережі зберігання даних SAN, хоча б одного з мережевих маршрутів до кластерів рівня доступу і хоча б одного кластера з серверами.

Другий варіант побудови – архітектура мережі віртуального ЦОД з декількома POD, показаний на рис. 5.17.

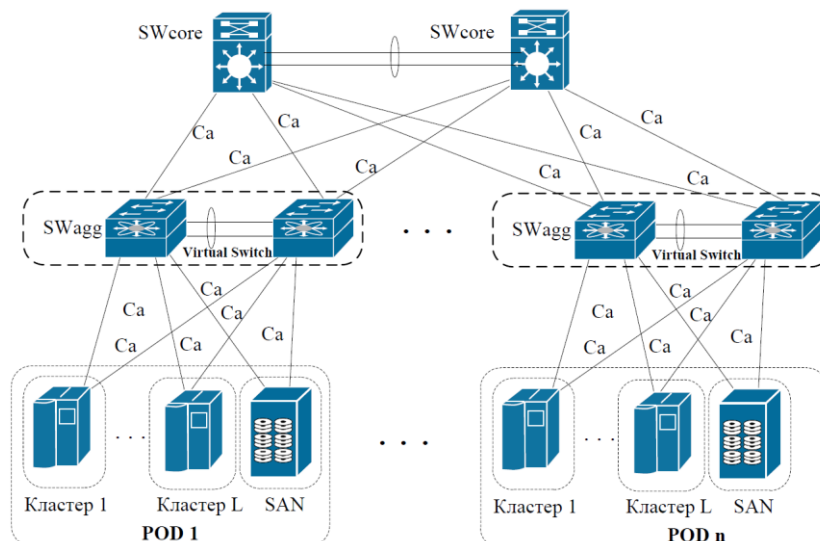


Рис. 5.17. Архітектура центру обробки даних (ЦОД) з декількома точками доставки (POD)

Таким чином, враховуючи вищенаведене, можна виділити наступні типи платформ, які можуть бути використані при побудові ЦОД ЄІС МВС (табл. 5.1).

Табл. 5.1. Типи платформ

Платформа	Тип кластеру	Архітектура мережі ЦОД	Централізація/децентралізація	Приблизна вартість, дол.
1	ToR	1 POD	Ц	200000
2	EoR/MoR	1 POD	Ц	300000
3	Hybrid	1 POD	Ц	150000
4	ToR	n POD	Ц	600000
5	EoR/MoR	n POD	Ц	700000
6	Hybrid	n POD	Ц	550000
7	ToR	n POD	ДЦ	900000
8	EoR/MoR	n POD	ДЦ	1000000
9	Hybrid	n POD	ДЦ	800000

Централізоване рішення передбачає, що всі основні компоненти мережі знаходяться в штаб-квартирі підприємства (окрім кластерів). Децентралізоване рішення – компоненти розподілені між штаб-квартирою та філіями підприємства.

Результати дослідження. Експериментальні дослідження базуються на проблемі вибору архітектури мережі ЦОД для віртуальної критичної інформаційної інфраструктури, що складається з штаб-квартири (саме Міністерство) та 2-х філій (територіальні підрозділи МВС).

Для дослідження використані наступні варіанти можливих рішень (табл. 5.2) [8].

Табл. 5.2. Варіанти конфігурацій

Конфігурація	Штаб-квартира	Філія 1	Філія 2
1	Платформа 1	Платформа 1	Платформа 1
2	Платформа 1	Платформа 1	Платформа 3
3	Платформа 1	Платформа 2	Платформа 3
...	...	...	...
84	Платформа 9	Платформа 9	Платформа 9

По суті, особи, що приймають рішення щодо вибору архітектури для критичної інформаційної інфраструктури, мають вирішити, чи потрібно спроектувати всю ІТ-інфраструктуру на платформі найнижчої вартості (Платформа 3) або знайти варіант, який буде дорожчий, але буде більш надійним,

ефективним тощо. Кожний варіант рішення потрібно оцінити з точки зору майбутніх потенційних проектів та внесення змін до ІТ-інфраструктури, які ці проекти можуть вимагати.

В табл. 5.3 наведені події, які можуть викликати потребу в змінах для платформ.

Табл. 5.3. Події, що можуть викликати потребу в змінах для платформ

Тип події	Приклад	Виграш	Витрати
Нове застосування	Встановлення нового застосування через нові бізнес або ІТ-потреби	Нові фінансові виграши, що надає нове застосування	Витрати на розгортання нового застосування в ІТ-інфраструктуру
Масштабування	Збільшення числа транзакцій через зростання кількості користувачів тощо	Додаткові фінанси від нових користувачів	Витрати на масштабування інфраструктури
Інтеграція	Інтеграція нових застосувань через зміну виробничих процесів тощо	Додаткові фінансові надходження або економія від використання покращених виробничих процесів	Витрати на інтеграцію
Модифікація системи	Потреба в комбінуванні різних типів даних тощо	Виграш від використання нових підходів щодо прийняття рішень	Витрати на модифікацію компонент з метою підтримки нових типів даних тощо
Безпека	Підвищена увага хакерів вимагає покращення фаєрволів, захисту серверних операційних систем тощо	Зменшення потенційного негативного ефекту від атак та збереження даних	Витрати на проведення заходів, спрямованих на покращення безпеки

Критичність	Перехід сервісу в ранг критичних	Підвищена надійність, безпечність тощо	Витрати на проведення заходів щодо переведення сервісу в ранг критичних

Кожний тип події далі поділяється на декілька категорій згідно з їх розміром, поширенням запиту на зміну та супутні витрати/виграши.

В табл. 5.4 наведена структура даних, що використана для дослідження запропонованої моделі.

Табл. 5.4. Структура даних

Штаб-квартира/Філія 1/Філія n				Платформа 1		...	Платформа 84	
Категорія	Подія	Розмір події	Очікувана частота виникнення події, в місяць	Очікувані витрати, k\$	Очікуваний виграш, k\$	...	Очікувані витрати, k\$	Очікуваний виграш, k\$
1	Нове застосування	Незначна	20	10	15	...	12	13
2	Нове застосування	Середня	5	20	30	...	22	44
3	Нове застосування	Велика	1	50	70	...	55	80
4	Масштабування	Незначна	5	5	10	...	3	5
5	Масштабування	Середня	3	15	25	...	10	12
6	Масштабування	Велика	1	30	40	...	40	80
7	Інтеграція	Незначна	15	5	7	...	8	10
8	Інтеграція	Середня	8	10	12	...	15	30
9	Інтеграція	Велика	2	30	45	...	45	70
10	Модифікація системи	Незначна	20	5	6	...	3	5
11	Модифікація системи	Середня	10	10	17	...	12	18

12	Модифікація системи	Велика	1	25	55	...	30	44
13	Безпека	Незначна	100	5	15	...	6	9
14	Безпека	Середня	20	20	100	...	30	70
15	Безпека	Велика	5	50	200	...	60	120
16	Критичність	Незначна	5	5	40	...	2	5
17	Критичність	Середня	2	10	70	...	12	60
18	Критичність	Велика	1	100	800	...	120	700

У цій роботі порівнюється сукупний виграш, отриманий від рекомендацій, що надає запропонована модель, яка розглядає витрати та виграш за критеріями одночасно.

Для моделювання зроблено припущення, що фактична кількість запитів на зміну архітектури для кожної філії за певний період (pr_{zft}) відповідає розподілу Пуасона за параметром λ . Значення λ коливається від 25 % до 200 % базового значення з кроком 5 % (36 точок проектування).

В ході дослідження проаналізовано вплив змін у розподілі запитів на зміну архітектури філій дочірніх компаній на вибір платформи. Базовий розподіл запитів на зміну архітектури встановлений на рівні 30 % для штаб-квартири, 35 % – для філії 1 та 35 % – для філії 2. Для спрощення представлення результатів моделювання фіксує базовий рівень для штаб-квартири на рівні 30 % запитів, а для філій 1 та 2 цей рівень змінюється від 0 % до 70 % з кроком 5 % (14 точок моделювання). Сумарно така схема моделювання дає 504 точки моделювання. Згідно з табл. 5.4 спочатку визначається загальна кількість запитів на зміну архітектури для кожної філії, а далі, визначається розподіл запитів на зміну архітектури по категоріям, враховуючи їх відносну частоту виникнення. Далі, визначаються відповідні витрати та виграш по філіям. На рис. 5.18 показані рекомендації (з 84 можливих конфігурацій, наведених в табл. 5.2), наданих запропонованою моделлю для різних точок симуляції проектування.

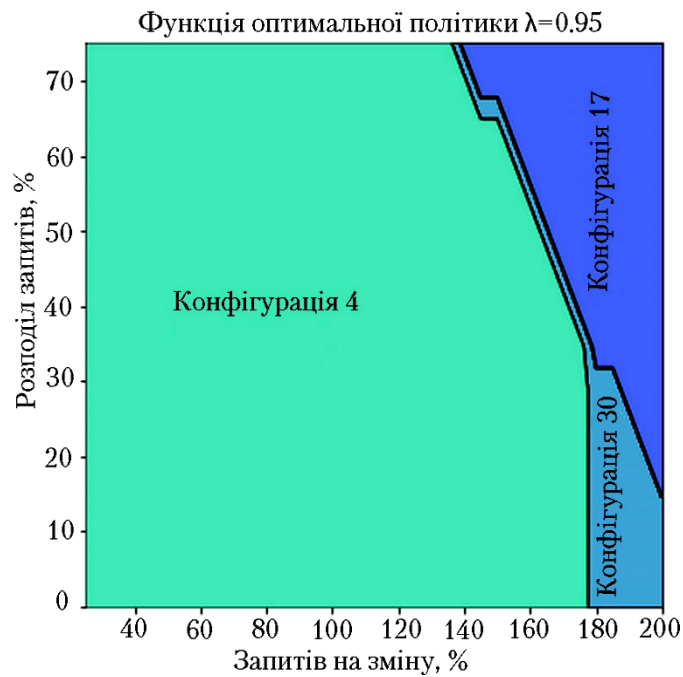


Рис. 5.18. Результати моделювання

Як видно з рис. 5.18, оптимальною з точки зору максимізації сукупного виграшу виявилася конфігурація 4.

Використання даної моделі дозволило швидко оцінити наявні варіанти побудови ЦОД для МВС та визначитись з його конфігурацією.

5.4 Дослідження методу структурної оптимізації нейронних мереж

Дослідження проведені у двох режимах: м'якої та агресивної оптимізації. Вибір режиму залежить від здатності звичайної мережі вирішувати поставлену задачу (75% точності класифікації і більше).

У режимі м'якої оптимізації алгоритм використовує лише додавання та видалення синапсів, кількість шарів та нейронів у них залишається незмінною. Це надає змогу дослідити мережу на наявність надлишкових ваг, спостерігаючи хід навчання однієї і тієї ж мережі (ініціалізованих з однаковими початковими вагами) з та без використання алгоритму. Цей режим використовується для задачі для якої уже відома структура мережі, що її вирішує.

Коли така структура невідома, використовується режим агресивної оптимізації у якому алгоритм використовує усі можливі види мутацій.

5.4.1 Розв'язання MONK's задач

Задачі MONK's [229] були одними із перших, що були використані для порівняння алгоритмів класифікації. Кожен навчальний приклад вибірки містить 7 атрибутів, де останній атрибут – номер класу до якого необхідно віднести приклад:

1. $a_1 \in \{1, 2, 3\}$
2. $a_2 \in \{1, 2, 3\}$
3. $a_3 \in \{1, 2\}$
4. $a_4 \in \{1, 2, 3\}$
5. $a_5 \in \{1, 2, 3, 4\}$
6. $a_6 \in \{1, 2\}$
7. $a_7 \in \{0, 1\}$

Задачі визначаються наступним чином:

- Проблема M_1 : $(a_1 = a_2) \vee (a_5 = 1)$
- Проблема M_2 : хоча б 2 з $\{a_1 = 1, a_2 = 1, a_3 = 1, a_4 = 1, a_5 = 1, a_6 = 1\}$
- Проблема M_3 : $((a_5 = 3) \wedge (a_4 = 1)) \vee ((a_5 \neq 4) \wedge (a_2 \neq 3))$

Нейронні мережі легко вирішують проблеми M_1 та M_2 та досягають 100% точності класифікації на тестовій вибірці. Навчальна вибірка для проблеми M_3 містить шум у вигляді 5% неправильно класифікованих прикладів, саме ця задача буде використана для дослідження.

Використані наступні значення параметрів навчання та параметри структурної оптимізації:

- швидкість навчання: $\eta = 0.001$;
- коефіцієнт інерції: $\mu = 0$;
- коефіцієнт затухання ваг: $\varepsilon = 0.5$;
- максимальна кількість мутацій при схрещенні: $M = 10$;

- кількість епох навчання початкової мережі: $T_0 = 100$;
- кількість епох навчання у ітерації: $T_i = 20$;
- види допустимих мутацій: додавання та видалення ваг.

Отримані залежності значення ціни та точності класифікації наведені на рис. 5.19 та 5.20. Отримана точність класифікації наведена в табл. Табл. .

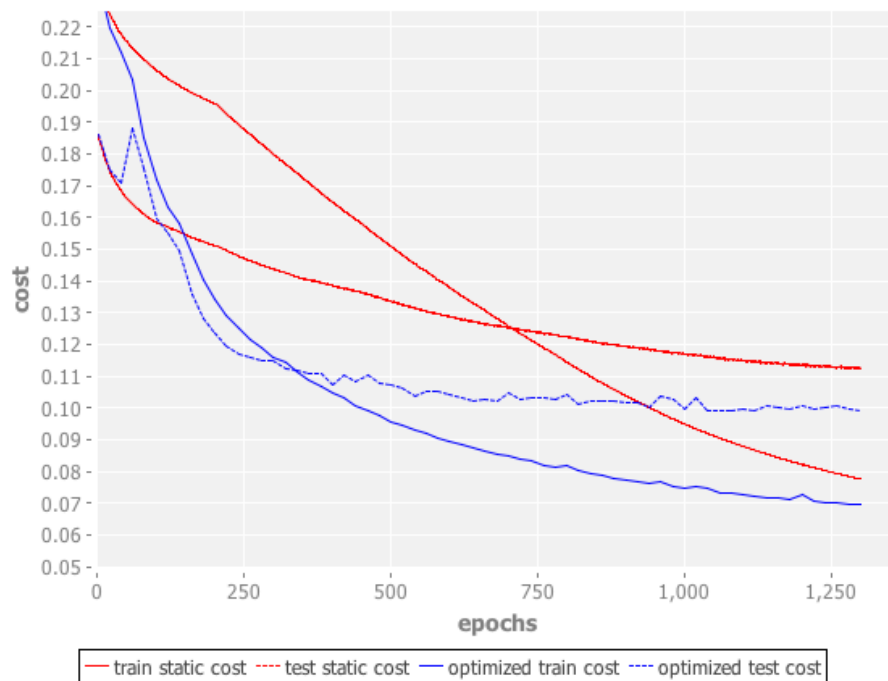


Рис. 5.19. Значення ціни звичайної та оптимізованої мереж

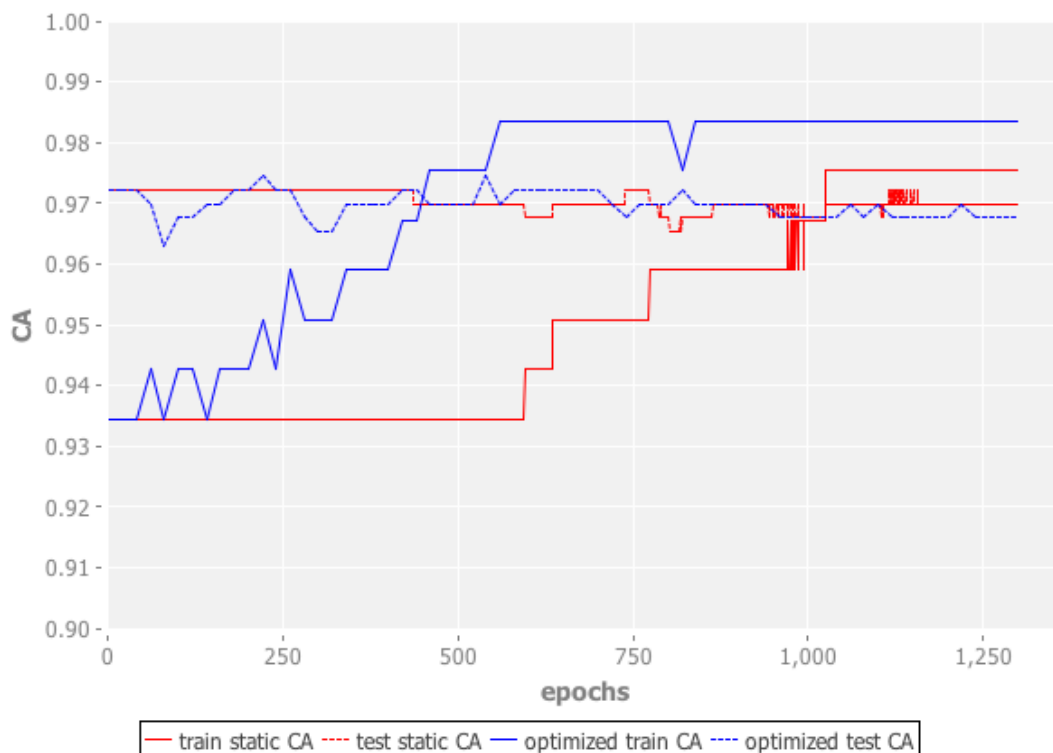


Рис. 5.20. Точність класифікації звичайної та оптимізованої мереж

Табл. 5.5. Результуюча точність класифікації для MONK's проблеми

	Навчання, %	Тестування, %
Звичайна	97.54	96.99
Оптимізована	98.36	96.75

Хоча значного збільшення точності класифікації не відбулося, з наведених залежностей можна зробити висновок, що в результаті оптимізації структури під час навчання мережа не зупинялась у точках локальних мінімумів і навчалась удвічі швидше.

5.4.2 Вирішення задачі TwoSpirals

Ця проблема є досить складною задачею класифікації та узагальнення для багатьох алгоритмів розпізнавання, була запропонована у [230]. Вибірка складається із набору точок, що утворюють дві двохвимірні спіралі. Необхідно правильно класифікувати точки, що не входять до навчальної вибірки.

5.4.2.1 Формування вибірки

Кожен кортеж тренувальної вибірки складається із трьох елементів: координати x та y у діапазоні $0..1$ та номер кривої, якій він відповідає. У графічному вигляді вибірка зображена на рис. 5.21.

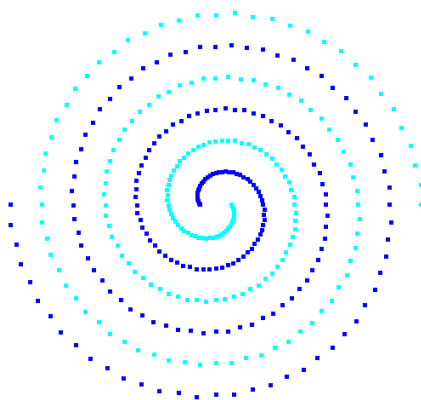


Рис. 5.21. Графічне представлення вибірки TwoSpirals

5.4.2.2 Архітектура мережі

Для вирішення задачі в якості початкової мережі була вибрана 2-шарова мережа з 1 прихованим шаром, що містить 10 нейронів з випрямленою лінійною функцією активації.

Із цією задачею найкраще справляються мережі типу 2-10-10-2, 2-5-10-2, що використовують непарні функції активації (біполярну сигмоїду або гіперболічний тангенс). Натомість, цікавим було дослідити можливість вирішення задачі лише шляхом оптимізації структури, використовуючи більш уживані топології мереж.

5.4.2.3 Дослідження роботи алгоритму

Використані наступні значення параметрів навчання та параметри структурної оптимізації [44]:

- швидкість навчання: $\eta = 0.005$;
- коефіцієнт інерції: $\mu = 0$;
- коефіцієнт затухання ваг: $\varepsilon = 0.1$;
- вірогідність активації нейрону прихованого шару: $p_h = 1$;
- вірогідність активації нейрону вхідного шару: $p_i = 1$.
- максимальна кількість мутацій при схрещенні: $M = 20$;
- кількість епох навчання початкової мережі: $T_0 = 50$;
- кількість епох навчання у ітерації: $T_i = 150$;
- види допустимих мутацій: усі;

На рис. 5.22 та 5.23 зображені залежності значення ціни та точності класифікації від кількості виконаних епох навчання.

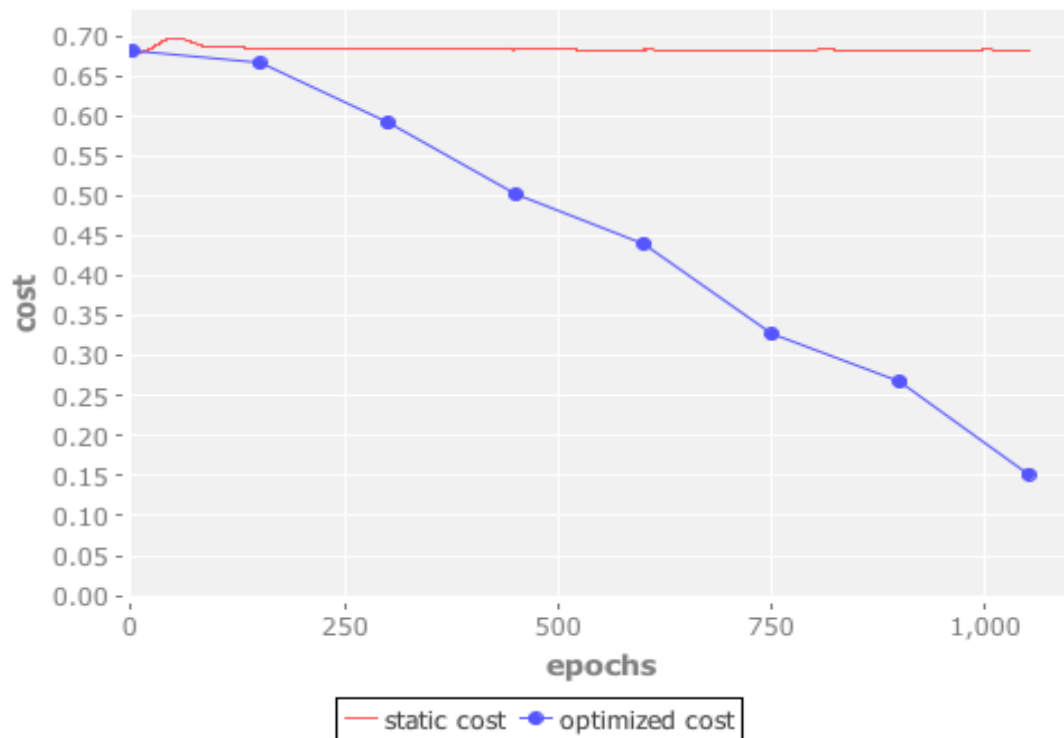


Рис. 5.22. Значення ціни звичайної та оптимізованої мереж під час навчання

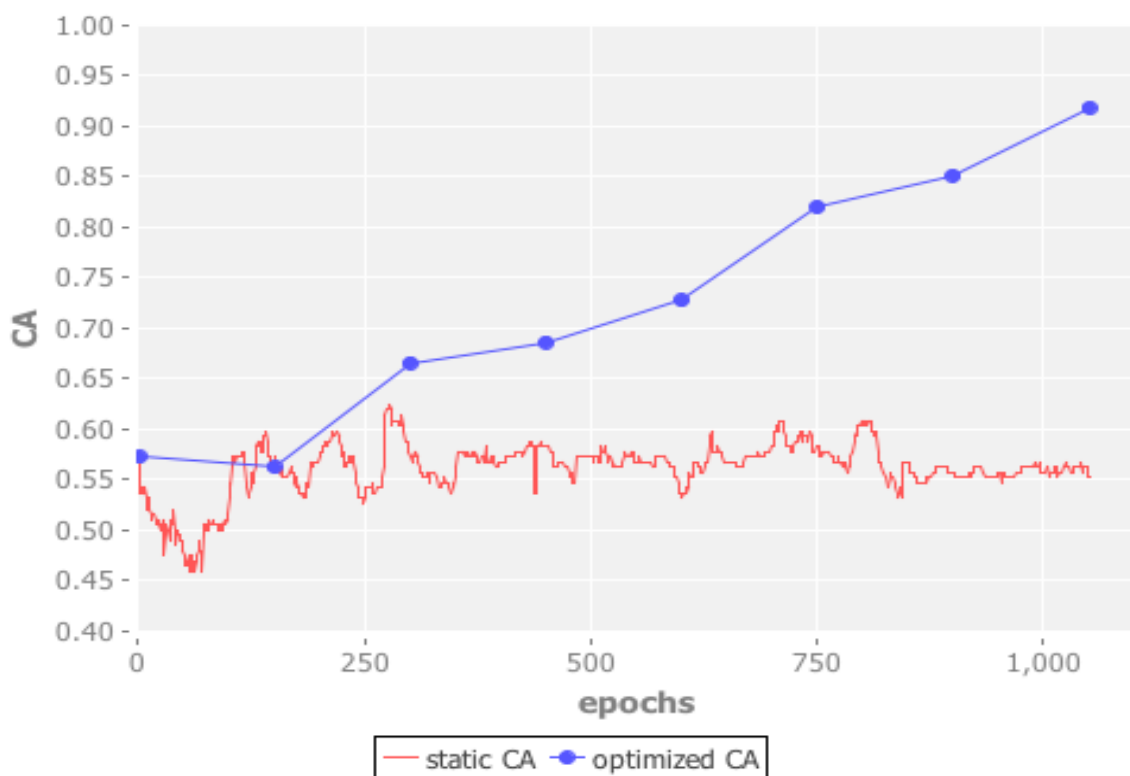


Рис. 5.23. Точність класифікації звичайної та оптимізованої мереж під час навчання

Протягом 7-ми ітерацій алгоритму було отримано [2-9-9-7-7-2] мережу та точність класифікації 92.7%.

5.4.3 Класифікація зображень

Алгоритм використаний для класифікації облич. В якості вихідних даних взята база зображень облич Єльського університету [231].

5.4.3.1 Формування вибірки

Вибрано 10 різних персон та відібрано 50 випадкових зображень кожної персони. Кожне зображення масштабовано до розміру 26 на 26 пікселів та закодовано у вигляді 676-мірного вектору, значення яскравості пікселів нормалізовані до діапазону 0..1. Номер кожної персони закодований вектором розміру 10, що містить 9 нулів та одиницю в позиції номеру персони. Отримані 500 прикладів випадково поділені на навчальну та тестову вибірки у відношенні 2:1.

На рис. 5.24 зображені вихідні зображення та зображення, що використовуються для навчання нейронної мережі.



Рис. 5.24. Приклад формування набору даних

5.4.3.2 Архітектура початкової мережі

Для оцінки роботи алгоритму використана мережа, архітектура якої зображена на рис. 5.25.

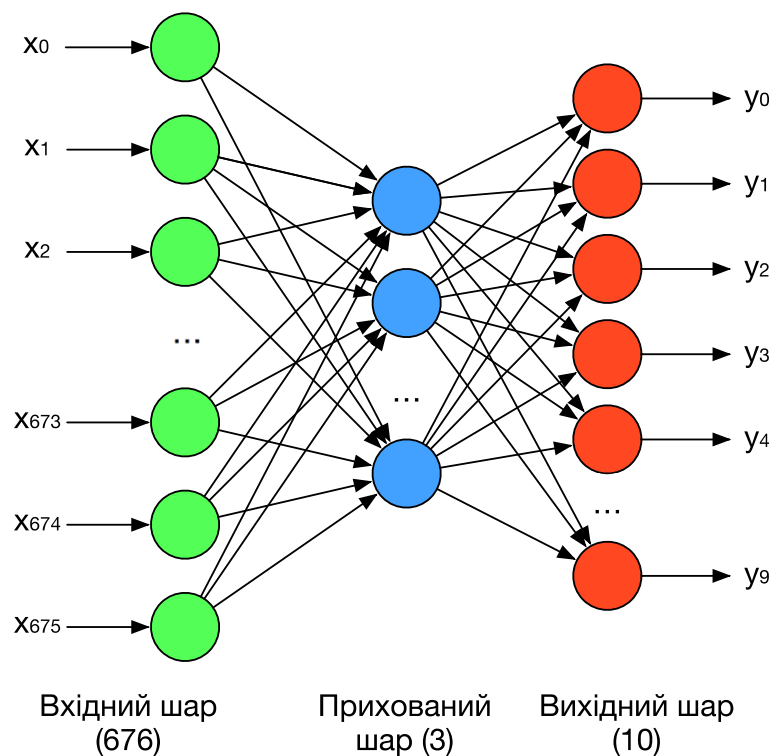


Рис. 5.25. Архітектура мережі для розпізнавання зображень

5.4.3.3 Дослідження роботи алгоритму

Використані наступні значення параметрів навчання [44]:

- швидкість навчання: $\eta = 0.002$;
- коефіцієнт інерції: $\mu = 0.1$;
- коефіцієнт згасання ваг: $\varepsilon = 0.1$;
- вірогідність активації нейрону прихованого шару: $p_h = 1$;
- вірогідність активації нейрону вхідного шару: $p_i = 1$.

Параметри алгоритму наступні [44]:

- початкова кількість нейронів прихованого шару: 3;
- функція активації прихованого шару: ReLU;

- функція активації та функція ціни вихідного шару: softmax;
- максимальна кількість мутацій при схрещенні: $M = 50$;
- кількість епох навчання початкової мережі: $T_0 = 100$;
- кількість епох навчання у ітерації: $T_i = 5$;
- види допустимих мутацій: додавання та видалення синапсів;
- частина навчальної вибірки що використовується для навчання: 1.

Протягом 40 ітерацій алгоритму здійснено 300 видалень та 128 додавань синапсів. На рис. 5.26 та 5.27 зображені залежності значення ціни та точності класифікації від кількості виконаних епох навчання, отримана точність класифікації наведена в табл. 5.6.

Табл. 5.6. Результуюча точність класифікації зображень для $T_i = 5$

	Навчання, %	Тестування, %
Звичайна	97.59	93.41
Оптимізована	98.19	95.80

Завдяки оптимізації структури зв'язків вдалось знизити відсоток помилкової класифікації з 6.6% до 4.2% на тестовій вибірці.

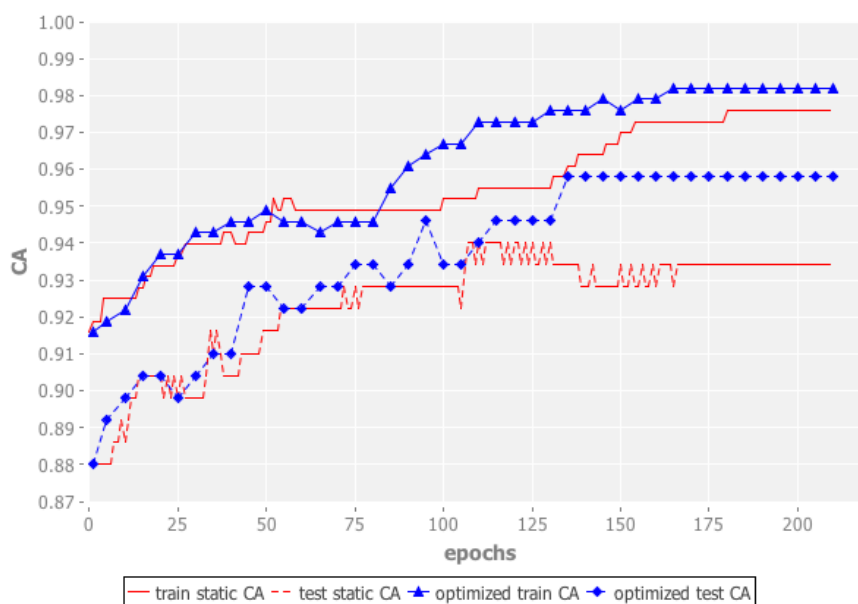


Рис. 5.26. Точність класифікації зображень для $T_i = 5$

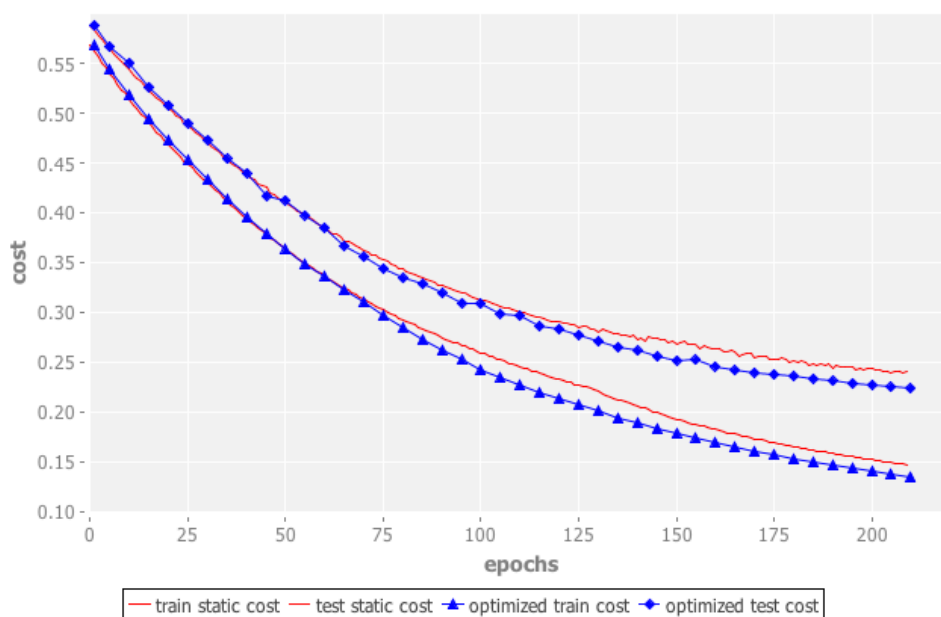


Рис. 5.27. Значення ціни при для класифікації зображень для $T_i = 5$

Також, проведений дослід прийнявши $T_i = 3$, зображено на рис. 5.28 та 5.29. Протягом 100 ітерацій алгоритму здійснено 645 видалень та 457 додавань синапсів. Відсоток помилкової класифікації вдалось знизити з 7.8% до 6.0%. на тестовій вибірці. Результати класифікації наведені в табл. 5.7.

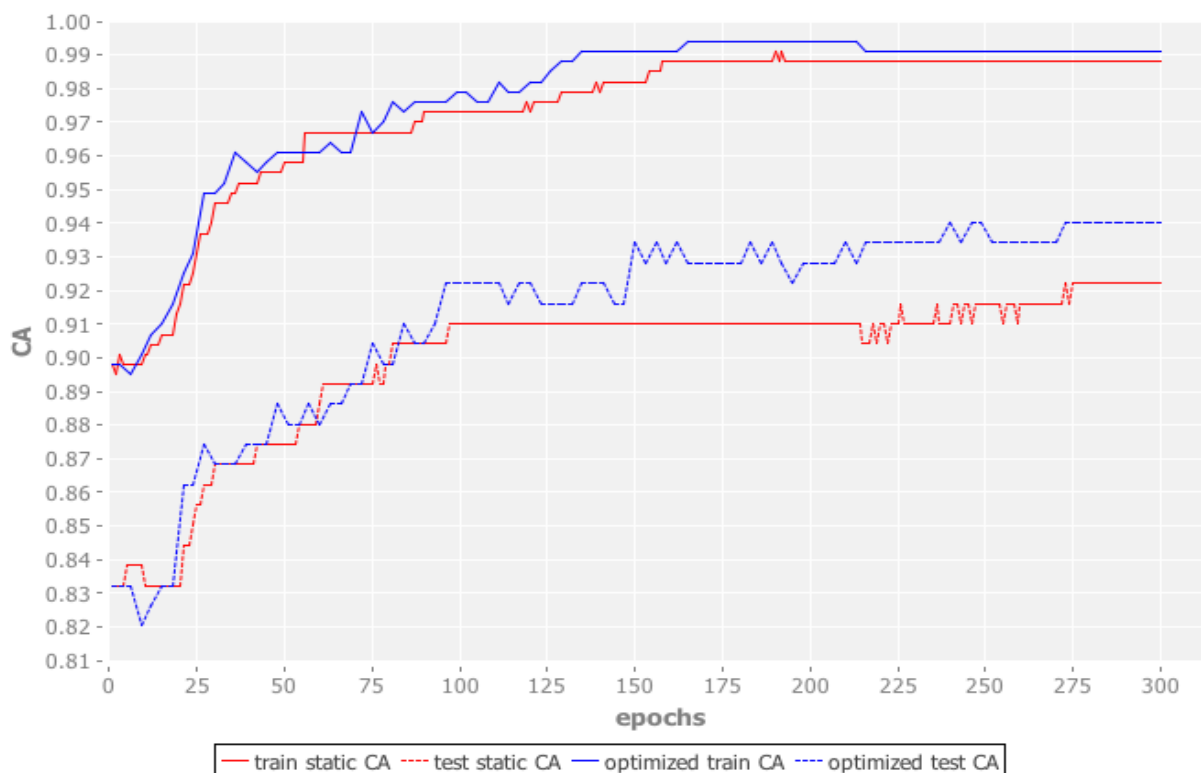


Рис. 5.28. Точність класифікації зображень для $T_i = 3$

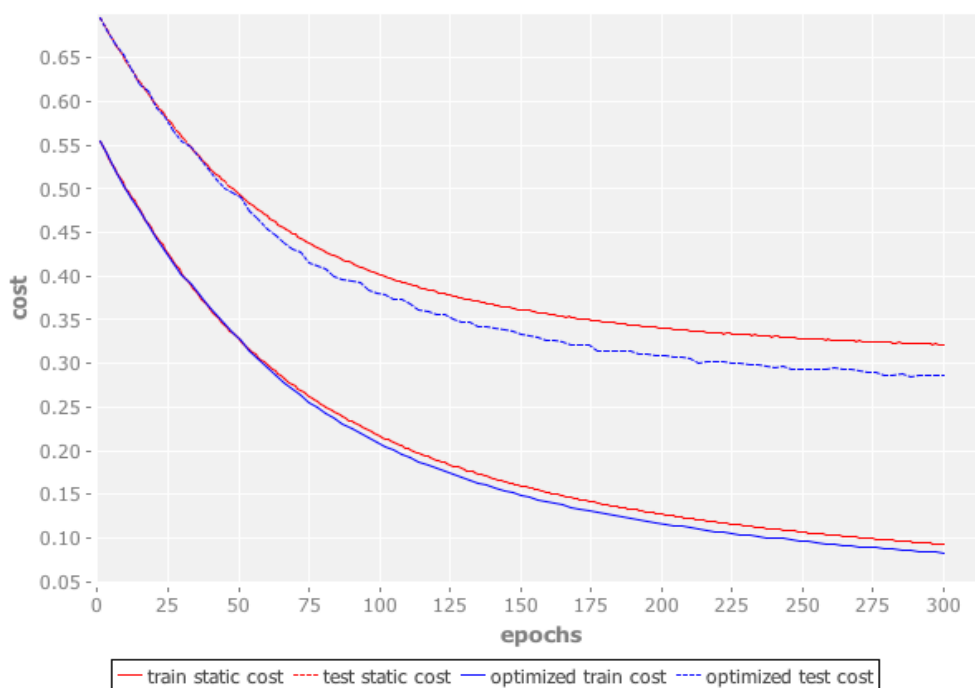


Рис. 5.29. Значення ціни при для класифікації зображень для $T_i = 3$

Табл. 5.7. Результуюча точність класифікації зображень для $T_i = 3$

	Навчання, %	Тестування, %
Звичайна	98.79	92.21
Оптимізована	99.09	94.01

5.5 Дослідження методу розподілу ресурсів критичної інформаційної інфраструктури на основі генетичного алгоритму з чіткими параметрами

Для порівняння вибрані та реалізовані декілька алгоритмів розміщення [22]:

- Random Allocation – випадкове розміщення;
- First Fit Decreasing – перший підходящий за спаданням;
- Demand Risk – алгоритм запропонований у [232];
- CloudSim – алгоритм, що використовується у середовищі CloudSim –

програмному засобі для симуляції роботи фізичних та віртуальних машин у хмарних середовищах.

Для експериментальних досліджень створено різні тестові сценарії для порівняння наведених алгоритмів розміщення.

Кількість поколінь генетичного алгоритму становить 20, розмір популяції кожного покоління становить 100. Розмір підпопуляції під час вибірки становить 5% від загального розміру популяції. Критерієм зупинки являється досягання алгоритмом 20-го покоління. Експериментально було досліджено, що у даному випадку більша кількість поколінь не приводить до значно кращого результату.

Отже, значення параметрів компонентів алгоритму є такими:

- У операції схрещення вірогідність наслідування гену від кожного предку дорівнює 0.5. Це означає, що у нащадка є однаковий шанс унаслідувати ген від обох предків. Як зазначено вище, це значення було підібрано для підвищення різновидності популяції.

- У операції вибірки розмір підпопуляції становить 5, що дорівнює 5% від розміру загальної популяції.

- Загалом, критерієм зупинки генетичного алгоритму являється досягнення алгоритмом ітерації 20. У експериментах ми побачимо, що це значення є достатнім, оскільки застосування більшої кількості поколінь практично не змінює значення функції пристосованості кращого індивіда.

- У генетичному алгоритмі вірогідність мутації дорівнює 0.02. Типові значення параметрів мутації зазвичай не перевищують 0.1, як видно у [233] та [234].

В усіх тестових сценаріях відбувалась оптимізація розміщення за трьома ресурсами: обчислювальна здатність у кількості інструкцій на секунду (CPU ips), об'єм оперативної пам'яті (RAM) та пропускна здатність мережі (Network). Також в усіх тестових сценаріях кількість фізичних машин фіксована та дорівнює 200, в той час як кількість сервісів варіюється між 200 та 1000 з кроком 200.

У першому сценарії параметри встановлено згідно табл. 5.8. Для порівняння алгоритмів визначено, як змінюється ступінь марнування ресурсу CPU зі збільшенням кількості сервісів. Результати дослідження наведені на рис. 5.30.

Табл. 5.8. Розподіл ресурсів сценарію 1

Ресурс	Сервіс	ФМ
CPU	250 Mips	2500 Mips
RAM	0.5 GB	5 GB
Network	0.1 Gbps	1 Gbps

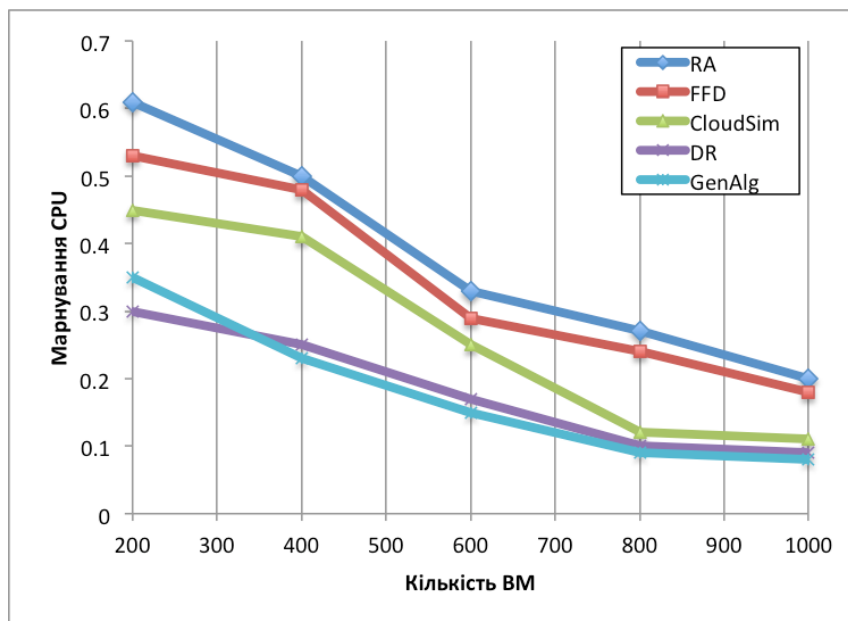


Рис. 5.30. Марнування ресурсу CPU у сценарії 1: алгоритми “Random Allocation”, “First Fit Decreasing”, “Demand Risk”, “CloudSim” та “Genetic Algorithm”

У другому сценарії параметри були вибрані випадковим чином за допомогою рівномірного розподілу, межі значень яких наведені в табл. 5.9 та 5.10. Для порівняння алгоритмів, ми визначили, як змінюється кількість задіяних фізичних машин кожного алгоритму. Результати експерименту наведені на рис. 5.31.

Табл. 5.9. Розподіл ресурсів сервісів, сценарій 2

Ресурс	Межі ресурсу
CPU	250 Mips – 1500 Mips
RAM	0.5 GB – 5 GB
Network	0.5 Gbps – 1.5 Gbps

Табл. 5.10. Розподіл ресурсів ФМ, сценарій 2

Ресурс	Межі ресурсу
CPU	1000 Mips – 3000 Mips
RAM	2 GB – 16 GB
Network	1 Gbps – 5 Gbps

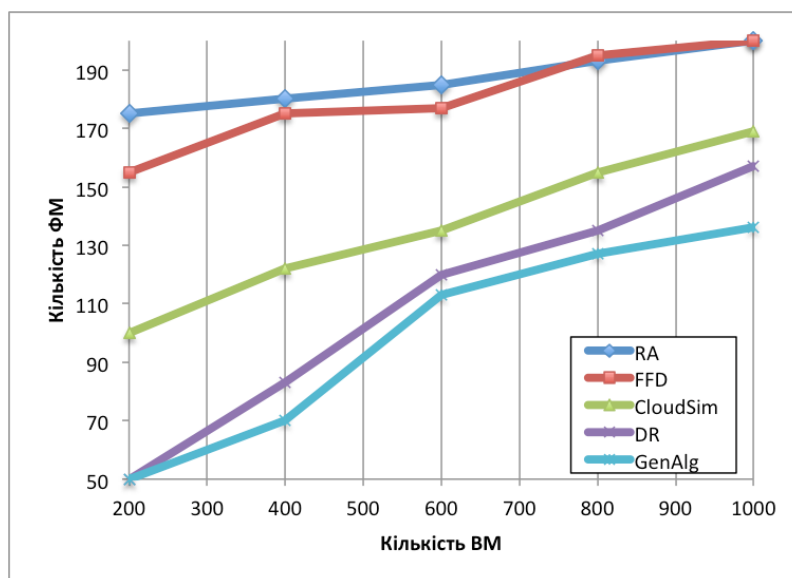


Рис. 5.31. Кількість використаних ФМ у сценарії 2: алгоритми “Random Allocation”, “First Fit Decreasing”, “Demand Risk”, “CloudSim” та “Genetic Algorithm”

У обох тестових сценаріях алгоритми розміщення RA та FFD демонструють найгіршу ефективність розміщення. Вбудований алгоритм CloudSim знаходить дещо кращі розміщення, в той час як алгоритми DR та GenAlg знаходять рішення близькі до оптимальних.

5.6 Дослідження методу розподілу ресурсів критичної інформаційної інфраструктури на базі методу рою часток з чіткими параметрами

На початку роботи алгоритму вектори позиції та швидкостей для усієї популяції обирають випадково. Під час першої ітерації алгоритму найкращі значення позиції кожної частки встановлюють відповідно до її початкової позиції.

Значення параметрів компонентів алгоритму МРЧ [23]:

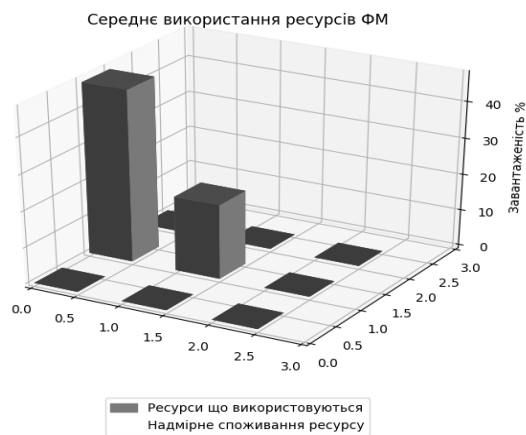
- кількість часток – 100;
- кількість поколінь – 50;
- індивідуальна швидкість навчання, $C_1 = 20$;
- соціальна швидкість навчання, $C_2 = 2,0$;
- коефіцієнт інерції – змінний від 0,9 до 0,4 протягом ітерацій;
- випадкові числа r_1, r_2 - рівномірний розподіл у межах $[0, 1]$;
- вектор позиції – дійсні числа $[0, 100]$;
- вектор швидкості – дійсні числа $[-100, 100]$.

В усіх тестових сценаріях відбувалась оптимізація розміщення за трьома ресурсами: обчислювальна здатність у кількості інструкцій на секунду (CPU ips), об'єм оперативної пам'яті (RAM) та пропускна здатність мережі (Network). В усіх тестових сценаріях кількість фізичних машин фіксована та дорівнює дев'яти, а кількість сервісів змінюється за значеннями X_1, X_5, X_{10} та X_{20} залежно від кількості фізичних машин.

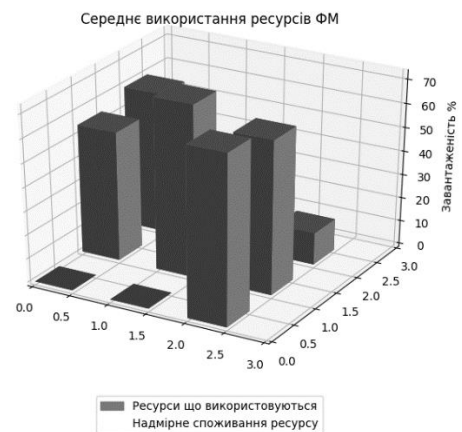
Для візуалізації руху часток МРЧ використано алгоритм зниження розмірності t-SNE [235], результат роботи якого наведено на рис. 5.32. За результатами моделювання можна зробити висновок, що частки у МРЧ справді демонструють просторову організацію «рою» та зберігають її протягом часу. Об'єми використаних ресурсів фізичних машин для кожного значення кількості сервісів наведено на рис. 5.33.



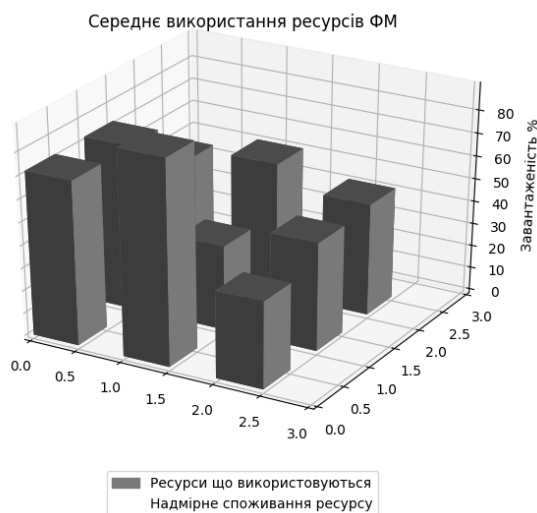
Рис. 5.32. Візуалізація руху часток МРЧ за допомогою алгоритму T-SNE



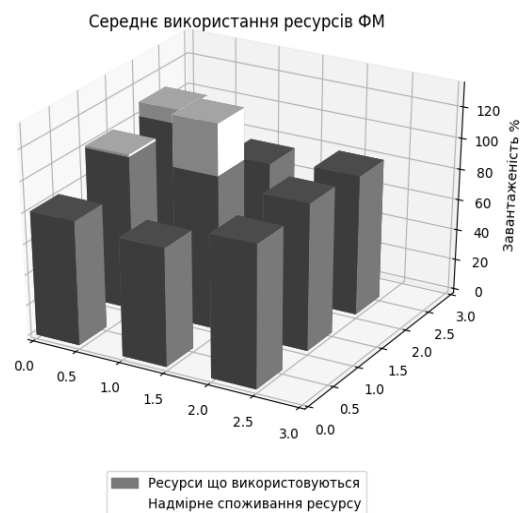
а



б



в



г

Рис. 5.33. Середнє значення використання ресурсів ФМ

Результати моделювання свідчать про ефективність алгоритму. Для розміщення малої кількості сервісів алгоритм дозволяє використати якомога меншу кількість фізичних машин. Сервери, у яких відсутнє навантаження, можна

вимкнути і таким чином зменшити споживання електроенергії у ЦОД. При збільшенні кількості сервісів алгоритм дає змогу розмістити більшу кількість серверів. У випадку, коли споживання ресурсів значно перевищує можливості ЦОД (для x_{20}), розміщення обирається так, щоб мінімізувати сумарне перевантаження на кожній фізичній машині.

5.7 Дослідження методу верифікації даних в інформаційних ресурсах критичної інформаційної інфраструктури (на прикладі ЄІС МВС)

Розглянемо метод верифікації даних, пропонується автором для вирішення проблеми верифікації даних при побудові єдиного сервісу ідентифікації фізичних осіб (ЄСІФО) для ЄІС МВС.

Найчастіше, при створенні нового реєстру, дані про особу збираються з декількох реєстрів. У випадку створення ЄСІФО будуть використані реєстри, перелік яких наведено при постановці проблеми. При цьому пропонується розглядати єдиний державний демографічний реєстр (ЄДДР) як еталонний реєстр [236], відомості до якого і надалі можуть вноситися виключно відповідальними особами, а ідентифікатор ЄСІФО використовуватиметься наскрізно, з можливістю створення умов довіреного внесення інформації іншими державними органами влади (наприклад, представниками Міністерства охорони здоров'я України або ДРАЦС при народженні дитини) шляхом використання Системи електронної взаємодії державних електронних інформаційних ресурсів “Трембіта” (далі – СЕВ ДЕІР) [237] або інших засобів електронної взаємодії.

Персональні дані (ПД) з реєстрів використовуються для ідентифікації та агрегування різних типів даних щодо особи (наприклад, біометричних та генетичних даних, клінічних даних, інформації про рухоме та нерухоме майно). Такі дані зазвичай включають в себе технологічний ідентифікатор (наприклад, ідентифікатор пацієнта, номер соціального страхування або індивідуальний податковий номер), прізвище, ім'я та по батькові (ПІБ), дату народження, місце народження, адресу, поштовий індекс [238]. Однак, обмін ПД може призвести до

розкриття особистої інформації і тому обмежений різними нормативно-правовими актами. Тому дослідження методів обміну ПД направлено на забезпечення цілісності, доступності та конфіденційності даних і є дуже важливим завданням при побудові ЄСІФО.

Існують різні способи захисту ПД особи в реєстрах. Серед них доцільно виділити деідентифікацію [215–217], деперсоніфікацію [242], анонімізацію даних [219, 220], використання обмеженого набору даних [245] і хеш-перетворення [222, 223]. Використання технології наскрізної ідентифікації є одним з найефективніших методів захисту ПД особи.

Створювана технологія наскрізної ідентифікації передбачає трансформування комбінації схем полів з ПД в хеш-коди за допомогою одностороннього алгоритму хешування. Вона може бути використана для ідентифікації особи в реєстрах, без передачі будь-якої частини ПД. Декілька полів з ПД можуть об'єднуватися в різні шаблони з метою полегшення пошуку в різних реєстрах. Технологія наскрізної ідентифікації передбачає, що передача ПД до ЄРІФО відбувається у відкритому або хешованому вигляді, тобто ПД фактично можуть не передаватися до реєстру у відкритому вигляді та зберігаються в реєстрах-донорах. Така ситуація (при використанні варіанту з передачею хешованих даних) буде зберігатися до введення в дію ЄСІФО. З введенням в дію ЄСІФО, хешована інформація в реєстрі трансформується до відкритого стану, ПД в реєстрах-донорах будуть видалені внаслідок нормативної вимоги та на виконання принципу одноразового внесення даних “once only” та буде нормативно узгоджена процедура наповнення ЄСІФО новими даними з реєстрів-донорів державних установ України, які виконують первинну реєстрацію особи (народження дитини, смерть особи, в'їзд іноземного громадянина на територію України).

З метою належного функціонування технології наскрізної ідентифікації та ЄСІФО, ПД мають бути попередньо зібрані та відфільтровані. Як зазначено вище, еталонним реєстром обрано Єдиний державний демографічний реєстр України та, відповідно, унікальний номер запису реєстру (УНЗР) обрано

форматом наскрізного ідентифікатора для інформаційної взаємодії. Для використання в інформаційних ресурсах та інформаційних системах також буде згенерований унікальний технологічний ідентифікатор для кожного верифікованого запису ЄСІФО. Основною функцією використання еталонного реєстру в цій технології є остаточна верифікація особи, тобто якщо дані про певну особу співпадають з даними в ЄДДР, то особа вважається верифікованою і всі персональні дані про особу в усіх реєстрах повинні бути видалені та, відповідно, внесено інформацію, про наскрізний ідентифікатор ЄСІФО (публічна частина – УНЗР, технологічна – згенерований унікальний ідентифікатор), що надасть реєстрам можливість в будь-який момент отримати персональну інформацію в межах компетенції та повноважень.

Запропонований метод [12] використано при створенні сервісу ідентифікації ЄІС МВС (див. розділ 5.1) та в ході виконання державного експериментального проекту верифікації даних про фізичних осіб, що обробляються в деяких національних електронних інформаційних ресурсах [248].

5.7.1 Первинне формування ЄСІФО

Первинне формування записів ЄСІФО відбувається з використанням 20 полів ПД для ідентифікації особи, включаючи 5 обов'язкових полів БД і 15 опціональних полів (див. табл. 5.11). Як правило, вони є унікальними для особи і не змінюються протягом життя. Кожне поле ПД має свою апроксимовану ймовірність ідентифікування двох різних осіб. Це означає, що 2 різні особи можуть бути випадковим чином ідентифіковані в межах популяції, і які мають такі ж значення для цього поля ПД.

Табл. 5.11. Поля ПД

Тип	Ім'я	Значення
	Family (F)	Прізвище

Тип	Ім'я	Значення
Обов'язковий	Name (N)	Ім'я
	Patronymic (PT)	По батькові
	Address (AD)	Місце народження
	DateOfBirth (DOB)	Дата народження
Опціональний	Sex (SX)	Стать
	Country (CNTR)	Країна видачі документа особи
	DocSeries (DS)	Серія документа особи
	DocNumber (DN)	Номер документа особи
	DocDataGet (DDG)	Дата видачі документа особи
	DocExpDate (DED)	Дата закінчення дії документа особи
	DocIssued (DI)	Ким видано документа особи
	IPN	Індивідуальний податковий номер
	SocNum (SN)	Номер соціального страхування
	MotherFN (MFN)	Ім'я матері особи
	MotherLN (MLN)	Прізвище матері особи
	MotherDateOfB (MDOB)	Дата народження матері особи
	FatherFN (FFN)	Ім'я батька особи
	FatherLN (FLN)	Прізвище батька особи
	FatherDateOfB	Дата народження батька особи

Кожне поле ПД програмно нормалізується з метою їх приведення до уніфікованого формату (тільки великі літери і цифри, без пробілів, знаків пунктуації). Всі поля нормалізованої база даних хешуються за допомогою алгоритму SHA-512. Саме цей тип хешування обрано через його поширеність та наявність підтримки у сучасних СКБД. Саме за такою процедурою пропонується знеособити всі вищезазначені державні реєстри з метою їх подальшої передачі

до розробника ЄСІФО та використання для первинного його наповнення у разі застосування хешованих ПД. Наступним кроком передбачається поступове внесення всіх записів зі знеособлених або відкритих реєстрів в ЄСІФО шляхом додаткової обробки кожного запису. Така обробка передбачає створення додаткової таблиці БД, що буде містити записи для кожного суб'єкта реєстру у вигляді додаткових полів з хеш-кодами від шаблонів комбінацій персональних даних, вже попередньо оброблених та представлених у вигляді хеш-кодів (див. табл. 5.12). Фактично, у разі використання знеособлених реєстрів, для зведення записів реєстрів в єдину базу даних пропонується застосовувати подвійне хешування.

Табл. 5.12. Шаблони комбінацій ПД

Хеш-код	Шаблони комбінацій
1	Family + Name + Patronymic + Sex ^o + Country ^o + IPN ^o
2	Family + Name + Patronymic + Address + IPN ^o
3	Family + Name + Patronymic + DateOfBirth + SocNum ^o
4	Family + Name + Patronymic + Address + IPN ^o + SocNum ^o
5	Family + DateOfBirth + DocSeries ^o + DocNumber ^o + DocExpDate ^o
6	Family + Name + Patronymic + Sex ^o + IPN ^o
7	Family + Name + Patronymic + DocSeries ^o + DocNumber ^o + SocNum ^o
8	Family + Name + Patronymic + FatherFN ^o + FatherLN ^o + FatherDateOfB ^o
9	Family + Name + MotherFN ^o + MotherLN ^o + MotherDateOfB ^o
10	Family + Name + Address + MotherFN ^o + MotherLN ^o + IPN ^o

^o – поле , яке НЕ є обов'язковим.

Кожен шаблон комбінації перетворюється в 64-байтовий хеш-код за допомогою одностороннього алгоритму хешування SHA-512. До кожного результуючого коду додається додатковий байт, який вказує кількість відсутніх полів ПД для хеш-коду. Кожна комбінація є достатньою для того, щоб впевнено розрізняти особу. Далі, генерується випадковий унікальний код UID та

тимчасовий УНЗР і зв'язується з хеш-кодами комбінацій. UID і його зв'язані хеш-коди зберігаються на сервері і використовуються для визначення особи.

Правило співставлення хеш-кодів та особи. Кожний хеш-код складається з 64-байтового хеш-значення, яке обчислюється з шаблону комбінації ПД за допомогою одностороннього алгоритму хешування та додаванням 1 додаткового байту, яке вміщує кількість пропущених полів ПД у хеш-коді. Таким чином, будь-яка помилка в полі ПД, що використовуються в комбінації, призведе до помилки співставлення хеш-коду [12].

Пропонується використовувати 3 типи хеш-кодів: ідеальний, хороший і поганий. Для кожного хеш-коду використовуються 2 параметри для визначення його типу: нижній поріг (L) і верхній поріг (U) (див. табл. 5.13). Для ідеального хеш-коду вимагається, щоб кількість пропущених полів ПД дорівнювала або була меншою L. Кількість відсутніх полів ПД для генерації хорошого хеш-коду обмежується інтервалом (L, U). Якщо кількість пропущених полів ПД більше U, то хеш-код буде визначений як поганий. Співпадіння 2 ідеальних хеш-кодів для двох записів БД свідчить про те, що вони ймовірно належать одному і тому ж суб'єкту, а співпадіння між двома хорошими хеш-кодами для двох записів визначають ці хеш-коди ймовірними кандидатами, що належать одному і тому ж суб'єкту.

Табл. 5.13. Пороги відсутніх полів для визначення типу хеш-коду

Параметри	Хеш-код 1	Хеш-код 2	Хеш-код 3	Хеш-код 4	Хеш-код 5	Хеш-код 6	Хеш-код 7	Хеш-код 8	Хеш-код 9	Хеш-код 10
Нижній поріг	1	0	0	1	1	1	1	1	1	1
Верхній поріг	2	1	1	2	2	2	2	2	2	2

При введенні нових записів до реєстру автоматично підраховується кількість ідеальних та хороших співпадінь. Це надасть можливість з'ясувати, чи є вже дана особа у реєстрі. Для цього використовуються 3 параметри для

визначення відповідності особи: поріг для ідеального співпадіння (P), поріг для хорошого співпадіння (G) і поріг для змішаного співпадіння (X). Два записи збігаються один з одним, якщо кількість ідеальних співпадінь більша або рівна P , або кількість хороших співпадінь більша або рівна G , або сума ідеальних і хороших співпадінь більша або рівна X . Для створення ЄРІФО встановлено такі порогові значення $P = 2$, $G = 3$, $X = 3$.

Ймовірні шаблони комбінації ПД для ідеальних або хороших хеш-кодів. Хеш-коди генеруються з комбінацій полів ПД в реєстрі, тому кожен з них може розглядатися як набір перетворених полів ПД. Крім того, у різних хеш-кодах є поля, що перекриваються. Завдяки цьому можливе використання теорії множин для систематичної перевірки сумнівних полів ПД. Доки хеш-код узгоджується, його відповідні поля з ПД можуть бути виключені операціями над множинами з пошуку сумнівних полів ПД. Оскільки дозволено мати відсутні значення опціональних полів ПД, то спочатку необхідно проаналізувати всі ймовірні схеми поєднання полів ПД для ідеальних та хороших хеш-кодів, після чого можна розробити алгоритм перевірки сумнівних ПД.

Особа ідентифікується тільки за допомогою ідеальних або хороших хеш-кодів. Відсутні поля ПД можуть впливати на збіг хеш-коду. Під час реєстрації особи, при розгляді відсутніх полів, можна уникнути деяких неправильних співставлень. Наприклад, хеш-код 8 з табл. 4.3 генерується з комбінації обов'язкових полів Family, Name, Patronymic і опціональних полів FatherFN, FatherLN і FatherDateOfB. Нехай особа зареєстрована вперше, поле FatherDateOfB пропущено, а інші поля введені правильно – в результаті згенеровано хеш-код 8^0 . Далі, коли особа повторно зареєстрована в іншому реєстрі, надані правильні значення всіх вищевказаних полів ПД, включаючи FatherDateOfB, відповідно, згенеровано хеш-код 8^1 . Оскільки поле FatherDateOfB пропущено при генерації хеш-коду 8^0 , то хеш-код 8^1 не відповідатиме хеш-коду 8^0 . Однак, існує ідеальна відповідність між хеш-кодом 8^1 і хеш-кодом 8^0 . Якщо поле FatherDateOfB вважати відсутнім полем для генерування хеш-коду 8^1 , то він буде ідеально зіставлятися з попереднім хеш-кодом 8^0 і таким чином, це надасть

можливість уникнути неправильного порівняння хеш-кодів 8. Отже, всі ідеальні або хороші хеш-коди осіб, які зареєстровані в реєстрах, повинні бути ретельно проаналізовані на рахунок правильного зіставлення.

Кожен хеш-код генерується з різних шаблонів комбінацій полів ПД, які є опціональними або необхідними. На основі розроблених шаблонів комбінацій ПД можуть бути проаналізовані та ідентифіковані всі ймовірні ідеальні або хороші хеш-коди, визначені правила співставлення хеш-кодів (див. табл. 5.14). Наприклад, хеш-код 5 генерується з комбінації з полів Family, DateOfBirth, DocSeries, DocNumber і DocExpDate. З них поля Family і DateOfBirth є обов'язковими полями, а інші 3 поля є опціональними. Відповідно до правил зіставлення хеш-кодів, ідеальний хеш-код 5 може мати 1 відсутнє поле та хороший хеш-код 5 може мати 2 відсутні поля. Тобто ідеальний хеш-код 5 може містити 1 відсутнє поле з DocSeries, DocNumber або DocExpDate, і хороший хеш-код 5 може використовувати тільки одне з цих полів ПД. Отже, є 4 варіанти ймовірних ідеальних і 3 ймовірних хороших хеш-коди 5.

Табл. 5.14. Ймовірні шаблони комбінації ПД для ідеальних або хороших хеш-кодів

Індекс	Хеш-код	Комбінації полів з ПД						Відсутні поля	Тип хеш-коду
1	1	F	N	PT	SX ^o	CNTR ^o	IPN ^o	-	Ід
2		F	N	PT	SX ^o	CNTR ^o	-	IPN ^o	Ід
3		F	N	PT	-	CNTR ^o	IPN ^o	SX ^o	Ід
4		F	N	PT	SX ^o	-	IPN ^o	CNTR ^o	Ід
5		F	N	PT	SX ^o	-	-	IPN ^o CNTR ^o	Хор
6		F	N	PT	-	CNTR ^o	-	IPN ^o SX ^o	Хор
7		F	N	PT	-	-	IPN ^o	CNTR ^o SX ^o	Хор
8	2	F	N	PT	AD	IPN ^o	-	-	Ід
9		F	N	PT	AD	-	-	IPN ^o	Хор
10	3	F	N	PT	DOB	SN ^o	-	-	Ід
11		F	N	PT	DOB	-	-		Хор
12	4	F	N	PT	AD	IPN ^o	SN ^o	-	Ід
13		F	N	PT	AD	-	SN ^o	IPN ^o	Ід

Індекс	Хеш-код	Комбінації полів з ПД						Відсутні поля	Тип хеш-коду
14		F	N	PT	AD	IPN ^o	-	SN ^o	Ід
15		F	N	PT	AD	-	-	IPN ^o SN ^o	Хор
16	5	F	DOB	DS ^o	DN ^o	DED ^o	-	-	Ід
17		F	DOB	-	DN ^o	DED ^o	-	DS ^o	Ід
18		F	DOB	DS ^o	-	DED ^o	-	DN ^o	Ід
19		F	DOB	DS ^o	DN ^o	-	-	DED ^o	Ід
20		F	DOB	DS ^o	-	-	-	DN ^o DED ^o	Хор
21		F	DOB	-	DN ^o	-	-	DS ^o DED ^o	Хор
22		F	DOB	-	-	DED ^o	-	DS ^o DN ^o	Хор
23	6	F	N	PT	SX ^o	IPN ^o	-	-	Ід
24		F	N	PT	-	IPN ^o	-	SX ^o	Ід
25		F	N	PT	SX ^o	-	-	IPN ^o	Ід
26		F	N	PT	-	-	-	SX ^o IPN ^o	Пог
27	7	F	N	PT	DS ^o	DN ^o	SN ^o	-	Ід
28		F	N	PT	DS ^o	DN ^o	-	SN ^o	Ід
29		F	N	PT	-	DN ^o	SN ^o	DS ^o	Ід
30		F	N	PT	DS ^o	-	SN ^o	DN ^o	Ід
31		F	N	PT	DS ^o	-	-	DN ^o SN ^o	Хор
32		F	N	PT	-	DN ^o	-	DS ^o SN ^o	Хор
33		F	N	PT	-	-	SN ^o	DS ^o DN ^o	Хор
34		F	N	PT	-	-	-	DS ^o DN ^o SN ^o	Хор
35	8	F	N	PT	FFN ^o	FLN ^o	FDOB ^o	-	Ід
36		F	N	PT	-	FLN ^o	FDOB ^o	FFN ^o	Ід
37		F	N	PT	FFN ^o	FLN ^o	-	FDOB ^o	Ід
38		F	N	PT	FFN ^o	-	FDOB ^o	FLN ^o	Ід
39		F	N	PT	FFN ^o	-	-	FLN ^o FDOB ^o	Хор
40		F	N	PT	-	FLN ^o	-	FFN ^o FDOB ^o	Хор
41		F	N	PT	-	-	FDOB ^o	FFN ^o FFN ^o	Хор

Індекс	Хеш-код	Комбінації полів з ПД						Відсутні поля	Тип хеш-коду
42	9	F	N	MFN ^o	MLN ^o	MDOB ^o	-	-	Ід
43		F	N	-	MLN ^o	MDOB ^o	-	MFN ^o	Ід
44		F	N	MFN ^o	MLN ^o	-	-	MDOB ^o	Ід
45		F	N	MFN ^o	-	MDOB ^o	-	MLN ^o	Ід
46		F	N	MFN ^o	-	-	-	MLN ^o MDOB ^o	Хор
47		F	N	-	MLN ^o	-	-	MFN ^o MDOB ^o	Хор
48		F	N	-	-	MDOB ^o	-	MFN ^o MLN ^o	Хор
49	10	F	N	AD	MFN ^o	MLN ^o	IPN ^o	-	Ід
50		F	N	AD	MFN ^o	MLN ^o	-	IPN ^o	Ід
51		F	N	AD	-	MLN ^o	IPN ^o	MFN ^o	Ід
52		F	N	AD	MFN ^o	-	IPN ^o	MLN ^o	Ід
53		F	N	AD	MFN ^o	-	-	MLN ^o IPN ^o	Хор
54		F	N	AD	-	MLN ^o	-	MFN ^o IPN ^o	Хор
55		F	N	AD	-	-	IPN ^o	MFN ^o MLN ^o	Хор

Оскільки хеш-код формується з комбінації полів ПД, він пов'язаний з набором полів ПД. Як тільки він збігається з одним з хеш-кодів ідентифікованої особи, відповідний набір полів ПД також є таким, що збігається, і ці поля ПД вважатимуться перевіреними.

5.7.2 Уточнення інформації ПД щодо сумнівних записів

Відповідно до етапу первинного формування ЄСІФО, кожний запис реєстру може знаходитися в одному з чотирьох станів:

д) “verified” – запис верифікований з еталонним реєстром;

е) “approved” – запис визнано верифікованим через зіставлення даних з різних реєстрів, але для запису відсутній аналогічний запис в еталонному реєстрі;

ж) “unapproved” – запис визнано не верифікованим через брак ідентичних даних в різних реєстрах;

з) “unknown” – запис щодо особи, який створено вперше і відсутні інші джерела для порівняння (наприклад, інформація щодо новонародженої дитини).

Після зіставлення реєстрів для кожного запису з статусами “approved” і “verified” генерується унікальний ідентифікатор та сформоване значення тимчасового УНЗР.

Для записів “unapproved” потрібно провести ручну верифікацію шляхом запрошення до органів розпорядників реєстрів відповідних осіб для проходження ідентифікації в режимі опитування.

5.7.3 Розгортання публічного сервісу ідентифікації

На момент закінчення первинного наповнення ЄСІФО та початку етапу уточнення інформації ПД щодо сумнівних записів, шляхом використання прикладних програмних інтерфейсів має вже бути побудована та налаштована інформаційна взаємодія з метою синхронізації внесених змін між реєстрами-наповнювачами та ЄСІФО. Відповідно, така синхронізація повинна бути двосторонньою.

5.7.3.1 Штатне функціонування публічного сервісу ідентифікації

Транзакції на отримання інформації. В штатному режимі ЄСІФО на запити інших інформаційних систем та електронних інформаційних ресурсів щодо персональних даних через прикладний програмний інтерфейс видає останню актуальну версію значень ПД відносно запитуваної особи шляхом використання унікального ідентифікатора. При правильно налаштованій синхронізації такі запити виконуються тільки тими інформаційними реєстрами та електронними інформаційними ресурсами, які не приймали участі в первинному формуванні ЄСІФО і з якими відсутня будь-яка синхронізація.

Фактично, у майбутньому, після видалення базового набору ПД з інших реєстрів з метою реалізації у всіх інформаційних системах та електронних інформаційних ресурсах принципу “once only” та відсутності дублювання інформації, така процедура використовуватиметься усіма інформаційними системами та електронними інформаційними ресурсами.

Транзакції на створення нових записів. Транзакції щодо створення нових записів ЄСІФО виконуються у штатному режимі посадовими особами державних органів, яким надано таке право. Кожне таке внесення повинно контролюватися в автоматичному режимі засобами логічного контролю на коректність введення інформації (тип даних, мова введення) та повинно відбуватися з прив’язкою до деякого набору вивірених довідників (наприклад, довідник поштових адрес). Кожна така успішна транзакція завершується створенням для запису БД унікального ідентифікатора та формуванням для нього тимчасового УНЗР.

Транзакції на внесення змін. Кожна транзакція на внесення в ЄСІФО змін до існуючих записів відбувається згідно з алгоритмом консенсусу.

5.7.3.2 Алгоритм консенсусу

У алгоритмі консенсусу для перевірки транзакцій власники реєстрів у ході голосування вибирають валідаторів транзакцій, які будуть відповідати за певний набір полів ПД. Вага кожного голосу визначається сумою рангів полів ПД, щодо яких вносяться зміни. Завдяки цьому можна досягти високої якості щодо внесеної інформації.

Алгоритм консенсусу залишається децентралізованим в тому сенсі, що всі реєстри беруть участь у виборі набору полів ПД, які перевіряються транзакціями, а, також, він є централізованим через те, що всі рішення приймає невелика група реєстрів-учасників. Реалізація алгоритму підтримує постійний процес голосування і систему тасування, яка може вибирати, у разі збільшення кількості учасників, випадкових валідаторів.

Наприклад, при внесенні змін до поля «Прізвище» будь-яким реєстратором, якому надано право вносити зміни до ЄСІФО, згідно алгоритму консенсусу валідатором буде обрано Міністерство внутрішніх справ (МВС), яке і буде приймати остаточне рішення щодо внесення даних змін до реєстру. А при внесенні змін до полів ПД «Прізвище» та «Місце народження», валідаторами будуть МВС та Міністерство юстиції (МЮ), і саме ці дві структури прийматимуть рішення щодо внесення або невнесення вказаних змін ПД.

Пропонується така схема розподілення полів ПД, їх ваг та валідаторів (див. табл. 5.15).

Табл. 5.15. Поля ПД, ваги, валідатори

Тип	Поле ПД	Вага	Валідатор
Обов'язковий	Прізвище	5	МВС
	Ім'я	5	МВС
	По батькові	5	МВС
	Місце народження	5	МЮ
	Дата народження	5	МЮ
Опціональний	Стать	2	МЮ
	Країна видачі документа особи	2	МВС
	Серія посвідчення особи	2	МВС
	Номер документа особи	2	МВС
	Дата видачі документа особи	2	МВС
	Дата закінчення дії документа особи	2	МВС
	Ким видано документа особи	2	МВС
	Індивідуальний податковий номер	5	МінФін
	Номер соціального страхування	5	МінСоц
	Ім'я матері особи	3	МЮ
	Прізвище матері особи	3	МЮ
	Дата народження матері особи	3	МЮ
	Ім'я батька особи	1	МЮ

Тип	Поле ПД	Вага	Валідатор
	Прізвище батька особи	1	МЮ
	Дата народження батька особи	1	МЮ

5.7.4 Результати використання методу верифікації

За допомогою запропонованого методу верифікації отримано зведений з 4-х державних реєстрів масив даних, який містить 98% верифікованих записів. 2% записів містять набір даних, який є досить неповним, а тому для його подальшого використання необхідно провести верифікацію даних з залученням осіб, яким ці персональні дані належать.

5.8 Висновки до п'ятого розділу

У даному розділі представлений детальний опис використаного в роботі прикладу критичної інформаційної інфраструктури – ЄІС МВС, наведена її структура та функціональні можливості. Далі наведене дослідження моделі оцінки впливу забезпечуючих систем на архітектуру критичної інформаційної інфраструктури. Дана модель використана при проектуванні ЦОД ЄІС МВС, що дозволило врахувати взаємозалежність систем охолодження та електроживлення на всю роботу ЦОД, і таким чином, якісно продумати резервування відповідних систем. На прикладі вибору структури ЦОД ЄІС МВС досліджено метод оцінки та вибору оптимальної конфігурації компонент критичної інформаційної інфраструктури за допомогою Марківського процесу прийняття рішень. Наведені дослідження методу структурної оптимізації нейронних мереж для різних класів задач, доведена його працездатність та надійність. І насамкінець, досліджено два методи розподілу ресурсів критичної інформаційної інфраструктури. Перший метод побудовано з використанням методу структурної оптимізації нейронних мереж, другий – з використанням методу роя часток.

Обидва методи дають можливість оптимально розподіляти ресурси критичної інформаційної інфраструктури в ЦОД.

РОЗДІЛ 6

ЄДИНА СИСТЕМА ПРОЕКТУВАННЯ КРИТИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ

6.1 Концепція проектування критичної інформаційної інфраструктури

Пропонована концепція проектування КІІ орієнтована на створення єдиного універсального середовища проектування ІТ-інфраструктури підприємств з критичною інфраструктурою і представляє собою систему шляхів, методів вирішення проблеми підвищення ефективності процесу проектування КІІ, що об'єднує наступні принципи та положення:

1. Чотирирівнева узагальнена структура КІІ.
2. Життєвий цикл КІІ.
3. Загальна схема пошуку проектних рішень щодо архітектури на базі компенсаційно-декомпенсаційного підходу.
4. Виокремлення чотирьох точок зору на архітектуру.
5. Критерій оптимальності проектування КІІ.
6. Виокремлення чотирьох універсальних процесів при проектування КІІ.
7. Використання розширених UML-діаграм для представлення процесу міркувань щодо архітектури.
8. Використання розмічених транзиційних систем та темпоральних логік для формалізації моделей архітектурних проектів та їх верифікації.
9. Використання методів штучного інтелекту для побудови моделей оцінювання та обґрунтування архітектурних проектів.
10. Використання методів штучного інтелекту для побудови моделей вдосконалення критичної інформаційної інфраструктури.
11. Представлення єдиної системи проектування КІІ (ЄСПК) у вигляді замкнутої системи.
12. Інтегроване проектування КІІ.

13. Врахування вимог бізнес-процесів.

Більшість вищезазначених принципів не є новими, але в запропонованій концепції проектування КІІ вони вперше застосовані в сукупності. Крім того, тут не тільки систематизовані відомі методи і технології з подальшою інтеграцією їх в ЄСПК, але і пропонуються оригінальні методи і підходи. Реалізація запропонованої концепції відбувається за допомогою ЄСПК, яка забезпечує централізоване проектування КІІ.

Крім систематизації відомих та пропонуваніх технологій, однією з основних задач концепції є трансформація процесу мислення архітекторів ЄСПК - відмова від фрагментарного підходу, орієнтованого на проектування окремих елементів та систем КІІ та прийняття системного бачення, що розглядає ЄСПК як цілісну систему, підпорядковану цілям підприємства з критичною інфраструктурою.

1. Чотирирівнева архітектура КІІ. За основу береться запропонована в [3] чотирирівнева архітектура КІІ, коли виокремлюються чотири взаємопов'язаних ієрархічних рівня (бізнес-застосувань, універсальних сервісів, обчислювальних ресурсів і мережевої взаємодії) та декілька підрівнів критичних сервісів та елементів (див. розд. 6.1.1). Виокремлення рівнів здійснено, виходячи із спільності виконуваних функцій і сервісів, що надаються кожним з рівнів.

2. Життєвий цикл КІІ. Створення КІІ представляє собою багатоетапний ітеративний процес. Життєвий цикл починається в момент появи ідеї створення нової критичної інформаційної інфраструктури [4] і складається з декількох етапів (див. розд. 6.1.2).

3. Загальна схема пошуку проектних рішень щодо архітектури. Для побудови загальної схеми пошуку проектних рішень при проектуванні архітектури критичної інформаційної інфраструктури з врахуванням недоліків і переваг методів, підходів та ФОАП, що розглянуто в першому розділі, потрібно також розглянути існуючі стратегії прийняття рішень та чинники, які впливають на вибір стратегії. Пропонована схема на базі КДК-підходу (див. розд. 6.1.3) дає можливість архітектору провести пошук проектних рішень в цілому або відносно

якоїсь точки зору на архітектуру, використовуючи загальнозживані стратегії прийняття та обґрунтування проектних рішень та відповідно до поставлених вимог, цілей та критеріїв.

4. Виокремлення чотирьох точок зору на архітектуру. Основними точками зору на архітектуру, які використовуються в даній концепції наступні:

- бізнес-орієнтована – дозволяє оцінювати проектні рішення з точки зору бізнес-вимог;
- орієнтована на дані – дозволяє оцінювати проектні рішення відповідно до вимог, які накладаються на дані, що циркулюють в КІІ;
- орієнтована на технології – дозволяє оцінювати проектні рішення відповідно до використовуваних в проекті технологій та технологічних компонент;
- орієнтовані на критичність – дозволяє оцінювати проектні рішення відповідно до вимог критичності окремих елементів, компонент, систем або ресурсів.

Використовувані точки зору дозволяють архітектору зосереджуватись на окремих аспектах при проектуванні КІІ. Такий підхід дозволяє відшукувати найоптимальніші проектні рішення, які задовольняють поставлені вимоги та цілі проектування.

5. Критерій оптимальності проектування КІІ. Для формування критерію оптимальності проектування і подальшого функціонування критичної інформаційної інфраструктури пропонується використати наступну множину параметрів: доступність, надійність, живучість, забезпеченість, відновлюваність, економічність, безпечність, адаптивність, строк життя, ефективність (див. розд. 2.4).

При формуванні вимог критерію проектування критичної інформаційної інфраструктури необхідно також враховувати особливості розв'язуваних нею задач. Слід зазначити, що визначення всієї множини параметрів не можна повністю звести до системи формалізованих процедур, бо деякі з них вимагають якісного аналізу. Для такого аналізу слід використати метод структуризації, який

дозволяє поділити задачу на підзадачі, визначитись за допомогою експертів або без них з методами розв'язання цих підзадач, обмеженнями використання цих розв'язків та методами поєднання розв'язків.

6. Виокремлення чотирьох універсальних процесів при проектування КІІ. Основними процесами при проектування КІІ є: представлення процесу міркувань щодо архітектури (див. розд. 2.2), формалізація та верифікація моделей проектних рішень (див. розд. 2.3), обґрунтування вибору проектного рішення (див. розд. 3), відстежування змін та їх впливу на архітектурне рішення (даний процес не розглядається в роботі).

7. Використання розширених UML-діаграм для представлення процесу міркувань щодо архітектури. Розширення основних конструктів UML таких як пакет, клас, об'єкт та артефакт надають архітектору можливість пристосувати потужний інструментарій UML для процесу фіксації міркувань щодо архітектури КІІ, її поступової еволюції, а також зберігати інформацію про альтернативні проектні рішення, які виникають в процесі проектування (див. розд. 2.2).

8. Використання розмічених транзиційних систем та темпоральних логік для формалізації моделей архітектурних проектів та їх верифікації. Пристосування математичного апарату розмічених транзиційних систем для представлення UML-моделей архітектурних рішень у формалізованому вигляді та темпоральних логік для їх подальшої верифікації надає архітектору можливість застосувати широкий вибір існуючих методів та підходів верифікації параметризованих моделей до створених моделей проектних рішень та знайти в них логічні помилки, які в подальшому б могли призвести до неможливості побудови КІІ або окремих її частин (див. розд. 2.3).

9. Використання методів штучного інтелекту для побудови моделей оцінювання та обґрунтування архітектурних проектів критичної інформаційної інфраструктури. Задача оцінювання та обґрунтування архітектурних проектів є зазвичай багатокритеріальною задачею. Одним з найпоширеніших підходів для розв'язання таких задач є використання методів

штучного інтелекту через їх просту реалізацію у вигляді комп'ютерних моделей (див. розд. 3).

10. Використання методів штучного інтелекту для побудови моделей вдосконалення критичної інформаційної інфраструктури. Одним з найпоширеніших підходів для розв'язання таких задач є використання методів штучного інтелекту через їх просту реалізацію у вигляді комп'ютерних моделей (див. розд. 4).

11. Представлення єдиної системи проектування КІІ (ЄСПК) у вигляді замкнутої системи. Замкнені системи проектування КІІ зі зворотним зв'язку не тільки підвищують оперативність і якість проектування, а й дозволяють мінімізувати вплив людського фактора (див. розд. 6).

12. Інтегроване проектування КІІ. В цей час створення системи проектування ІТ, як правило, здійснюється шляхом безсистемної інтеграції розрізнених застосунків проектування, не об'єднаних спільними цілями і завданнями. При цьому кожне ПЗ проектування ІТ використовується за своїм функціональним призначенням, а їх спільну роботу важко підпорядкувати головній меті – підвищенню ефективності процесу проектування.

Перспективним буде такий підхід до побудови ЄСПК, коли всі підсистеми, використовувані в процесі проектування об'єднуються в єдину систему з загальними цілями і завданнями. В цьому випадку ЄСПК дозволяє спроектувати не тільки окремі елементи, застосунки, а надає можливість інтегрованого проектування всіх компонентів та КІІ в цілому, виходячи з єдиних цілей. Під інтегральним проектуванням КІІ розуміється загальне керівництво різноманітними інформаційними технологіями та засобами проектування, виходячи з єдиних цілей, визначених значимістю бізнес-процесів та відповідно до цілей та завдань підприємства з критичною інформаційною інфраструктурою (див. розд. 6).

13. Врахування вимог бізнес-процесів. Автоматизацію проектування КІІ доцільно починати з автоматизації окремих функцій проектування. На цьому етапі формується стек автоматизованих функцій. Наступна фаза – інтеграція

механізмів проектування, яка вимагає системного підходу. Створення ефективних технологій бізнес-аналізу з проекцією на процес проектування КІІ можна розглядати, як організацію своєрідного глобального зворотного зв'язку. Останньою стадією стає організація інтегрованого проектування КІІ. На цій стадії найважливіша роль відводиться методам оцінювання проектних рішень (МОПР). Їх мета – аналізувати проектні рішення з метою оптимізації продуктивності відповідно до встановлених критеріїв. МОПР забезпечується зворотний зв'язок, коли в результаті аналізу критеріїв, а також метрик і параметрів проектних рішень, архітектор ЄСПК корегує саме проектне рішення. Для цього МОПР здійснюють: розрахунок критеріїв для кожного проектного рішення щодо архітектури; відстеження впливу змін конкретних елементів на всю архітектуру; пошук альтернативних рішень.

Одне з основних завдань, що розв'язуються МОПР, полягає у координації цілей і завдань КІІ із засобами проектування на всіх ієрархічних рівнях системи проектування КІІ. Інструментарій МОПР дає можливість архітектору ставити стратегічні і тактичні цілі, після чого, вимірюючи значення критеріїв, що характеризують якість процесу проектування, вносити відповідні зміни в архітектурне рішення. При зміні бізнес-цілей або значущості бізнес-процесів ЄСПК повинна оперативно реагувати на ці зміни, наприклад, шляхом пошуку альтернативних проектних рішень (див. розд. 6).

6.1.1 Узагальнена структура критичної інформаційної інфраструктури підприємства

ІТ-інфраструктура будь-якого підприємства або організації становить собою комплекс взаємопов'язаних структур, систем, об'єктів тощо, який забезпечує належне функціонування ІТ-системи. Конкретна схема ІТ-інфраструктури визначається розміром організації, характером вирішуваних бізнес-задач, переліком ІТ, що використовуються тощо [91].

Підприємства, які обмежуються використанням в локальній мережі декількох встановлених на сервер бізнес-застосувань, що дозволяють дещо підвищити продуктивність праці співробітників за рахунок автоматизації ряду бізнес-процесів, можуть нести значні збитки через, наприклад, втрату даних внаслідок відсутності системи резервного копіювання. ІТ-інфраструктура, в якій відсутні тільки деякі важливі складові, буде більш дешевою, але не обов'язково ефективною. Максимальний ефект від функціонування ІТ-інфраструктури може бути досягнутий тільки за наявності комплексної повноцінної ІТ-інфраструктури, що дозволяє раціонально виконувати процеси діяльності та ефективно вирішувати бізнес-завдання. Водночас організація такої ІТ-інфраструктури – це складний і тривалий процес, що вимагає від підприємства серйозних капіталовкладень на створення ІТ-інфраструктури і значних поточних витрат на її підтримку.

При розгляді питань, пов'язаних з управлінням ІТ-інфраструктурою, виділяється різна кількість ієрархічних рівнів в залежності від специфіки ведення бізнесу і характеру розв'язуваних системою управління інфраструктурою (СУІ) завдань. Так, у праці [249] при розгляді метрик операторів телекомунікаційних сервісів (ОТС) виділяється п'ять ієрархічних рівнів: телекомунікаційної технології; мережі; сервісів; абонентського; телекомунікаційного бізнес. Відповідно до моделі OSI [250], концепція системи управління мережами (TMN) розглядає мережеву логічну архітектуру, що включає в себе п'ять рівнів управління [251] - це рівень мережевих елементів, рівні управління елементами, мережею, послугами та рівень бізнес-управління. У праці [252] концепція управління ІТ-інфраструктурою розглядається стосовно ієрархічної структури, яка містить три шари: нижній – управління мережами, середній - управління системами, верхній - управління сервісом ІТ.

Доцільно розглядати ІТ-інфраструктуру підприємства з критичною інфраструктурою у вигляді узагальненої ієрархічної схеми, яка представлена на рис. 6.1.

В критичній інформаційній інфраструктурі – організаційно-технічній сукупності ІТ-системи та ТКМ [3][110] – пропонується виділити чотири ієрархічних рівня з двома або трьома підрівнями в кожному: рівень бізнес-застосувань (підрівні критичних та інших бізнес-застосувань), рівень універсальних сервісів (підрівні захисту сервісів, критичних та інших сервісів), рівень обчислювальних ресурсів (підрівні критичних та інших ресурсів) та рівень мережевої взаємодії (підрівні захисту об'єктів критичної інформаційної інфраструктури, критичних та інших об'єктів КІІ).

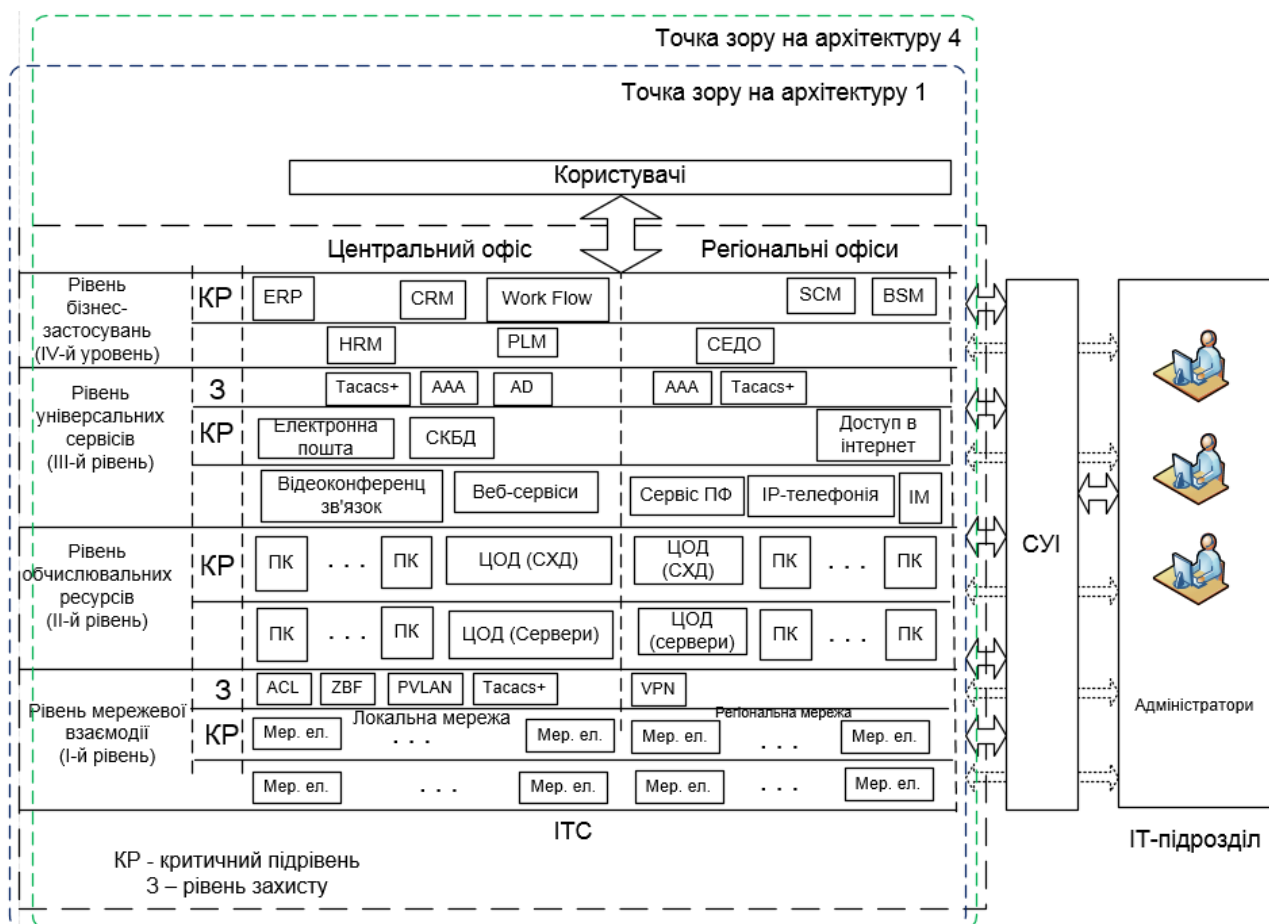


Рис. 6.1. Узагальнена структура критичної інформаційної інфраструктури

На IV-му рівні виконуються розподілені застосування, які мають безпосереднє відношення до автоматизації бізнес-процесів або процесів діяльності. На цьому рівні працюють системи: планування ресурсів підприємства (ERP), управління персоналом (HRM), управління взаємодією з клієнтами (CRM), управління інформацією про вироби (PLM), управління

робочими процесами (Work Flow), документообігом (СЕДО), BSM та багато інших ІТ, які є необхідними для успішного ведення бізнесу підприємства. Деякі з цих процесів є критичними (ERP, Work Flow та ін.), їхня зупинка може привести до серйозних фінансових втрат або техногенної катастрофи. Також, на цьому рівні розташовані програмні засоби, які визначають значимість бізнес-процесів для підприємства, їхню критичність, а також компоненти СУІ, які відповідають за розподіл обчислювальних та комунікаційних ресурсів.

На III-му рівні розташовані сервіси, що не залежать від специфіки бізнесу підприємства. До таких сервісів належать: електронна пошта, СКБД, відеоконференцз'язок, різні веб-сервіси, сервіси передачі файлів, ІР-телефонія, засоби забезпечення доступу в Інтернет та ін. Серед цих сервісів також є критичні. Порушення їх роботи приведе до втрати працездатності критичних застосувань бізнес-рівня. Рівень захисту сервісів відповідає за безпеку та надання доступу до універсальних сервісів, а також забезпечує надання засобів захисту для рівня бізнес-застосувань. Рівень обчислювальних ресурсів включає в себе термінали користувачів, а також ресурси центру оброблення даних (ЦОД).

Підприємства малого розміру, а також філії практично завжди замість ЦОД використовують серверні або кластерні рішення, на яких розміщуються серверні частини ПЗ рівнів бізнес-застосувань та універсальних сервісів.

Усі обчислювальні ресурси пропонується реалізувати у вигляді хмарних ресурсів з сервісами у вигляді EaaS (все – як послуга).

У випадку використання зовнішніх хмарних ресурсів ІТ-підрозділ повинен вирішити організаційні питання та відповідає за контроль якості надання сервісів. Хмарні ресурси можна використовувати лише для реалізації некритичних сервісів критичної інформаційної інфраструктури.

Доступ користувачів здійснюється засобами рівня мережевої взаємодії. На цьому рівні критичні елементи повністю резервуються, основні фізичні топології, які використовується для з'єднання елементів – Full Mesh або Partial Mesh.

Система безпеки критичної інформаційної інфраструктури впроваджена на всіх рівнях [25, 27, 43]. Питання її створення в контексті критичної інформаційної інфраструктури розглянуті в [20, 35, 40–42, 47]. Основні її елементи структурно відносяться до I та III рівнів (позначка «3») [3].

В рамках створення системи безпеки критичної інформаційної інфраструктури потрібно реалізувати наступний комплекс заходів:

- класифікація активів критичної інформаційної інфраструктури, в тому числі, категорювання відомостей, поділ інформації за категоріями доступу;
- розробка та корегування політики інформаційної безпеки;
- розробка та корегування вимог до системи безпеки;
- реалізація вимог при створенні системи безпеки;
- розробка та впровадження системи моніторингу інформаційної безпеки;
- контроль.

Класифікація активів критичної інформаційної інфраструктури полягає в визначенні елементів, що є важливими для виконання бізнес-операцій. При класифікації активів потрібно враховувати: значимість (критичність) для виконання ключових бізнес-операцій; захищеність (наявні захисні заходи та їх достатність). В якості критеріїв класифікації пропонується використати наступні:

- критерій економічної значущості;
- критерій екологічної значущості;
- критерій значущості для забезпечення обороноздатності;
- критерій значущості для національної безпеки;
- критерій соціальної значущості;
- критерій значущості в частині реалізації управлінських функцій.

На базі запропонованих критеріїв критичні інформаційні інфраструктури має сенс розділити на три класи небезпечності: високої, середньої та малої.

Управління критичною інформаційною інфраструктурою здійснюється за допомогою СУІ. Завданнями СУІ є ведення інвентаризаційних БД щодо ресурсів

підприємства, моніторинг стану та управління підтримкою стану цих ресурсів на потрібному рівні, управління доступом та розподілом обчислювальних та телекомунікаційних ресурсів, контроль, аналіз поведінки та підтримка користувачів, планування та управління закупівлею ІТ-активів, автоматизація діяльності та ведення звітності ІТ-підрозділу, а також розв'язання багатьох інших задач, перелік яких визначається вимогами, розмірами, особливостями функціонування відповідної критичної інфраструктури. СУІ дозволяє не тільки зменшити витрати на утримання ІТ-підрозділу, але також є і основним засобом автоматизації роботи ІТ-підрозділу.

Обов'язкова складова ІТ-інфраструктури підприємств з критичною інфраструктурою – ІТ-підрозділ. Підприємства з критичною інфраструктурою через свою специфіку не можуть користуватись послугами аутсорсингових компаній. Головне завдання ІТ-підрозділу – обслуговування запитів бізнес-процесів в жорстких часових рамках. Вимоги до надання ІТ-послуг регламентуються пакетом угод про рівень сервісу (SLA). В SLA визначаються ключові показники ефективності (KPI) та якості (KQI) — обмежений набір об'єктивно вимірюваних параметрів, які дозволяють оцінити результативність роботи критичної інформаційної інфраструктури. Для підтримки значень KPI та KQI на зафіксованому в SLA рівні, адміністратори забезпечують безперервне функціонування критичної інформаційної інфраструктури, виконують її обслуговування та ремонт. Стандарт ISO 20000 [253] визначає важливу роль для СУІ, яка підтримує управління ІТ-послугами, а саме контроль метрик, що характеризують поведінку ІТ-процесів.

Для управління ІТ-послугами в критичній інформаційній інфраструктурі потрібно мінімум три відділи ІТ-підрозділу: підтримки критичних сервісів та інфраструктурних елементів, підтримки не критичних сервісів, управління критичною інформаційною інфраструктурою, яке відповідає за функціонування ІТМ в цілому.

З метою визначення окремих особливостей функціонування кожен критичну інформаційну інфраструктуру доцільно розглядати з різних ракурсів –

точок зору на архітектуру. В даній роботі використовуються три класичні точки зору на архітектуру з ФОАП TOGAF такі як бізнес-орієнтована, орієнтована на дані та орієнтована на технології і додаткова точка зору на архітектуру, орієнтована на критичні елементи.

6.1.2 Життєвий цикл критичної інформаційної інфраструктури

Створення КІІ представляє собою багатоетапний ітеративний процес. Життєвий цикл починається в момент появи ідеї створення нової критичної інформаційної інфраструктури [4, 37] і складається з наступних етапів (рис. 6.2):

- Планування (P)
 - стратегічне планування;
 - аналіз вимог та розробка специфікацій вимог до критичної інформаційної інфраструктури (в формі цілей та стратегій), визначення того, що повинна робити ця критична інформаційна інфраструктура. Результатами аналізу на даному етапі є: функціональні моделі, інформаційні моделі та відповідні їм діаграми; дерево цілей та шляхів досягнення;
- Проектування (D)
 - розробка методики проектування (попередньої та детальної) критичної інформаційної інфраструктури в цілому та її окремих підсистем та елементів;
 - визначення того, як будуть досягатись поставлені цілі та виконуватись поставлені завдання (технічні рішення);
 - розробка – створення функціональних компонентів та окремих підсистем критичної інформаційної інфраструктури, з'єднання підсистем в одне ціле. Розробка включає в себе всі роботи по створення критичної інформаційної інфраструктури та її компонент у відповідності до заданих вимог.

- атестаційне тестування на відповідність вимогам та доналаштування;
- Імплементация (I)
 - впровадження – установка та введення критичної інформаційної інфраструктури в дію;
 - експлуатація (використання) - роботи по підготовці до впровадження компонентів критичної інформаційної інфраструктури в експлуатацію. До основних напрямків діяльності на цьому етапі можна віднести:
 - конфігурування серверів, мережевого обладнання;
 - налаштування програмного забезпечення;
 - конфігурування робочих місць користувачів;
 - забезпечення експлуатаційною документацією;
 - проведення навчання персоналу;
- Експлуатація (Op)
 - локалізація проблем та усунення проблем їх виникнення;
 - супровід критичної інформаційної інфраструктури – забезпечення штатного процесу експлуатації критичної інформаційної інфраструктури;
- Оптимізація (Oz)
 - модифікація критичної інформаційної інфраструктури в рамках встановленого регламенту;
 - підготовка пропозицій по вдосконаленню, розвитку та модернізації елементів та алгоритмів критичної інформаційної інфраструктури.

Кожний етап життєвого циклу створення нової критичної інформаційної інфраструктури повинен включати наступні кроки [4]:

- визначення мети, постановку задач;
- визначення критерію оптимальності;
- побудову математичної моделі, збір даних;

- визначення алгоритму пошуку оптимального розв'язку;
- перевірку моделі та її рішень;
- реалізацію знайденого рішення;
- накопичення історії;
- ретроспективний перегляд результатів попередніх етапів та можливий відхід до попереднього етапу при виникненні нових ідей, алгоритмів, методів, що можуть суттєво покращити попередні результати.



Рис. 6.2. Життєвий цикл критичної інформаційної інфраструктури

Наявність фактору невизначеності початкової інформації приводить до використання поетапних математичних методів побудови, які дозволяють поступово зменшувати цю невизначеність. В такій ситуації, на перший план виходять методи пошуку допустимих розв'язків, які засновані на принципі задоволення потреб.

Принцип задоволення потреб формується на наступних засадах:

- мета розвитку критичної інформаційної інфраструктури багатозначна, кількість параметрів, що формують критерій оптимальності, більша за 1;
- при обґрунтуванні розв'язків доступна обмежена множина альтернатив;

– пошук найбільш оптимального розв'язку виконується за обмежений час або в реальному часі при обмежених матеріальних, інформаційних та інших ресурсах.

Згідно з [254] обрано 18 найбільш важливих задач, що розв'язуються на протязі всього життєвого циклу ІТ-інфраструктури. Розглянемо ці задачі в ракурсі життєвого циклу критичної інформаційної інфраструктури.

1. Управління взаємовідносинами з замовниками (Customer relationship management) – процес та заходи, що відповідають за налагодження відносин з внутрішніми і зовнішніми замовниками ІТ-послуг. Сприяє оптимізації клієнт-орієнтованих процесів: управління замовленням сервісів, інцидентами, рівнем обслуговування.

2. Управління фінансами (IT financial management) – використання традиційного бухгалтеру та заснованого на значущості інструментарію і методів для забезпечення ефективного бізнес-орієнтованого управління ресурсами інфраструктури при наданні клієнт-орієнтованих ІТ-послуг.

3. Управління знаннями (Knowledge management) – відповідає за збір, аналіз, зберігання та поширення знань в рамках ІТ-організації. Процес використовується для розповсюдження знань на всіх рівнях техпідтримки. Підвищення ефективності за рахунок зниження необхідності заново відкривати знання.

4. Управління релізами та розгортанням (наданням) (Release and deployment (service provisioning) management) – управління релізами процесу планування, складання графіка і контролю за переміщенням релізів (наборів апаратних засобів, ПЗ, документації чи інших компонентів, як єдиного цілого) для перевірки в реальному середовищі. Управління розгортанням відповідає за просування нових або модифікованих видів обладнання, ПЗ, документації або процесів в реальні умови. Управління релізами використовується для управління процесами, системами і функціями, необхідними для компоновки, збірки, тестування і розгортання релізів у виробництво.

5. Управління каталогом сервісів (Service catalog management) – створення та підтримка каталогу сервісів із зазначенням статусу, критичності, інтерфейсів і порядку надання послуг. Використовуються для надання деталей і стану існуючих сервісів і бізнес-процесів, підтримуваних інфраструктурою, а також сервісів, що знаходяться в розробці або вже не використовуються.

6. Метрики сервісів та звітність (Service metrics and reporting) – ідентифікація, збір та подання інформації про надання ІТ та бізнес-послуг, з вказанням типу і кількості послуг, їх внесок, ризики і поліпшення можливостей для бізнесу. Метрики використовуються для вимірювання ефективності процесів і сервісів, можливості їх резервування під критичні процеси і сервіси.

7. Управління життєвим циклом ІТ-активів (IT asset life-cycle management) – керування життєвим циклом ІТ-активів від закупівлі до утилізації, з акцентом на розташуванні, вартості та праві власності, а також критичності процесу або послуги, яка цим активом забезпечується. Автоматизація процесу дозволяє отримати знання щодо доступних апаратних і програмних активів для прийняття рішень, управління потужністю і планування. При використанні хмарних технологій необхідно відстежувати інфраструктурні та програмні ІТ-активи в режимі реального часу.

8. Управління портфелем послуг (Portfolio management of services) – опис послуг в термінах значимості для бізнесу, сформульованих з погляду замовника, зазначаючи, які ІТ-послуги є, які з них критичні, які важливі, як вони об'єднані в пакет і які переваги надають.

9. Організація часу та управління ресурсами (Time and resource management) – розподіл часу персоналу і ресурсів для задач операційної діяльності, проектів та/або функцій. Процес планування, призначення і контролю часу і ресурсів для ефективного розміщення співробітників з необхідними навичками на обслуговування критичних та важливих процесів та послуг.

10. Управління потужністю (Capacity management) – оптимізація потужності ІТ-послуг, бізнес-послуг та інфраструктури таким чином, щоб

доступні ресурси були здатні задовольнити потреби критичних процесів критичної інформаційної інфраструктури, а також, по можливості, надати не критичні послуги з узгодженим рівнем обслуговування економічно-ефективно і своєчасно. Процес включає планування, оцінку та постійне управління потужностями. Процес покликаний уникнути непотрібних інвестицій, що не відповідають потребам бізнесу, і ситуацій, коли робота критичної інформаційної інфраструктури знаходиться під загрозою через брак або неефективне використання існуючих ресурсів, що підтримують ІТ-послуги або бізнес-послуги.

11. Управління змінами (Change management) – відповідає за управління і моніторинг життєвого циклу і ризиків всіх змін (додавання, модифікація або видалення ІТ-послуг, конфігураційних образів, процесів, документації тощо) гарантуючи, що використовуються нормативні методи і процедури відстеження змін. Процес необхідний для забезпечення якості та безперервності всіх ІТ-послуг (критичних і не критичних), починається з запиту на зміну, що вимагає реєстрації зміни, прийому та класифікації, затвердження та планування, фактичного здійснення змін, оцінки змін і закриття запиту на зміну.

12. Управління доступністю та продуктивністю (Availability and performance management) - відповідає за вимірювання та поліпшення доступності та продуктивності не критичних ІТ-послуг та процесів і пов'язаних з ними елементів конфігурації або ІТ-активів (сервери, мережі, БД, застосування). Використовується для пошуку причин проблем з продуктивністю застосувань або бізнес-послуг. Розуміння залежності роботи користувача від конкретних застосувань, БД, ПО проміжного шару і компонентів інформаційно-телекомунікаційної системи (ІТС) в комплексі дозволяє ІТ-підрозділу управляти і автоматизувати доступність і продуктивність не критичних процесів, згідно вимог бізнесу. В ракурсі критичних ІТ-послуг та процесів відповідає за пошук додаткових ресурсів для підвищення надійності та живучості критичної інформаційної інфраструктури.

13. Управління конфігураціями (Configuration management) – управління конфігураційними елементами, що надають ІТ-послуги або частиною ІТ-послуг, протягом життєвого циклу елементів. Процес використовується для підтримки бізнес-послуг та їх адаптації відповідно до потреб бізнесу, і необхідний для підтримки роботи критичної інформаційної інфраструктури на належному рівні та зниження витрат на ІТ-послуги та підтримку інфраструктури.

14. Управління рівнем послуг (Continual service-level management) – відстежує процеси та послуги для їх налаштування згідно заданого SLA до мінливих потреб бізнесу. Відстежує узгодженість цілей ключових не критичних ІТ-сервісів між собою і їх відповідність цілям бізнесу і користувачів, а також гарантує заданий рівень критичних процесів і ІТ-послуг.

15. Управління подіями (Event management) – управління подіями протягом всього їх життєвого циклу – від отримання і протоколювання повідомлень тривоги до автоматизованого розв’язання. Дозволяє зрозуміти, які проблеми виникають у критичній інформаційній інфраструктурі, для подальшого розв’язання автоматизованим шляхом або шляхом ескалації подій експертам предметної області.

16. Управління інцидентами (Incident management) – управління незапланованим перериванням критичних та не критичних або зниженням якості не критичних ІТ-послуг від виявлення інциденту до відновлення функціонування. Процес включає всі проблеми, які порушують або можуть порушити не критичні поточні сервіси і, як правило, інформує користувача або інструментарій управління інфраструктурою. У випадку критичних сервісів процес локалізує проблему та організує роботу по найшвидшому його відновленню.

17. Управління проблемами (Problem management) – управління життєвим циклом проблем для запобігання інцидентів і мінімізації наслідків інцидентів, яким не можна запобігти. Реактивне управління проблемами полягає в ліквідації наслідків інциденту. Проактивне управління дозволяє вирішити проблеми, перш ніж вони зроблять негативний вплив на бізнес-процеси. Процес

використовується при аналізі та пошуку причини інцидентів та діяльності, спрямованої на запобігання появи інцидентів.

18. Управління замовленням сервісів (Service request management) – управління життєвим циклом запитів користувачів для отримання інформації про сервіси або наявні аналоги, поради чи доступ до ІТ-послуг. Зазвичай входить в робочі процеси служби підтримки.

В табл. 6.1 вказані етапи життєвого циклу та задачі, які виконуються на даному етапі.

Табл. 6.1. Задачі етапів життєвого циклу КІІ

№пп	Задача	Етап
1.	Управління взаємовідносинами з замовниками	P, D, Op
2.	Управління фінансами	P, D, I, Oz
3.	Управління знаннями	P, D, Oz
4.	Управління релізами та розгортанням	P, D, I, Oz
5.	Управління каталогом сервісів	P, D, Oz
6.	Метрики сервісів та звітність	P, D, Oz
7.	Управління життєвим циклом ІТ-активів	P, D, I, Op, Oz
8.	Управління портфелем послуг	P, D, Oz
9.	Організація часу та управління ресурсами	I, Op, Oz
10.	Управління потужністю	I, Op, Oz
11.	Управління змінами	Op, Oz
12.	Управління доступністю та продуктивністю	Op, Oz
13.	Управління конфігураціями	Op, Oz
14.	Управління рівнем послуг	Op, Oz
15.	Управління подіями	Op
16.	Управління інцидентами	Op

17.	Управління проблемами	Op
18.	Управління замовленням сервісів	Op

Примітка: P – етап планування, D – етап проектування, I – етап імплементації, Op – етап експлуатації, Oz – етап оптимізації.

Важливе значення при створенні критичної інформаційної інфраструктури має правильний вибір оптимальних рішень на всіх стадіях життєвого циклу, а особливо, на перших його етапах.

Тому, всеосяжне дослідження проблеми побудови критичної інформаційної інфраструктури можливе тільки при використанні наступних аспектів системного підходу:

- функціонального, що дозволяє визначити функції підсистем, елементів та критичної інформаційної інфраструктури в цілому, а також встановити функціональні залежності між ними;
- елементного, що дозволяє визначити ключові елементи, що надають критичній інформаційній інфраструктурі її особливості, і таким чином, через аналіз перейти до синтезу цілісності критичної інформаційної інфраструктури;
- структурного, що дозволяє встановити взаємозв'язки між елементами та структуру в цілому, їх цілі, і таким чином, визначитись щодо використання технічних рішень;
- інтегративного, що дозволяє визначити механізми та фактори, які впливають на розвиток та функціонування критичної інформаційної інфраструктури;
- комунікаційного, що дозволяє визначити місце критичної інформаційної інфраструктури в оточуючому середовищі та зовнішні фактори, які можуть на неї впливати;
- історичного, що дозволяє зберігати та використовувати набуту інформацію для подальшого розвитку критичної інформаційної інфраструктури.

6.1.3 Загальна схема пошуку проектних рішень на базі КДК-підходу

Для побудови загальної схеми пошуку проектних рішень при проектуванні архітектури критичної інформаційної інфраструктури з врахуванням недоліків і переваг методів, підходів та ФОАП, що розглянуто в першому розділі, потрібно також розглянути існуючі стратегії прийняття рішень та чинники, які впливають на вибір стратегії.

Стратегії прийняття рішень. Стратегії прийняття рішень зазвичай діляться на дві основні категорії: компенсаційні та некомпенсаційні.

В процесі прийняття компенсаційних рішень альтернативи оцінюються вичерпно, беручи до уваги всі критерії та їх компроміси. Критерії з високими пріоритетом компенсують критерії з більш нижчим. Нарешті, обирається альтернатива з найвищим балом. Компенсаційні стратегії спрямовані на забезпечення найкращих можливих рішень, беручи до уваги дані про рішення. Проте компенсаційні стратегії вимагають повної інформації про те, як альтернативи оцінюються за всіма критеріями, і для оцінювання потребують багато часу.

З іншого боку, некомпенсаційні стратегії узгоджуються з концепцією обмеженої раціональності. Це означає, що обґрунтування прийняття рішення обмежується такими факторами, як важкі обмеження, часові обмеження та когнітивне навантаження особи, що приймає рішення. Таким чином, некомпенсаційні стратегії оцінюють альтернативи евристично, використовуючи лише обмежену кількість критеріїв та компромісів.

Основними характеристиками некомпенсаційних стратегій є те, що вони:

- зменшують зусилля щодо прийняття рішень;
- не вимагають інформації, необхідної для прийняття рішення.

Таким чином, для осіб, які приймають рішення, загальною практикою є застосування некомпенсаційних стратегій в ситуаціях, обмеження яких впливають на процес прийняття рішень. Прикладами таких ситуацій можуть бути часовий стрес або важкі обмеження. Однак, за визначенням, особи, що

приймають рішення, не враховують усі критерії при застосуванні стратегій некомпенсаційних рішень.

І на останок, в деяких випадках вимагається використання комбінації компенсаційних та некомпенсаційних стратегій прийняття рішень [255–257]. Наприклад, дизайнер, який приймає рішення, починає свій процес оцінювання, виключаючи альтернативи, які не відповідають певним некомпенсаційним критеріям, і лише після цього оцінює залишкові альтернативи за допомогою компенсаційної стратегії.

Чинники вибору стратегії прийняття рішень. Рішення, зокрема на вибір стратегії прийняття рішень, впливають такі чинники, як час, повнота інформації, когнітивне навантаження особи, що приймає рішення та багато іншого [258].

Коротко підсумуємо ці чинники і як вони впливають на процес прийняття рішень:

- *Часовий стрес:* одна з найпоширеніших ситуацій у процесі прийняття рішень – часовий стрес. Приймаючи під час тиску повинні приймати критичні рішення. Як правило, ці рішення приймаються в останню мить, адхоком та без достатньої раціоналізації.

- *Неструктуровані проблеми:* проблема рішення часто є складною з точки зору причинно-наслідкових зв'язків, кореляцій та петлі зворотного зв'язку між відповідними факторами. Таким чином, проблеми вирішення важко зрозуміти, а також оцінити їх з точки зору наслідків та результатів, які вони мають.

- *Неповнота інформації:* це означає, що на практиці інформація може бути неоднозначною або навіть відсутньою.

- *Зміщені, погано визначені чи конкуруючі цілі:* у процесі прийняття рішень люди, що приймають рішення, можуть мати суперечливі цілі. Орган, який приймає рішення, повинен правильно визначати кожен з цих цілей при прийнятті рішення.

- *Дії та зворотній зв'язок:* прийняття рішень містить ряд циклів, які дизайнер повинен враховувати [259]. Ранні помилки та неякісна інформація

генерують рішення, які слід переглянути. Наприклад, порушення в архітектурному проекті важко визначити на ранньому етапі, і це означає, що процес прийняття рішень повинен бути повторений знову після їх виявлення.

- *Ситуація з кількома гравцями*: коли до процесу прийняття рішень залучають декілька зацікавлених сторін, ситуація ускладнюється. Зацікавлені сторони мають різні інтереси, цілі та очікування від конкретного рішення. Ще однією поширеною проблемою є відсутність спільного розуміння між зацікавленими сторонами стосовно конкретної проблеми. У багатьох ситуаціях така ситуація може призвести до затримок у прийнятті рішень, що, можливо, призведе до перегляду рішення.

- *Організаційні цілі та норми*: організації діють за конкретними цілями та нормами [259]. Особи, що приймають рішення, повинні приймати рішення в контексті цих цілей та норм, а також повинні уникати прийняття рішень на основі лише їхніх особистих уподобань. Особи, які приймають рішення, незалежно від їхніх особистих переваг, повинні оцінювати альтернативи критеріям, які встановлює організація.

Беручи до уваги вищенаведене, а також враховуючи результати, отримані в розділі I, побудована загальна схема пошуку проектних рішень щодо архітектури критичної інформаційної інфраструктури [6, 48], яка представлена на рис. 6.3.

Основними елементами схеми є:

- *Вибір дизайну КІІ* – фактичне рішення щодо проектування, яке було зроблено після процесу оцінки, верифікації та має структуру, функціональні задачі та життєвий цикл, представлені в розділах 6.1.1 та 6.1.2.

- *Архітектурна проблема КІІ* – архітектурна проблема, яку архітектори критичної інформаційної інфраструктури повинні вирішити.

- *Артефакт КІІ* – прямий результат, отриманий з набору виконаних виборів дизайну КІІ, або представлення цього результату. Наразі артефакт КІІ використовується для позначення архітектурних уявлень. Зокрема, він використовується як концепція-міст для мови моделювання АП ArchiMate,

завдяки якій артефакт КІІ дозволяє нам пов'язати вибір дизайну КІІ з елементами «концепція» в ArchiMate. Фактично, кожний *Артефакт КІІ* є архітектурною моделлю, яку перед використанням слід верифікувати. Методи верифікації архітектурних рішень щодо КІІ та їх основні означення представлено в розділі ІІІ.

– *Рівень моделі TOGAF* – відповідно до мови ArchiMate [260] та моделі TOGAF [142] підприємство визначається на трьох рівнях: Business, Application та Technology. Використовуючи ці три рівні, можна описати підприємство цілісно, показуючи не тільки програмне забезпечення та фізичну ІТ-інфраструк-

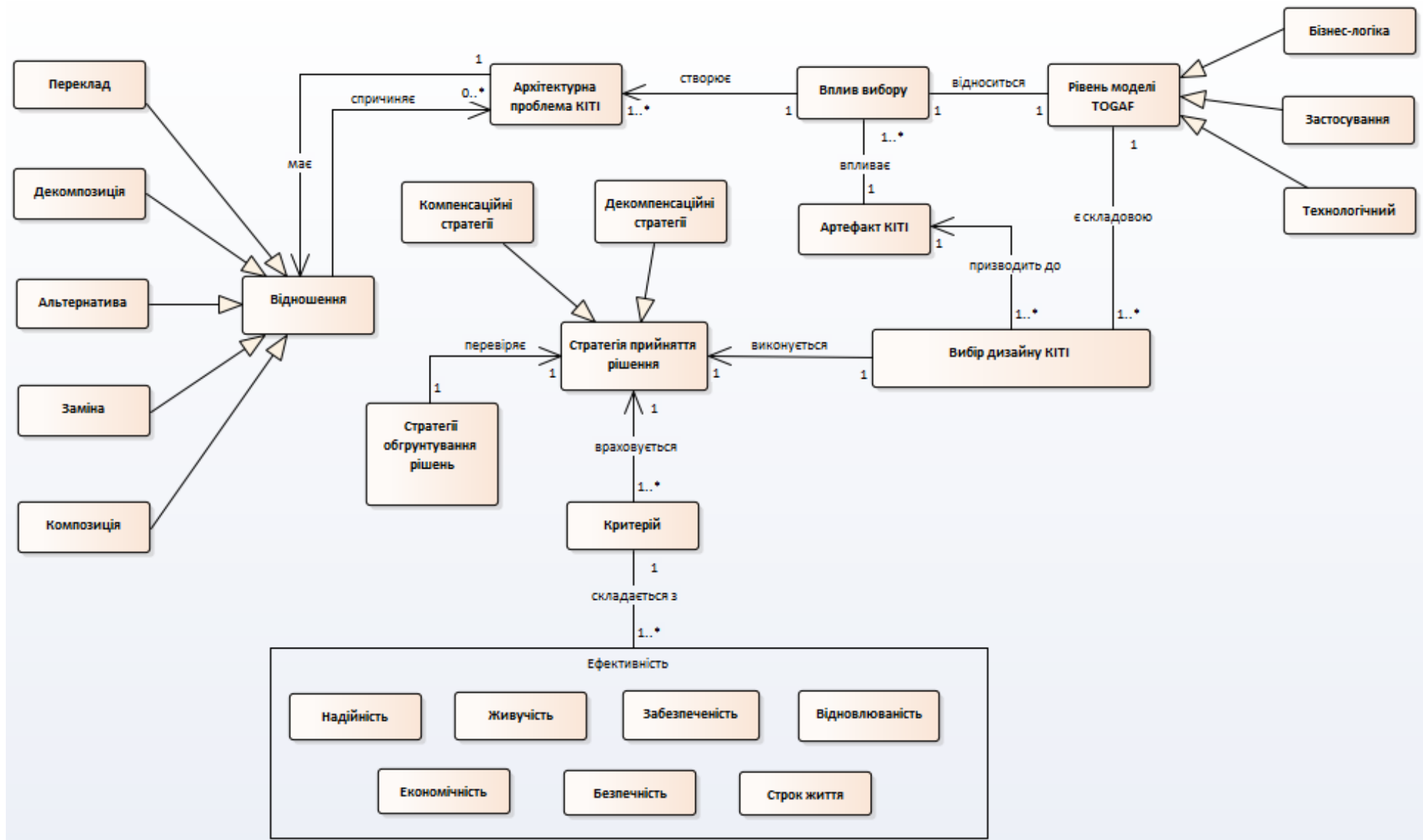


Рис. 6.3. Загальна схема пошуку проектних рішень щодо архітектури

туру (виражені через рівні застосування та технології), але також, і яким чином ІТ впливає/вплинув на діяльність підприємства та його бізнес-стратегію і процеси.

– *Вплив вибору* – цей елемент визначає непередбачуваний наслідок від вже прийнятого рішення на артефакт КІІ. Він протиставляється передбачуваним наслідкам, про які свідчать такі відношення, як композиція або переклад. У поточній повсякденній практиці архітектори моделюють передбачувані наслідки у вигляді сценаріїв за допомогою правил типу «ЯКЩО, ТО». Але на жаль, не можна передбачити всі можливі впливи прийнятих рішень щодо АІІ. Це особливо стосується критичної інформаційної інфраструктури, де розглядається вплив на всю критичну інфраструктуру, а не на окремі (наприклад, технічні) її частини. Деякі наслідки вибору дизайну КІІ виявляються на етапі реалізації або під час підтримки існуючого проекту архітектури. Ці непередбачені наслідки сприймаються точно елементом «Вплив вибору». Для архітектора головна корисність представлення непередбачених впливів вибору полягає в тому, що вони можуть бути використані для уникнення рішень з негативними наслідками у майбутніх архітектурних рішеннях.

Концепції процесу прийняття рішень. Концепції процесу прийняття рішень мета-моделі зосереджені на ухваленні стратегій прийняття рішень, які використовувались під час процесу архітектурного проектування конкретного дизайну КІІ, обґрунтуванні вибору стратегії прийняття рішень та наявних альтернатив та критеріїв. Опис відповідних елементів мета-моделі наведений нижче:

– *Стратегія прийняття рішень.* Ця концепція охоплює стратегію прийняття рішень, яку використовує архітектор підприємства, щоб оцінити альтернативи та зробити фактичне рішення щодо вибору дизайну КІІ. Як вже згадувалося раніше, стратегії прийняття рішень характеризуються як компенсаційні, некомпенсаційні, або як гібрид цих двох.

- *Компенсаційна стратегія:*
 - ✓ *Зважена сума (WADD):* у стратегіях WADD критерії оцінки альтернатив мають різні ваги. Оцінка кожної альтернативи обчислюється шляхом множення кожного критерію на його вагу, а потім, беручи суму цих значень. Обирається альтернатива з найвищим балом.
 - ✓ *Рівна вага:* оцінка кожної альтернативи обчислюється таким же чином, як і в стратегіях WADD. Різниця полягає в тому, що критерії мають однакову вагу.
- *Некомпенсаційна стратегія:*
 - ✓ *Кон'юнктивна:* у кон'юнктивних стратегіях альтернативи, які не відповідають одному або декільком критеріям, негайно виключаються з вибору.
 - ✓ *Диз'юнктивна:* у цій стратегії альтернатива обирається, якщо вона відповідає конкретному критерію, незалежно від значень інших критеріїв.

Некомпенсаційні стратегії є доцільними для оцінки альтернатив у ситуаціях неповноти інформації та відсутності числових даних. Наприклад, критерій «потужний комутатор» є суб'єктивним і його важко обчислити. Альтернативи, які не відповідають цьому критерію, можуть бути усунені з вибору.

Слід також зазначити, що в застосуванні додаткових стратегій прийняття рішень немає обмежень.

Елемент «Критерій» - це атрибут, який у більшій чи меншій мірі повинен бути задоволений протягом процесу прийняття рішень. Критерії використовуються як для компенсаційних, так і для некомпенсаційних стратегій прийняття рішень. Наприклад, якщо була застосована диз'юнктивна стратегія, приймається рішення про дотримання одного чи декількох критеріїв. Крім того, поняття *вартості* та *ваги* критерію також враховуються у моделі.

Концепція вартості являє собою значення, яке дизайнер визначає під час процесу оцінки, і яке вказує на те, наскільки добре артефакт відповідає певному критерію. Концепція ваги являє собою важливість цього критерію і зазвичай використовується в стратегіях WADD. До критеріїв оцінки критичної інформаційної інфраструктури відносяться [3]:

- некомпенсаторні:
 - надійність – показник надійності критичної інформаційної інфраструктури в період експлуатації;
 - живучість – можливість виконувати свої функції при втраті ресурсів, підсистем і т. ін.;
 - безпечність – показник неможливості виконання несанкціонованих дій, спрямованих на порушення роботи критичної інформаційної інфраструктури чи її частин;
- компенсаторні:
 - забезпеченість – показник максимальної кількості процесів та сервісів, що обслуговуються;
 - відновлюваність – тривалість відновлення готовності до експлуатації;
 - строк життя;
 - економічність – витрати різноманітних ресурсів на забезпечення функціонування критичної інформаційної інфраструктури;
- гібридного типу (можуть бути і компенсаторними, і некомпенсаторними):
 - ефективність – поєднання вище згаданих параметрів в кожному окремому випадку під визначену задачу;
 - вартість проекту.

Елемент «Стратегії обґрунтування рішень» - у процесі прийняття рішень архітектору потрібно не лише вибрати деякі альтернативи (фактичний

процес прийняття рішень), але також вибрати стратегію прийняття рішень, що задовольняє його поточні потреби в оцінці. Фактично, цей елемент включає в себе обґрунтування стратегії прийняття рішень, яка була обрана для процесу оцінки.

Як вже визначалось вище, різні фактори впливають на процес прийняття рішень, і відповідальним особам слід відповідно адаптувати свою стратегію прийняття рішень. Поняття стратегічного обґрунтування дає змогу особі, що приймає рішення, задовольнити його поточні потреби в оцінці.

Важливим елементом моделі є *відношення*. Роль концепцій відношень полягає в тому, щоб різноманітні типи зв'язків між елементами «Вибір дизайну КІІ» були явними. Виходячи з онтологій проектних рішень архітектури програмного забезпечення, в пропонованій моделі визначені п'ять типів відношень [6, 48]:

– *Переклад* – відношення даного типу ілюструють зв'язки між елементами «Вибір дизайну КІІ» та «Архітектурна проблема КІІ», що належать до різних рівнів або різних артефактів КІІ. Відповідно до мета-моделі відношення перекладу застосовуються для зв'язку елементів «Вибір дизайну КІІ» між собою або для зв'язку з елементами «Архітектурна проблема КІІ». Відношення перекладу базуються на типі загального відношення «is related to», тільки адаптованого до основної термінології архітекторів критичних інформаційних інфраструктур. Одне з ключових завдань для архітектора – шляхом комунікації перекласти вимоги, які нові артефакти КІІ накладають (*Архітектурна проблема КІІ*) для рішень, які будуть підтримувати ці вимоги за допомогою іншого артефакту КІІ.

– *Декомпозиція* – відношення декомпозиції означають, як загальні рішення про АП (*Вибір дизайну КІІ*) розкладаються на більш детальні проектні рішення.

- *Альтернатива* – тип альтернативних відношень ілюструють рішення, які були відхилені (альтернативи), для вирішення конкретної архітектурної проблеми КІІ.
- *Заміна* – відношення заміни пояснюють, як одне рішення щодо архітектури заміщає негативний результат іншого рішення.
- *Композиція* - відношення композиції визначають, як детальні проектні рішення складаються в загальне рішення про АП (*Вибір дизайну КІІ*).

6.2 Структура і принципи функціонування ЄСПК

Єдина система проектування критичної інформаційної інфраструктури включає наступні елементи (рис. 6.4):

- середовище для моделювання Enterprise Architect;
- систему верифікації та перетворення моделей;
- керовану моделлю систему розподілу ресурсів критичної інформаційної інфраструктури;
- систему автоматичного розгортання та управління критичною інформаційною інфраструктурою;
- систему обґрунтування проектних рішень.

Перший елемент ЄСПК використовується для створення розширених UML-моделей проектних рішень та фіксації за їх допомогою всіх міркувань в процесі пошуку оптимальної архітектури конкретних проектних рішень. Даний елемент дозволяє сформувати архітектору конкретні специфікації окремих проектних рішень (компоненти, програмні засоби тощо), отриманих у результаті їх пошуку у відповідності до принципів, критеріїв та підходу, які описані в розділі 6.1, а також отримати файл проекту у вигляді ХМІ, який потім буде використано при автоматичній верифікації моделей проектних рішень.

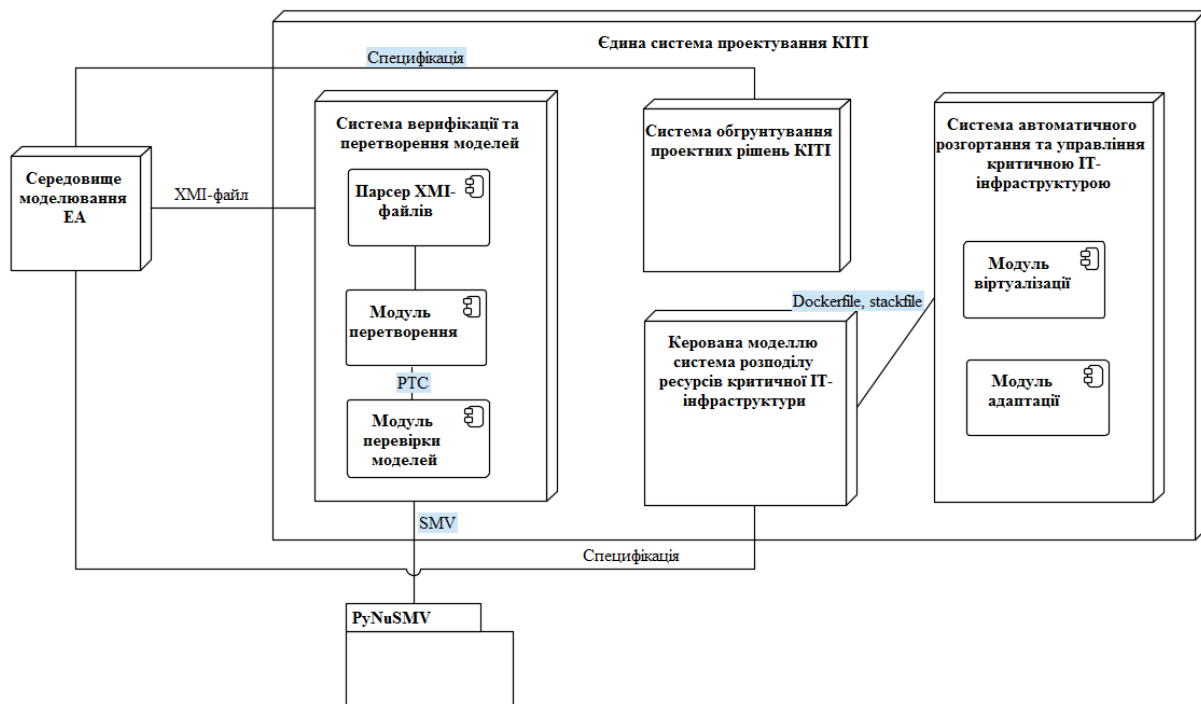


Рис. 6.4. Діаграма розгортання ЄСПК

Система верифікації та перетворення моделей включає наступні елементи:

- парсер ХМІ-файлів;
- модуль перетворення діаграм з ХМІ-файлів в розмічені транзиційні системи;
- модуль перевірки моделей.

Система дозволяє витягнути з ЕА діаграми у вигляді ХМІ-файлів та трансформувати їх в нотацію SMV [261]. Після трансформації їх можна перевірити на правильність за допомогою модуля верифікації моделей, створеного на базі пакету PyNuSMV [262].

Система обґрунтування проектних рішень дозволяє провести дослідження отриманих проектних рішень за варіантами специфікацій з ЕА за допомогою побудованих моделей та визначити оптимальні рішення.

Керована моделлю система розподілу ресурсів критичної інформаційної інфраструктури дозволяє під задану конфігурацію ресурсів та компонент керувати процесом автоматичного розгортання критичної інформаційної

інфраструктури. Система розраховує оптимальний варіант розміщення ресурсів та генерує відповідні файли конфігурацій Dockerfile та Stackfile.

Система автоматичного розгортання та управління критичною інформаційною інфраструктурою розгортає відповідно до отриманих Dockerfile та Stackfile фізичну інфраструктуру, запускає відповідні сервіси та управляє нею у відповідності з заданими критеріями.

6.3 Технологічні особливості реалізації розроблюваної ЄСПК

Єдина система проектування критичної інформаційної інфраструктури представляє собою складну гетерогенну систему. Основа проблема, яку потрібно вирішити при її створенні – вибір технології налагодження взаємодії між її компонентами і системами. Існує декілька варіантів вирішення цієї проблеми, які і розглянуто в даному розділі.

Веб-сервіси. Зазвичай, технології CORBA і DCOM використовуються для побудови застосунків внутрішньо-корпоративної мережі, і вони не можуть перетинати кордони мережевого екрану. Для підтримки інтеграції інтернет-застосунків в глобальній мережі, в останні роки були розроблені веб-сервіси. Веб-сервіс має набір функцій, які упаковані як єдиний об'єкт і можуть бути опубліковані в мережі для використання іншими програмами. Доступ до веб-сервісів можливий на будь-якій мові і може працювати в будь-якій операційній системі. Вони використовують HTTP-протокол в якості основного транспортного протоколу, що дозволяє передавати запити функцій через мережевий екран. Для форматування вхідних і вихідних параметрів запиту використовується XML, тому запит не прив'язаний до конкретної мови програмування компонентів або узгоджень про виклик об'єктів. Основні протоколи веб-сервісів включають SOAP (протокол для взаємодії з веб-службою), WSDL (мова визначення інтерфейсу веб-сервісу) і UDDI (сховище

для зберігання посилань на веб-сервіси, щоб клієнти могли їх знайти) [263, 264].

Веб-сервіси побудовані на основі концепції обміну повідомленнями [265] та складаються з трьох компонентів: брокеру сервісів, замовника і постачальника сервісів. Постачальник публікує сервіси в брокері сервісів. Потім, коли замовник запитує, де знайти відповідного постачальника сервісів, брокер сервісів зв'язує замовника з постачальником (рис. 6.5) [264].



Рис.6.5. Архітектура веб-сервісів

Веб-сервіси відіграють важливу роль для інтеграції різних систем підпрограмного забезпечення. Вони можуть забезпечити рівень абстракції для інтеграції сучасних застосунків [266].

Шина даних. ESB – це комплексне підпрограмне забезпечення, що поєднує в собі набір інструментів та надає інфраструктуру для керування процесом обміну повідомленнями. Зазвичай до складу ESB входять інструменти для керування чергою повідомлень, кодування та декодування даних, автоматична переадресація, трансформування, моніторинг. Кожен компонент може бути замінений, а також можуть бути додані нові компоненти.

На рис. 6.6 зображене середовище інтеграції з використанням ESB.



Рис. 6.6. Середовище інтеграції з використанням ESB

Для визначення кращого способу реалізації обрано впроваджені в стандарт RM-ODP вимоги до розподілених систем [267, 268]. Ці вимоги можуть бути використані для порівняльного аналізу при виборі методу інтеграції систем:

- Відкритість: інфраструктура проміжного програмного забезпечення повинна дозволяти розширювати застосування, не поглиблюючись в деталі його реалізації.
- Масштабованість: інфраструктура проміжного програмного забезпечення повинна бути гнучкою та дозволяти масштабувати застосунок в горизонтальному та вертикальному напрямках.
- Продуктивність: інфраструктура проміжного програмного забезпечення повинна забезпечувати ефективне і передбачуване, якщо необхідно, виконання застосунків, побудованих над ним.
- Прозорість поширення: це властивість, що визначає, чи сприймається застосунок користувачами або розробниками в цілому, а не як набір незалежних складових елементів.

Вимоги прозорості поширення є досить загальними, і вони зазвичай поділяються на:

- Прозорість доступу: інфраструктура повинна забезпечувати однаковий доступ до локальних і віддалених елементів застосунку.
- Прозорість розташування: інфраструктура повинна забезпечувати доступ до елементів застосунку без знання їх фізичного розташування.

- Прозорість паралелізму: інфраструктура повинна забезпечувати паралельну обробку ресурсів без втручання.
- Прозорість збоїв: інфраструктура повинна забезпечувати надання послуг, незважаючи на виникнення збоїв.
- Прозорість міграції: інфраструктура повинна забезпечувати засоби для зміни місця розташування елементів застосунку без впливу на інші елементи системи.
- Прозорість персистентності: інфраструктура повинна забезпечувати засоби для приховування дезактивації і реактивації елементів з інших елементів, які їх використовують.
- Прозорість транзакцій: інфраструктура повинна забезпечувати засоби для координації виконання атомарних і ізольованих транзакцій.

В цілому, кожна інфраструктура проміжного програмного забезпечення передбачає використання архітектурного стилю, якого повинні дотримуватися застосунки, побудовані на його основі. Для цього архітектурного стилю необхідно дотримуватися трьох основних принципів підтримки розробки відкритих і масштабованих застосунків:

- Модульність: застосунок має складатися з набору елементів, кожен з яких надає послуги, які використовуються іншими. Модульність дозволяє ідентифікувати залежності між елементами, що входять в систему. Отже, вона дозволяє визначати, на які елементи впливає можливе додавання, видалення або оновлення сервісів.
- Інкапсуляція: для кожного складового елемента існує чіткий поділ між інтерфейсом елемента і його реалізацією. Інтерфейс являє собою чітко визначену специфікацію послуг, що надаються, контракт між елементом і сутностями, що використовують його. Реалізація елемента може бути безпечно замінена за умови збереження інтерфейсу елемента. Зміна інтерфейсу елемента без шкоди для загальної цілісності додатки вимагає, щоб інша частина програми не залежала від цього конкретного інтерфейсу під час зміни.

– Спадкування: специфікація інтерфейсу елемента (відповідно реалізація) може бути отримана з іншого. Отриманий інтерфейс (відповідно реалізація) надає щонайменше сервіси базового інтерфейсу (відповідно реалізації). Спадкування дозволяє використовувати вертикальну і горизонтальну композицію сервісів.

Загальний архітектурний стиль, використаний в рамках порівняння відповідає вищезазначеним принципам. Він також спирається на архітектурний стиль, запропонований в RMODP для відкритих розподілених систем [267].

Результати порівняння підходів за наведеними критеріями наведено в табл. 6.1.

Табл. 6.1. Результати порівняння підходів

Критерії порівняння		DCOM	CORBA	JRMI	MOM	Веб-сервіси	ESB
Загальні	Конкретні						
	Модель даних	Об'єкти DCOM	Об'єкти CORBA	Java об'єкти	повідомлення	повідомлення	повідомлення
	Канал передачі даних	DCE RPC	GIOP	RMI	AMQP MSQP HTTP	HTTP	HTTP
	модульність	+	+	+	+	+	+
	інкапсуляція	+	+	+	+	+	+
	спадкування	+	+	+	-	+	-
Масштабованість		+/-	+	+/-	-	+	+
Продуктивність		-	-	+	+	+	+
	доступу	+	+	+	-	+	+
	розташування	-/+	+/-	-/+	+	-	+
	паралелізму	+/-	+/-	+	+	+	+
	збоїв	-	+	-	+	-	+
	міграції	-	+/-	-/+	-	-	+
	персистентності	-	+	+	-	-	+
	транзакцій	+	+	+/-	-	+/-	+

Для вибору відповідного вимогам розроблювальної системи методу, необхідно враховувати ряд критеріїв. В даному розділі запропоновано набір основних критеріїв, за допомогою яких можна порівняти методи інтеграції та визначити ті, що відповідають вимогам системи вже на етапі проектування. Вірно обраний метод інтеграції дозволить побудувати ефективну розподілену

гетерогенну систему використовуючи менший ресурс та спростить процес розробки під час створення програмного забезпечення та в майбутньому, під час розширення. Для побудови ЄСПК вирішено використати метод інтеграції з використанням шини обміну.

6.4 Розробка ЄСПК

6.4.1 Керована моделлю система розподілу ресурсів критичної інформаційної інфраструктури

Система має архітектуру типу “клієнт-сервер”. Діаграма розгортання системи наведена на рис. 6.7. Загалом, система складається з двох частин:

1. серверного застосунку, що виконує навчання нейронної мережі та реалізує алгоритм структурної оптимізації;
2. клієнтського застосунку, що реалізує графічний інтерфейс користувача.

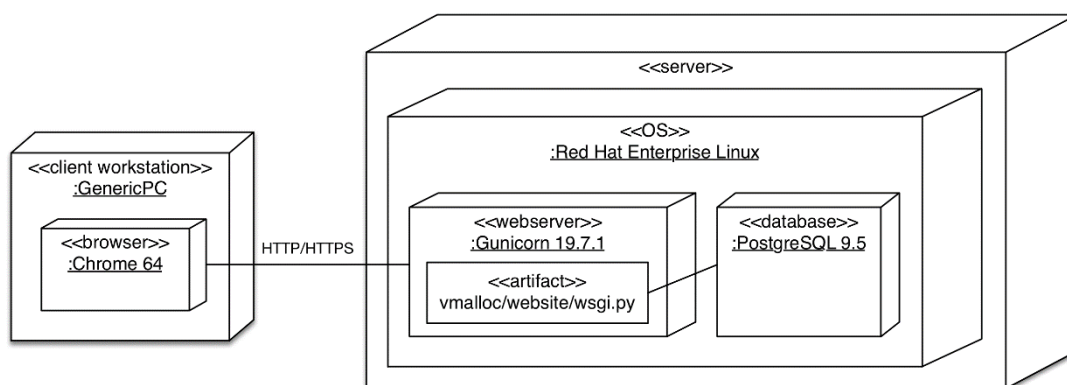


Рис. 6.7. Діаграма розгортання системи

Логічно система розділена на 4 компоненти, кожен з яких може бути замінений незалежно один від одного (рис. 6.8). До складу входять:

- головний модуль управління – здійснює аналіз, планування та управління станом критичної ІТ-інфраструктури;

- система розрахунку розміщення – здійснює пошук субоптимального розміщення елементів критичної інформаційної інфраструктури за прийнятний час;
- модуль рішення задачі пакування – здійснює пошук субоптимального рішення задачі багатомірного пакування за прийнятних час. Модуль є необхідним для розрахунку розміщення;
- система управління інфраструктурою – здійснює фактичне управління станом об’єктів критичної ІТ інфраструктури.

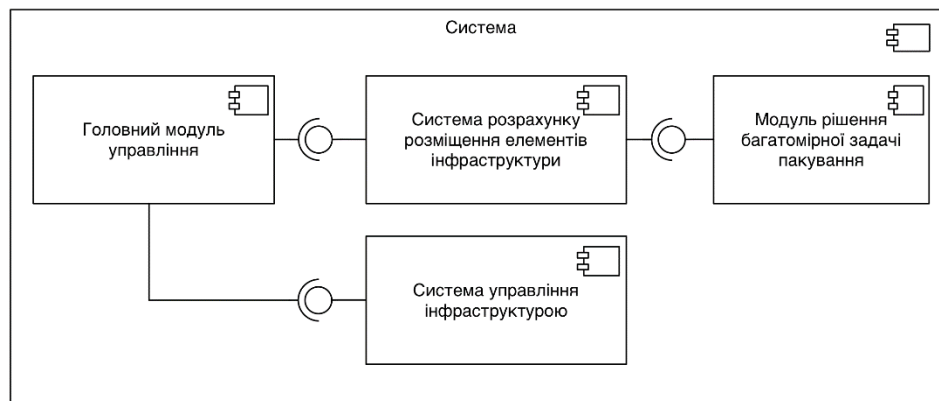


Рис. 6.8. Діаграма компонентів системи

Взаємодія із системою здійснюється за допомогою веб-інтерфейсу, у якому множина доступних операцій визначається в залежності від ролі користувача:

- роль «клієнт» - користувач, що є власником критичних сервісів, що підтримуються за рахунок потужностей критичної інформаційної інфраструктури;
- роль «оператор» - володіє адміністративними правами та є представником компанії-власника критичної інформаційної інфраструктури.

Залежності між ролями користувачів та доступними операціями зображені у вигляді діаграми прецедентів на рис. 6.9.

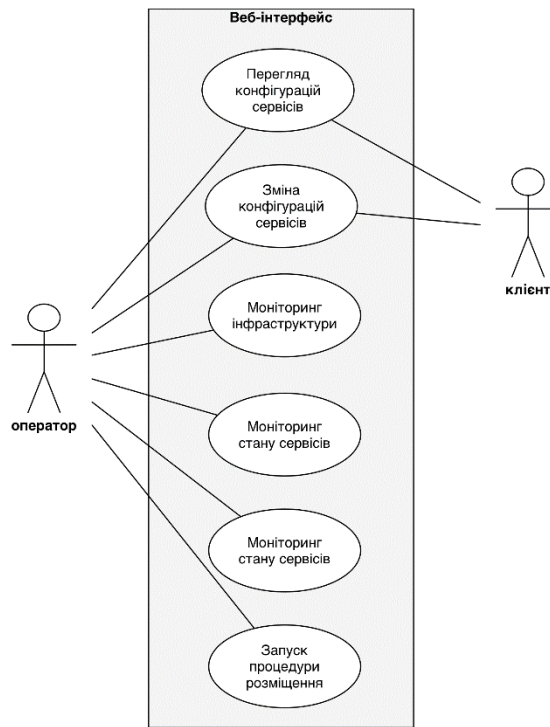


Рис. 6.9. Діаграма прецедентів (сценаріїв використання)

Також набір доступних операцій залежить від стану, в якому перебуває система. На рис. 6.10 зображена діаграма станів системи. У стані “узгоджено” усі сервіси розміщені таким чином, що вимоги кожного сервісу повністю задовільняються. При зміні вимог хоча б одного сервісу або при зміні конфігурації інфраструктури система переходить у стан “Оцінка”, в якому оператор здійснює якісну оцінку необхідності застосування нового розміщення. При підтвердженні оператором система переходить у стан “Розміщення”, в якому відбувається запуск, зупинка або фізичне переміщення компонентів інфраструктури. Після завершення усіх операцій, що приводять компоненти інфраструктури до нового стану, система переходить у стан “узгоджено” та процес повторюється знову.

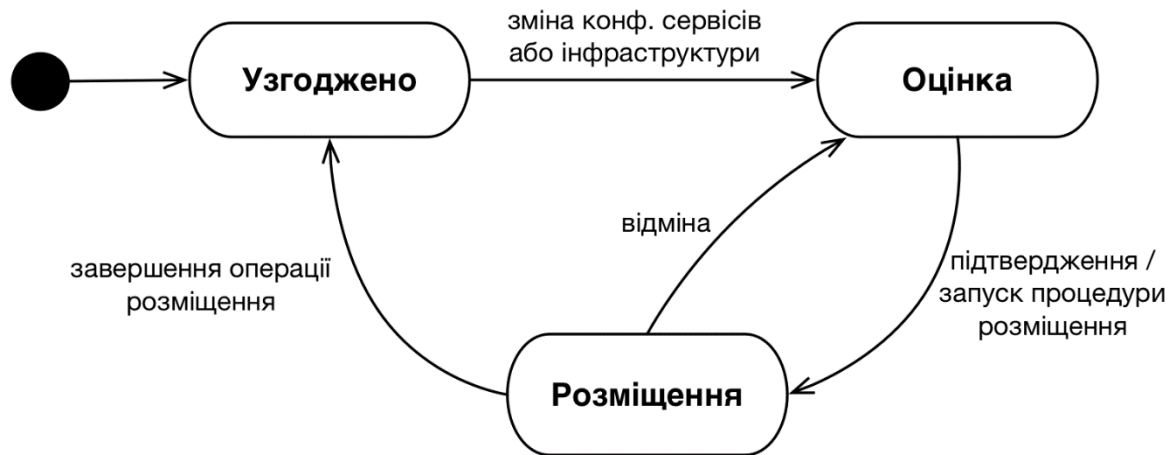


Рис. 6.10. Діаграма станів системи

6.4.1.1 Деталі реалізації

Компоненти системи виконані у вигляді окремих Python-пакетів. У контексті моделі трирівневої архітектури структура пакетів та їх залежності зображені на рис. 6.11:

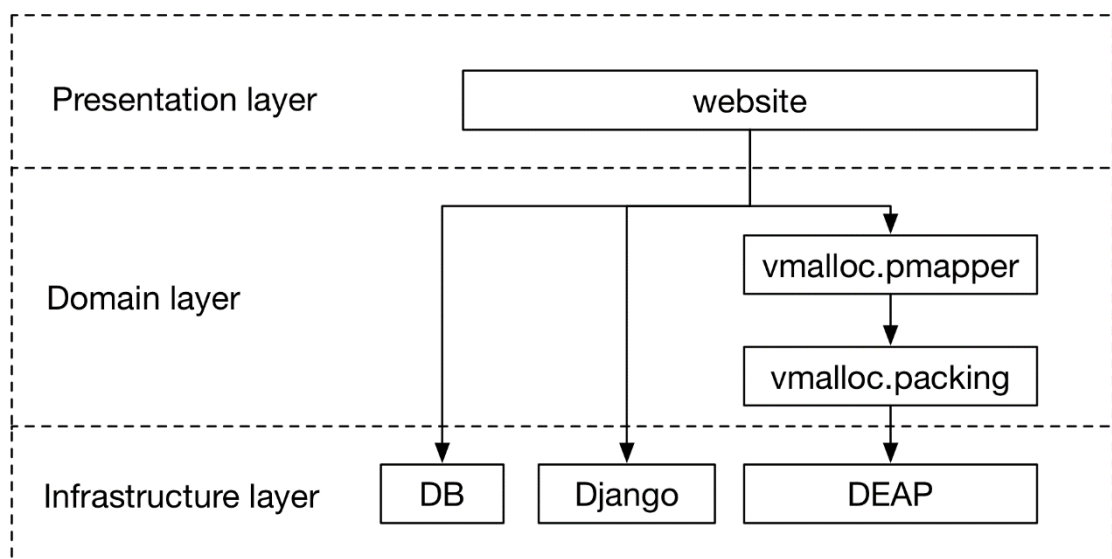


Рис. 6.11. Реалізація моделі трирівневої архітектури

До складу програмного забезпечення входять наступні Python-пакети:

- `vmalloc` – ядро системи, що розраховує оптимальне розміщення віртуальних машин та обчислювальних процесів критичної інформаційної інфраструктури. Пакет надає інтерфейс у вигляді чистих функцій, що приймають

та повертають незмінні структури даних (namedtuple). Не виконує будь-якого вигляду роботи із базами даних, зовнішніми системами або іншими statefull об'єктами;

- `vmalloc.packing` – універсальний модуль, що реалізує розв'язання задачі багатомірного пакування. До складу програмного інтерфейсу входять структури даних *Ресурс* (Resource), *Предмет* (Item), *Контейнер* (Container) та *Середовище* (Environment). Бібліотека, відповідно, розраховує оптимальне розміщення предметів серед контейнерів, що є найбільш оптимальним з точки зору використання ресурсів. Оскільки цей модуль є досить абстрактним (буквально нічого не знає про фізичні та віртуальні машини), використовувати його надзвичайно легко і для вирішення інших комбінаторних задач. Реалізує алгоритми генетичного програмування, метод рою часток та інші. Використовує потужний фреймворк DEAP [269] для реалізації еволюційних алгоритмів;

- `vmalloc.pmapper` (Process Mapper) – головний модуль, що відповідає за розрахунок розміщення компонентів критичної інформаційної інфраструктури. До складу програмного інтерфейсу входять структури даних для представлення ресурсів, фізичних та віртуальних машин, ЦОД та програмних процесів, що виконуються у них. Містить одну функцію `vmalloc.pmapper.allocate`, що для наданої конфігурації системи (ресурси, процеси, ЦОД) повертає об'єкт *Розміщення* (Placement), що описує оптимальне розміщення віртуальних машин серед елементів ЦОД, та оптимальне розміщення програмних процесів серед віртуальних машин. На обох рівнях використовує модуль рішення задачі багатомірного пакування. Діаграма класів модуля наведена на рис. 6.12.

- `website` – веб-інтерфейс платформи управління критичною інформаційною інфраструктурою, що реалізований за допомогою фреймворку Django [270]. Містить опис моделей бази даних, реалізацію інтерактивного веб-інтерфейсу та синхронізацію результатів, отриманих за допомогою `vmalloc.pmapper` та базою даних.

– website.exim – реалізовує завантаження та збереження конфігурацій систем у форматі JSON.

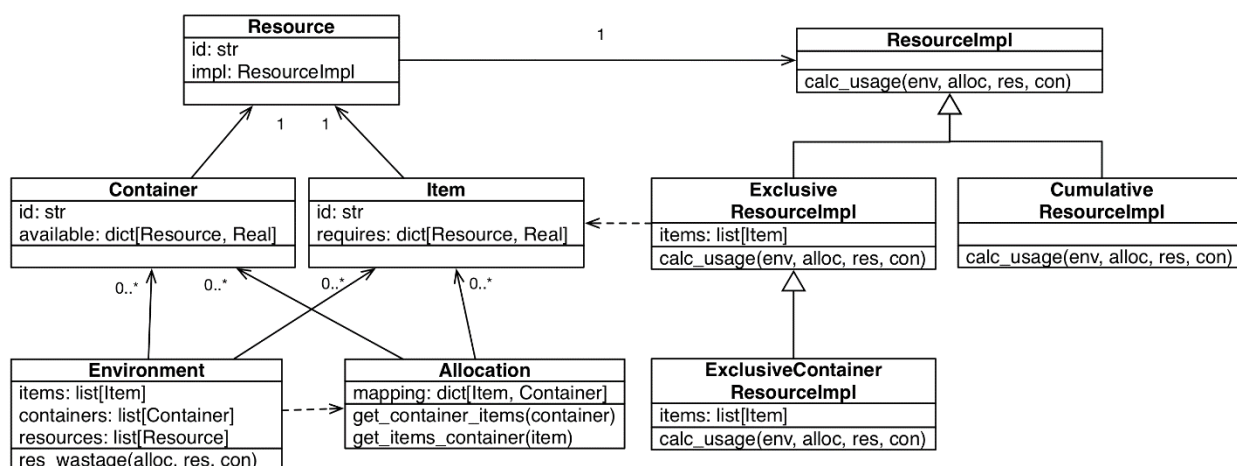


Рис. 6.12. Діаграма класів пакету vmalloc.pmapper

6.4.1.2 Опис інтерфейсу користувача

Взаємодія з інтерфейсом відбувається у контексті проекту. Кожен проект являє собою опис програмних процесів користувачів та доступної фізичної інфраструктури. На рис. 6.13 зображений приклад сторінки інтерфейсу, що використовується для налаштування проекту.

Configuration

Project name

Services

Name	Type	HA Status	Actions
Low priority service 1	Medium		<button>Configure</button>
Low priority service 2	Low		<button>Configure</button>
2-nodes	Low		<button>Configure</button>
4-nodes	Low		<button>Configure</button>
3-nodes	Low		<button>Configure</button>

Add service

Save

Рис. 6.13. Сторінка налаштування проекту

На рис. 6.14 зображено сторінку конфігурації програмного сервісу. Необхідно вказати назву процесу, рівень ізоляції кожного процесу, кількість

одночасно запущених реплік та вимоги до обчислювальних ресурсів. Рівень ізоляції вибирається в залежності від вимог користувача та визначає розміщення реплік відносно одна одної:

- рівень «critical» – репліки будуть розміщені у різних дата-центрах;
- рівень «high» – репліки будуть розміщені на різних фізичних машинах;
- рівень «medium» – репліки будуть розміщені на різних віртуальних машинах;
- рівень «low» – відсутність вимог до ізоляції.

The screenshot shows a web interface for configuring a service. At the top, there is a text input for 'Service name' containing 'Low priority service 1'. Below this is a 'Select isolation level' section with a horizontal slider ranging from 'low' to 'critical'. The slider is currently positioned at 'medium'. To the right of the slider is a button labeled 'Virtual Machine isolation'. Below the slider is a 'Scale factor' input field containing the value '2'. The 'Resource requirements' section contains a table with four rows: Network, RAM, CPU, and an empty row. Each row has a dropdown menu for the resource type, an input field for the amount, a unit, and a 'Delete?' checkbox.

Resource	Amount	Unit	Delete?
Network	0.5	GB/sec	☐
RAM	4.0	GB	☐
CPU	100.0	MIPS	☐
	None		☐

At the bottom of the form is a 'Save' button.

Рис. 6.14. Сторінка налаштування сервісу

Панель управління проекту надає змогу користувачу виконувати адміністративні дії щодо користувацьких сервісів, наприклад, ініціювати їх розміщення. Також існує можливість відслідкувати стан сервісів на поточний момент, кількість розміщених сервісів та реплік (процесів). Існує 2 режими розміщення: простий (використовуючи налаштування за замовчуванням) та складний (із заданням параметрів алгоритму розміщення на різних етапах). Приклад розміщення зображений на рис. 6.15.



Рис. 6.15. Панель управління проекту

Також користувачу надається можливість здійснювати моніторинг стану інфраструктури. На рис. 6.16 для кожної фізичної машини відображаються список розміщених на ній реплік сервісів, кількість розміщених віртуальних машин, наявний обсяг обчислювальних ресурсів та їх загальна утилізація.

У складному режимі виконання розміщення існує можливість вибрати тип та параметри алгоритму на двох етапах: під час розміщення серед фізичних машин інфраструктури, та під час розміщення серед віртуальних машин. На рис. 6.17 та 6.18 зображені приклади налаштування та результати роботи у вигляді графіків фітнес-функції.

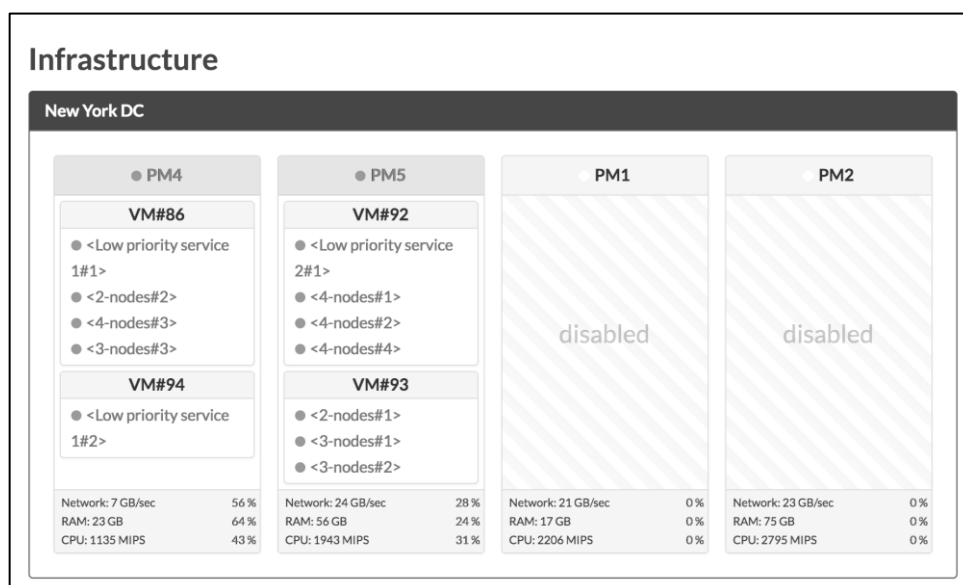


Рис. 6.16. Моніторинг стану інфраструктури

Per-Host strategy

Genetic Algorithm

Population size: 50
Number of generations: 100
Probability of mutating an individual: 0.2
Probability of mating two individuals: 0.5

Start

Per-VM strategy

Particle Swarm Optimization

Population size: 10
Number of generations: 50
Cognitive parameter (aka $C1, \Phi_c$): 2
Social parameter (aka $C2, \Phi_s$): 2

0

Рис. 6.17. Налаштування алгоритмів розміщення

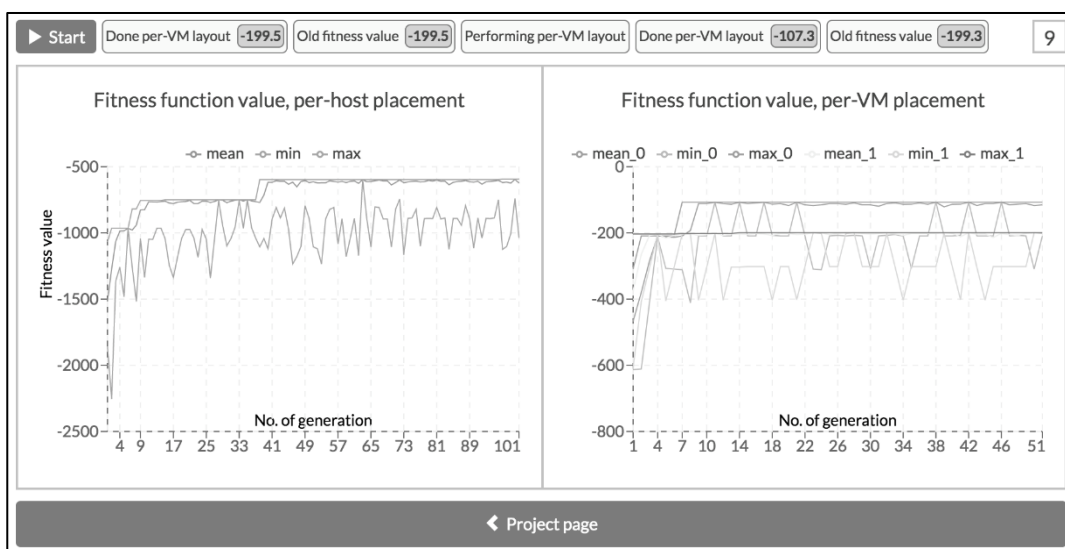


Рис. 6.18. Візуалізація роботи алгоритмів розміщення

Фізичним розгортанням системи відповідно до розрахунків розміщення та згенерованих файлів конфігурацій Dockerfile, Stackfile та Vagrant займається система автоматичного розгортання і управління критичної інформаційної інфраструктури (див. наступний розділ).

6.4.2 Система автоматичного розгортання і управління критичною інформаційною інфраструктурою

На рис. 6.19 представлена узагальнена схема системи автоматичного розгортання та управління критичною інформаційною інфраструктурою.

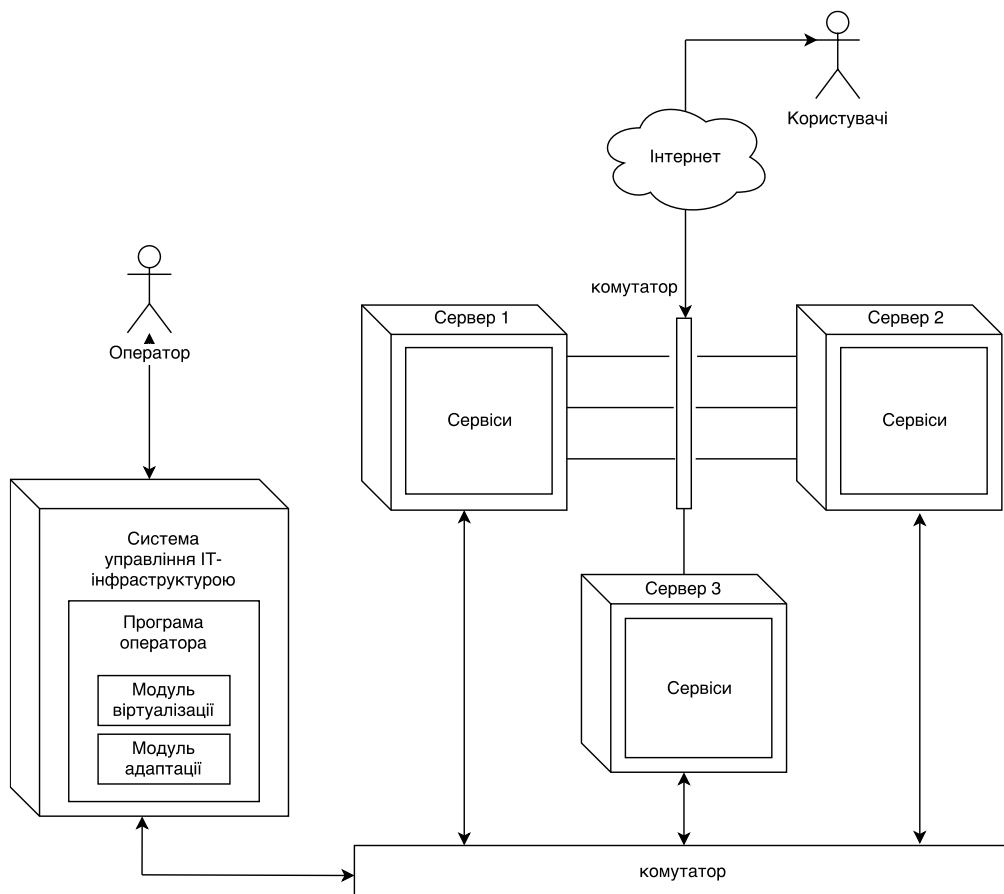


Рис. 6.19. Система автоматичного розгортання та управління критичної інформаційної інфраструктури

Основним модулем розроблюваної системи є автоматизоване робоче місце оператора (АРМ-О), який складається з модуля адаптації та модуля віртуалізації (vagrant).

Модуль адаптації робить запит на сервери із певним інтервалом з метою отримання даних про використання системних ресурсів.

Модуль віртуалізації використовується для створення віртуальних машин, на яких будуть розгорнуті сервіси, які є механізмом надання послуг для користувача через мережу Інтернет.

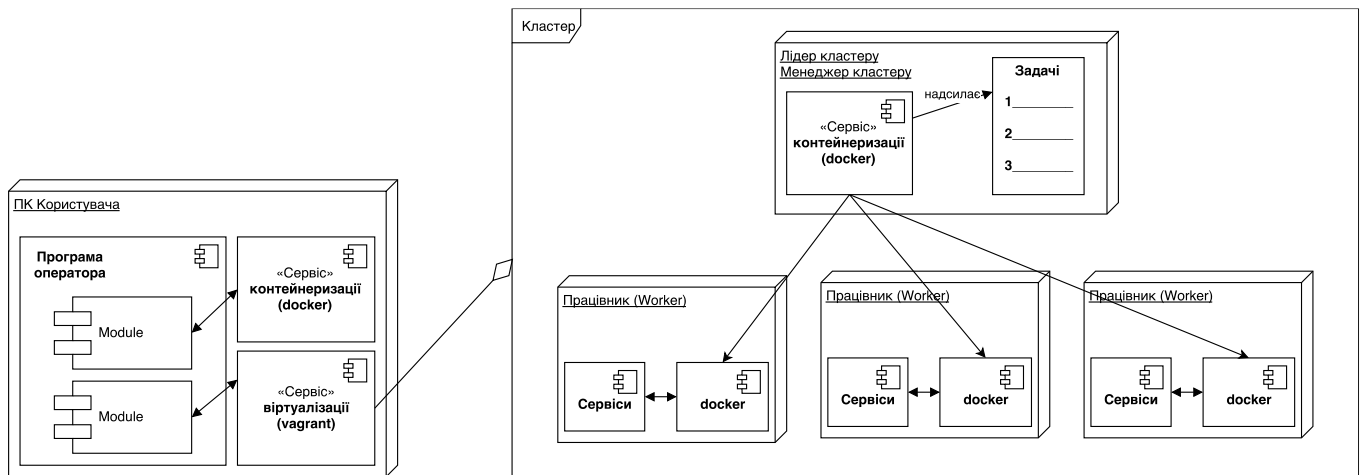


Рис. 6.20. Автоматичне розгортання критичної інформаційної інфраструктури

На рис. 6.20 представлено діаграму розгортання критичної інформаційної інфраструктури за допомогою АРМ-О.

АРМ-О реалізується у вигляді програмного модуля – клієнтського застосунку на базі компонентів для управління модулями контейнеризації (docker) та віртуалізації (vagrant) для запуску процесу створення серверної частини, всередині якої розгортаються сервіси та веб-застосунки. Серверна частина представлена у вигляді кластера за технологією Docker Swarm [271]. АРМ-О виконує команди сервісу контейнеризації для того, щоб загорнути існуючий проект у образ – створити файл з інструкцією для налаштування та розгортання такого проекту у вигляді контейнера – віртуальної структурної одиниці кластера, яка представляє собою атомарну віртуальну машину, в основі якої лежить операційна система Linux з необхідним набором пакетів для того, щоб запустити проект. В інструкції описано директорію, яка вважається кореневою для проекту, прописуються вхідні змінні та змінні середовища, які необхідні проекту для запуску. Вихідний код проекту завантажується всередині контейнера та проходить всі етапи налаштування.

Образ контейнера – дуже простий приклад сервісної архітектури. Більш наближеним до реальності буде кластер, який служить для запуску декількох сервісів, пов'язаних між собою логічно або асоціативно. Сервісом називається проект, який розгортається в середині кластера. На відміну від контейнерів,

сервіси також містять інструкції для визначення зв'язків з іншими сервісами, стратегію їх перезавантаження у випадку помилки і т.д.

Оператор запускає процес розгортання, клієнтська програма надсилає відповідну команду до модулю віртуалізації, який створює віртуальні машини за допомогою VSphere [272] та виконує ряд команд для того, щоб віртуальна машина мала всі необхідні компоненти для подальшого функціонування.

На діаграмі сценаріїв використання (рис. А.1) описані послідовності дій, які оператор може виконати для вирішення задач по управлінню та запуску процесу розгортання та адаптації критичної інформаційної інфраструктури.

За допомогою АРМ-О оператор може виконати наступні дії:

- авторизуватись у сервісі, який надає функціонал репозиторію для зберігання образів сервісів;
- контейнеризувати існуючий програмний застосунок, тобто на базі існуючого застосунку отримати файл з інструкцією по розгортанню контейнеру – віртуальної машини із сконфігурованим, готовим до роботи, не залежним від архітектури чи середовища розгортання сервісом. Всі налаштування та інструкції прописуються у спеціальному файлі – Dockerfile завдяки ряду команд, підтримуваних модулем контейнеризації;
- процес контейнеризації в свою чергу включає такі процеси, як побудова Dockerfile, перевірка його вмісту та процес побудови образу на його основі. Такий образ надалі можна зберегти у віддаленій репозиторій для легкого та швидкого поширення на інших датацентрах, де буде розгорнуто критичну інформаційну інфраструктуру. Підтримується можливість використання існуючого Dockerfile в якості шаблону. Таким чином, оператору потрібно лише відредагувати існуючі, або додати нові інструкції до файлу та перезаписати його. Всі ці процедури проходять автоматично та оператор виконує їх через наданий візуальний інтерфейс застосунку;
- для створених образів, оператор може в будь який час переглянути детальну інформацію про них;

- оператор може під'єднатись до приватного сервісу, який надає функціонал репозиторію, якщо такий існує. Для цього він повинен налаштувати програму оператора на роботу з таким сервісом шляхом зазначення шляху до сокету docker сервісу, або зазначенням адреси для TCP з'єднання;
- оператор може створити stackfile – файл з інструкціями для кожного сервісу, які разом будуть складати кластер. В такому файлі прописуються мережі для комунікації між сервісами в межах кластеру та за його межами, відкриті порти, резервується дисковий простір, який буде використаний сервісом для збереження даних та їх відновлення у разі його перезавантаження чи виходу з ладу, прописується порядок, в якому будуть розгорнуті сервіси, вказується залежність одного сервісу від іншого, режим запуску, кількість реплік, стратегія поведінки у разі виходу сервісу з ладу;
- оператор може переглядати список створених образів контейнерів, виконувати пошук по такому списку, переглядати детальну інформацію про будь який з них;
- видаляти образи вручну – видаляючи їх один за одним. В такому випадку можливе випадкове видалення образу, який ще буде використовуватись, або використовується одним із сервісів чи контейнерів, запущених на даний момент, тому можлива також опція видалення образів автоматично. В такому разі вони будуть перевірені на використання та видалені лише у разі підтвердження того, що такий образ чи така версія образу сервісами та контейнерами не використовується;
- встановлювати вхідні параметри для інфраструктури. Серед таких параметрів – кількість ядер, які будуть надані кожному із серверів в інфраструктурі, об'єм оперативної пам'яті та відсоток максимального навантаження на процесор. Також оператор може задавати коефіцієнти адаптації, які будуть впливати на частоту адаптації серверів під час їх дії. Чим вищий коефіцієнт, тим рідше сервер буде адаптовано до поточного стану сервісів на ньому. Серед доступних коефіцієнтів є відсоток навантаження на процесор та відсоток використаної оперативної системи;

– запускати процес розгортання інфраструктури, ставити на паузу сервери в складі стеку, відновлювати їх роботу, повністю видаляти розгорнуту критичну інформаційну інфраструктуру разом з файлами налаштувань та усіма даними про неї.

Систему реалізовано з урахуванням сучасного стеку технологій - на базі Node.js [273] з використанням фреймворку Electron [274] на базі ядра браузера Chromium [275], а також бібліотек для виконання команд для модулів контейнеризації – Docker [276] та віртуалізації – Vagrant [277].

Система підтримує операційну систему Linux, містить функціонал для підготовки файлів конфігурацій образів сервісів, yaml-мап розгортання кластеру, проводить пошук існуючих образів сервісів, автоматично запускає сервери у вигляді віртуальних машин на основі операційної системи Linux, об'єднує сервери у docker-кластер, розгортає критичну інформаційну інфраструктуру з наданого yaml файлу конфігурацій та збирає дані щодо використання системою поточних ресурсів серверу – оперативної пам'яті, ядер процесору, відсотку навантаження на процесор та виводить їх у вигляді графіків для їх зручного аналізу оператором.

Додатково система встановлює модуль Grafana [278] разом з сервісом Prometheus [279], який збирає статистику кластеру та відображає її у вигляді графіків.

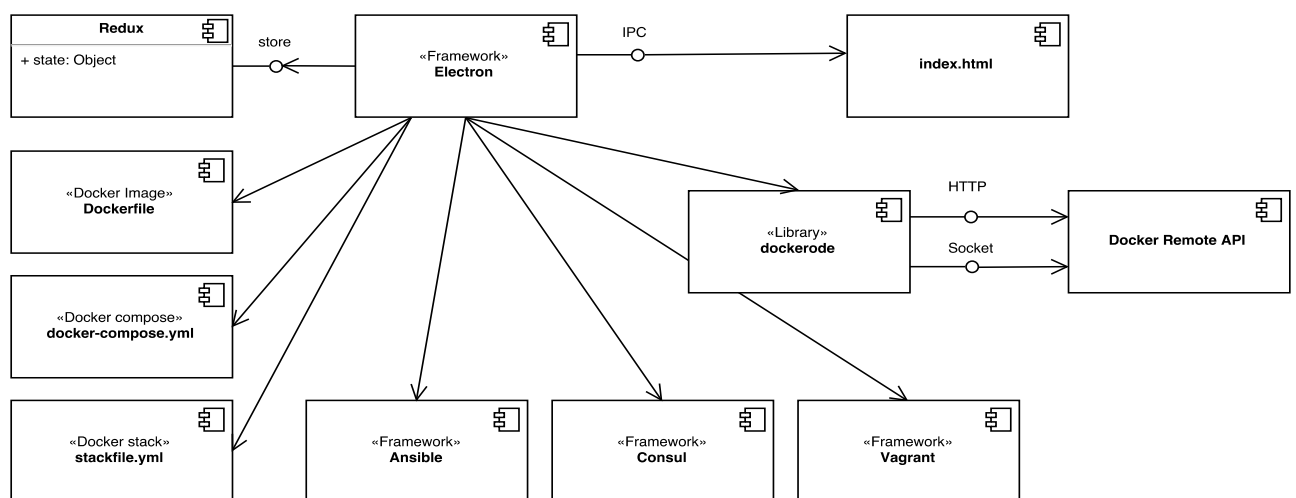


Рис. 6.21. Діаграма компонентів клієнтського застосунку

Діаграма компонентів клієнтського застосунку представлена на рис. 6.21 і складається з наступних модулів:

- `dockerode` – модуль для роботи з модулем контейнеризації. Оскільки робота з ним проводиться через HTTP запити до сервісу, даний модуль містить в собі готові функції для взаємодії по даному протоколу;
- `json-storage` – модуль для збереження даних користувача у вигляді json файлу. Використовується для збереження конфігурацій, даних авторизації;
- `react` – сучасний фреймворк для створення візуального інтерфейсу на основі компонентів;
- `redux` – модуль запам'ятовування стану `react` – компонентів;
- `grommet` – набір готових `react` компонентів візуального інтерфейсу на основі CSS Flexbox;
- `vagrant` – модуль виконання команд модулю віртуалізації у терміналі;
- `yamljs` – парсер `yml`, `yaml` файлів;
- `Docker Swarm` – засіб організації серверів у кластер;
- `Swarm Manager` – менеджер кластеру, відповідальний за розгортання сервісів та доступ до них через будь-який із серверів кластеру;
- `Swarm Worker` – сервер, що виконує команди менеджера;
- сервіси;
- контейнери;
- образи;
- мережі.

Робота АРМ-О розподілена на два потоки – потік, відповідальний за оновлення та виконання задач на візуальному рівні та основний потік, відповідальний за усі процеси, що не мають відношення до візуального інтерфейсу. Комунікація між двома потоками здійснюється по принципу, схожому до `socket` – через IPC канал.

В основі всього лежить компонент `Electron` – фреймворк для запуску веб-застосунків у вигляді програм для ПК та з підтримкою доступу до файлової

системи. В ньому є `index.html` файл, який є єдиним файлом і на якому і відбувається вся логіка візуальної частини застосунку.

Завдяки компоненту `dockerode` програма може використовувати функціонал, який надає модуль контейнеризації – `docker remote API`. Підключення до такого модулю може відбуватись або через `http`, або через сокет.

Компонент `Vagrant` відповідає за створення віртуальних машин та їх налаштування, запуск, зміну стану.

Компонент `Consul` [280] не було включено в дану систему за замовчуванням, проте його можна використовувати для полегшення управління ІТ-інфраструктурою, а також для підтримки функціоналу балансування навантаження в інфраструктурі.

Компонент `Ansible` [281], як і `Consul`, може бути додатково встановлений для полегшення процесу розгортання інфраструктури. З його допомогою можна зберігати ресурси у спеціальному сховищі та використовувати їх в нових проектах і не починати писати з нуля.

`Stackfile.yml` є компонентом, який містить налаштування стеку сервісів. Він є необхідним для розгортання інфраструктури та буде створений оператором або системним програмістом.

`Docker-compose.yml` є менш функціональним видом `stackfile`. Відмінністю є те, що такий файл дозволяє розгорнути стек сервісів в межах лише одного серверу.

`Dockerfile` є найменшим компонентом. Він містить інструкції для налаштування контейнеру та сервісу, який в ньому буде розгорнуто. Такий файл також вважається образом контейнера.

`Redux` [282] зберігає важливі дані в процесі роботи застосунку та тримає їх в оперативній пам'яті. Використання такого компоненту полегшує процес написання складних програм, адже централізує роботу з даними в одному місці та нав'язує ряд правил та специфікацій, описуючих процес зміни таких даних, створюючи таким чином гарантію однакового алгоритму їх зміни та полегшуючи розуміння коду.

6.5.2.1 Розробка клієнтської частини системи

Виходячи зі списку сценаріїв використання, створено АРМ-О. Робота застосунку розподілена на 2 потоки – візуальний та основний. Візуальний потік виконує код, пов'язаний з відображенням візуальних компонентів, які являють собою react-компоненти.

Технологія react надає можливість перетворити html код типового елемента візуального інтерфейсу на атомарний компонент, який має певну череду подій, функціонал, стан та параметри. Завдяки цій технології, зменшується кількість коду, відповідального за логіку та візуальний стиль компонентів, зменшує навантаження на процесор через розумний рендеринг компонентів (рис. 6.22) та замість перемалювання усієї сторінки, перемальовує лише ті компоненти, стан яких змінився. Саме в цьому потоці обробляються взаємодія користувача з візуальним інтерфейсом.

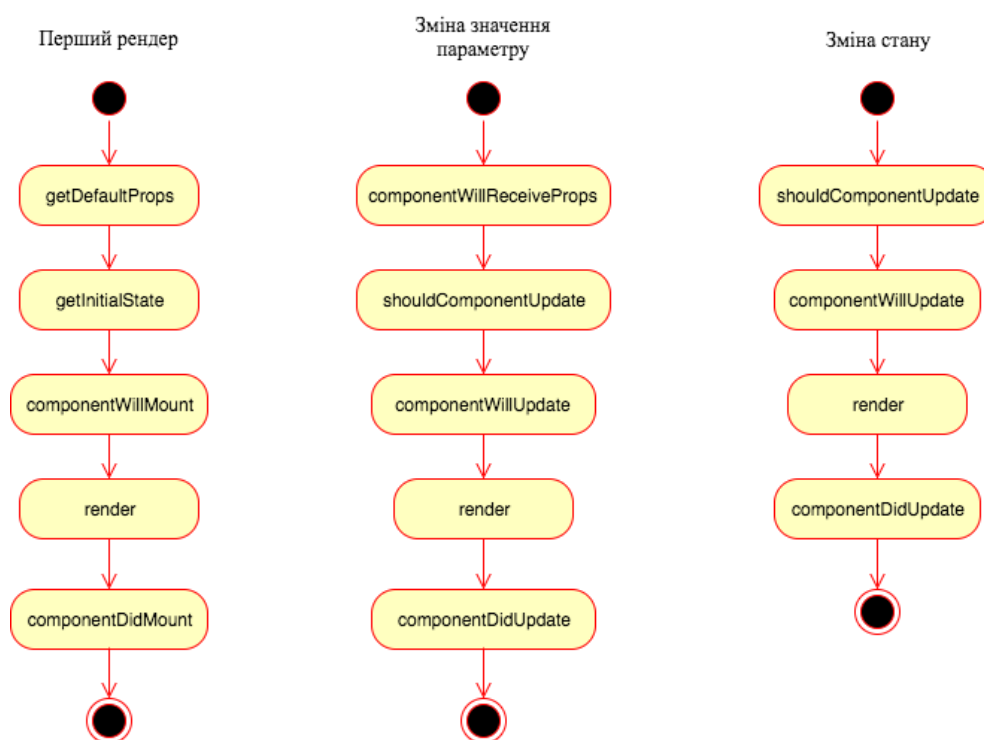


Рис. 6.22. Ланцюг подій React-компоненту

Основний потік відповідальний за створення файлів у файловій системі, зчитування, запис важливої інформації, яка необхідна програмі для відновлення стану під час наступного використання. Цей потік відповідальний за опрацювання запитів із візуального потоку, та виконання команд у терміналі.

В момент увімкнення, основний потік перевіряє останній стан кластеру, збережений у вигляді json-файлу, намагається відновити підключення до сервісу контейнеризації, намагається відновити підключення до віддаленого репозиторію файлів конфігурацій сервісу контейнеризації. Перевіряються налаштування та стан кластеру – чи запущений він, з якими параметрами запущений, запускаються таймери та надсилаються HTTP-запити до менеджера кластеру з метою отримання поточного стану серверів, об'єм використовуваних ресурсів, кількість контейнерів та сервісів, які було запущено. Вся ця інформація постачається до візуального потоку, де надалі і зберігається у Redux Store [283].

Redux [282] – технологія для роботи зі станом react компонентів. Вона тримає глобальний стан усього застосунку у вигляді одного незмінного об'єкту. React-компонент слідкує за зміною глобального стану та коли той змінюється – змінює свій власний стан та перемальовується відповідно до нього. Даний об'єкт може бути змінений лише шляхом посилання сигналу – певної події (Action). Кожна подія має унікальну назву-код, завдяки чому обробляється належним оброблювачем – редюсером.

Приклад дії, яка викликає появу графічного повідомлення для користувача:

```
this.store.dispatch({
  type: 'SHOW_NOTIFICATION',
  notificationMessage: 'Creating tarball...',
  notificationType: 'unknown',
  notificationProgress: 30
});
```

Кожний редюсер приймає події, перевіряє їх по унікальному ключу та якщо таку подію підтримує, створює новий стан застосунку, копіюючи шматок

глобального стану, за який такий редюсер є відповідальним, та змінюючи його з урахуванням параметрів, які візуальний потік надсилає разом із подією.

Приклад редюсера, який очікує подію `SHOW_NOTIFICATION` та змінює глобальний стан застосунку:

```
// початковий стан
const initialState = {
  sidebarOpen: true,
  showNotification: false,
  notificationMessage: "",
  notificationType: "",
  notificationProgress: null
};

// функція редюсера
module.exports = (state = initialState, action) => {
  switch (action.type) {
    case 'TOGGLE_SIDEBAR': { // подія
      return Object.assign({}, state, { sidebarOpen: !state.sidebarOpen });
    }
    case 'SHOW_NOTIFICATION': {
      // зміна стану
      return Object.assign({}, state, _.omit(action, 'type'), { showNotification: true
});
    }
    case 'HIDE_NOTIFICATION': {
      return {
        sidebarOpen: state.sidebarOpen,
        showNotification: false,
        notificationMessage: "",
        notificationType: "",
        notificationProgress: null
      }
    }
  }
};
```

```

    };
  }
  default: {
    return state;
  }
}
};

```

Щоб перемалювати візуальний елемент, наприклад, текстове поле, візуальний потік надсилає відповідну подію, включає нове значення у тіло об'єкта, редюсер розпізнає таку подію, змінює стан, компонент отримує сигнал про зміну стану, порівнює власний стан із глобальним, синхронізує власний стан та перемальовується (рис. 6.23). Таким чином, клієнтська програма не перемальовує сторінку, вона лише перераховує стан компонентів та адаптує сторінку, а також кожний компонент функціонує окремо та знає в який момент йому з'являться, а коли зникати та має доступ до актуального глобального стану застосунку.

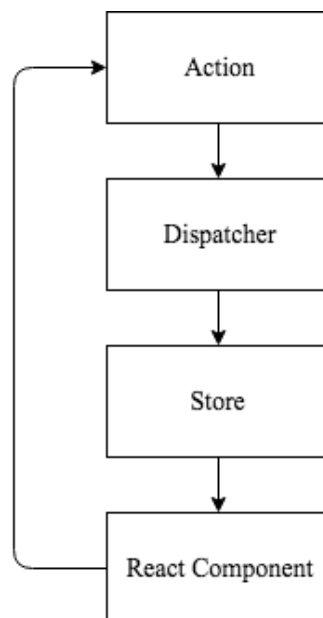


Рис. 6.23. Алгоритм роботи React-Redux архітектури

Для зменшення затрат часу на розробку кожного react-компоненту окремо, в програмі оператора було використано набір готових компонентів – Grommet [284]. Логіка розміщення компонентів побудована на CSS Flexbox [285] – новому підході до адаптивного розміщення компонентів на формі.

Завданням візуального потоку застосунку є обробка взаємодії користувача, зміна глобального стану застосунку, очікування на зміну такого стану та синхронізація, перемалювання компоненту згідно з новим станом. Всі дані, які необхідні для поточної роботи застосунку зберігаються у Redux Store – на кшталт бази даних, що зберігається у оперативній пам'яті.

Такі дані, як, наприклад, статистика віддаленого серверу, неможливо отримати від користувача, тому в такій ситуації необхідно задіяти основний потік (main process) та його працівники.

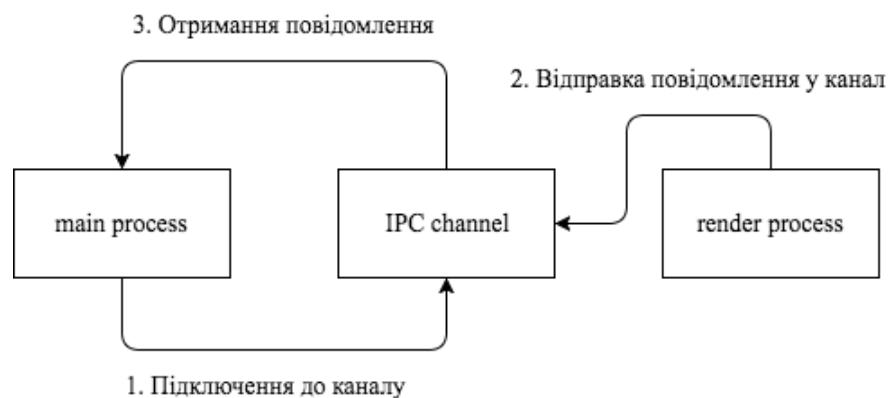


Рис. 6.24. Комунікація двох потоків через IPC-канал

Основний процес одразу ж після старту підключається до IPC-каналу (рис. 6.24) та слухає на повідомлення. Кожне повідомлення ідентифікується за унікальною назвою. В даній застосунку вирішено дотримуватись наступного шаблону подій: [назва] – для повідомлення-запиту, [назва]:rs – для повідомлення-відповіді.

Основний процес запускає декілька працівників – класів, відповідальних за виконання певних дій з певним інтервалом. Один із таких класів відповідальний за оновлення стану модуля контейнеризації, інший – за оновлення стану критичної інформаційної інфраструктури.

Працівник, відповідальний за оновлення стану засобу контейнеризації, використовує модуль `dockerode`, перевіряє статус сервісу – працює він чи ні. Якщо не працює – необхідні компоненти це відслідковують та повідомляють оператора про те, що сервіс не працює і програма не може визначити його стан. Якщо ж сервіс працює в нормальному режимі, проводиться серія запитів до нього та отримуються дані про доступні ресурси та контейнери. Інший працівник здійснює ряд HTTP-запитів на адресу менеджера кластеру з метою отримати дані про стан кластеру та його компонентів. Отримані дані пересилаються по IPC-каналу до візуального потоку.

Основною особливістю IPC-каналу на відміну від `socket`-у є те, що повідомлення, коли потрапляє до каналу, в якому немає отримувача, не може зникнути та чекає на того, хто його обробить.

Перш ніж програма зможе дізнатись про стан сервісу `docker`, вона повинна спочатку підключитись до нього або через сокет, або через посилання. В програмі оператора пріоритет наданий підключенню через сокет, проте є змога налаштувати підключення по посиланню. Для цього оператору лише потрібно вказати правильне посилання та порт. Після перевірки на підключення та сигналюванні про успішне підключення, програма почне збирати дані про статус сервісу та його компонентів. Ці дані будуть використані у випадку, якщо оператор або інший користувач використовуватимуть програму для генерації файлів налаштування образів та `uml` мап розгортання кластеру.

Отже, клієнтську частину можна зобразити наступним чином у вигляді діаграми класів, зображену на рисунку А.2.

Як видно із діаграми (рис. А.2), процеси в потоці рендерингу складаються здебільшого із `react`-компонентів. Цей потік відповідає за обробку взаємодії користувача із програмою. Кожен із компонентів є одним із візуальних елементів. Він містить в собі унікальні оброблювачі подій, певний стан та параметри, проте кожний із таких компонентів, щоб мати можливість називатись компонентом, повинен наслідуватись від класу `React.Component`. Такий клас

містить в собі логіку обробки зміни стану, стандартні оброблювачі подій, а також метод `render`, який відповідний за відображення компонента у вигляді `html`-коду.

Кожний із компонентів наслідується від такого класу та містить власну реалізацію тих оброблювачів подій, які необхідні для його роботи. Окрім цього, кожен компонент має посилання на об'єкт подій, що є розподіленням логіки застосунку. Всього є три основних об'єкти подій, які розподіляють між собою функціональну логіку застосунку.

`DockerAction` відповідає за операції над образами, сервісами, контейнерами, стеками, оновлення інформації про стан докер сервісу, підключення та авторизацію в репозиторії.

`ImageAction` відповідає за операції над образами – відображення, вилучення більш детальної інформації, видалення, пошук так інші.

`VagrantAction` – відповідає за аналіз стану ІТ-інфраструктури, запущеної за допомогою модуля віртуалізації (`Vagrant`).

Кожен оброблювач подій викликає події з оновлення `redux`-стану – глобального стану застосунку. Таким чином, виділено логіку зміни глобального стану в окремий список класів. Також класи оброблювачів подій мають можливість зберігати дані у вигляді `json`. Такі дані зберігаються навіть після завершення роботи з програмою, проте потребують більше часу на читання/запис, аніж `redux state`, тому що доступні лише через файлову систему та через це обмежені швидкістю відгуку системи.

Коли оброблювач подій повинен зберегти щось у `redux state`, він визиває певну подію, яка буде оброблена в редюсері. Редюсер в свою чергу змінює глобальний стан застосунку та зберігає його у вигляді `Redux state`. `Redux state` також грає роль бази даних, проте знаходиться вона увесь час в оперативній пам'яті, а тому є швидшою за `json` базу даних. Призначення `redux state` – збереження даних під час роботи для відновлення інформації на сторінці, коли користувач повертається на неї, а також зберігання мінімуму даних, необхідних програмі для стабільної роботи.

В основному процесі є модуль для роботи з модулем контейнеризації та працівник, який опитує такий сервіс та повертає об'єкт з теперішнім його станом. Оброблювачі подій в візуальному потоці підписані на зміну стану в основному процесі і таким чином відбувається обмін даних.

Так само це відбувається і для відпрацювання виклику контекстного меню. Оскільки його обробка повинна виконуватись в основному процесі через специфіку роботи модулю. Такий модуль було використано через недостачу альтернативних рішень, а також для економії часу на обробці та концентрації на функціональній частині, а не візуальній.

На діаграмі потоку (рис. А.3) зображено потік даних в межах клієнтського застосунку для кожної із можливих дій оператора.

Як видно з діаграми (рис. А.3), АРМ-О має два основні сховища даних – базу даних в форматі json та redux store. Перша зберігає дані навіть після завершення роботи програми. Інша ж зберігає дані в оперативній пам'яті. Таким даними зазвичай є дані, які користувач надавав через інтерфейс програми під час роботи з нею, стан модуля контейнеризації, стан критичної інформаційної інфраструктури.

Після кожного запиту до сервісу модуля контейнеризації, дані зберігаються до redux store, адже необхідності в їх постійному збереженні немає та від них залежать лише візуальні компоненти, які будуть відрисовані у вікні оператора.

В базу даних постійного збереження записуються дані лише після авторизації користувача та зміни конфігурації docker. Після авторизації зберігаються такі дані, як ім'я користувача, зашифрований пароль та результат авторизації. В результат зміни конфігурації докер, вона також записується до бази даних, а саме – ір-адреса сервісу, порт, тип підключення – сокет чи TCP, шлях підключення (у випадку сокету). Такі дані при наступному завантаженні програми автоматично зчитуються з бази даних та проводять автоматичну авторизацію та встановлення зв'язку з докер сервісом. Таким чином, користувачу не потрібно виконувати такі операції кожний раз. Як зображено на

діаграмі, перед тим, як ці дані зберегти, проводиться запит до API модуля контейнеризації для перевірки на їх валідність. У випадку авторизації – на правильність наданих даних. У результаті конфігурації – можливість підключення.

Якщо оператор створює Dockerfile, результатом такої операції стає створення нового файлу. Далі користувач може перевірити такий файл на правильність структури та синтаксису, а також на наявність невірних інструкцій чи нестачу прав користувача. Наприклад, у випадку запиту на отримання приватного образу без попередньої авторизації. Такий Dockerfile надсилається до API засобу контейнеризації і в результаті створюється образ контейнеру, або виводиться повідомлення про помилку.

Такий образ далі використовується для створення сервісів у Stackfile. Він також перевіряється на синтаксис, проте лише завдяки парсеру .yaml. У випадку зі Stackfile, перевірити його формат розгортанням інфраструктури на цьому етапі неможливо через те, що для цього потрібна вже запущена та сконфігурована інфраструктура. Такий Stackfile надалі приймає участь у розгортанні інфраструктури.

В процесі розгортання критичної інформаційної інфраструктури за допомогою модуля віртуалізації, оператор задає певні налаштування для початкового стану критичної інформаційної інфраструктури та вказує доступний об'єм таких системних ресурсів. Також, паралельно, запускається процес перевірки стану критичної інформаційної інфраструктури та проводить аналіз на необхідність її адаптації.

Після створення Vagrantfile конфігурації, він стає відправною точкою для створення та розгортання серверів. З його інструкцій створюються віртуальні машини на основі вказаного образу. Конфігуруються згідно наданих ресурсів, налаштовуються мережі, далі на готові сервери завантажуються модулі докеру, створюється кластер, генерується ключ для підключення серверу до кластеру, записуються IP адреси серверів у hosts файл, завантажуються Stackfile та проводить його запуск. Під час запуску, відбувається комунікація із API модуля

контейнеризації, завантажуються необхідні образи, створюються мережі, резервується дисковий простір та налаштовується комунікація із сервісами.

Якщо після запуску критичної інформаційної інфраструктури та її розгортання, в процесі перевірки стану буде помічено можливість адаптації сервісу згідно рекомендацій від модуля адаптації, запуститься процес конфігурації. Вхідними даними є рекомендації із модуля адаптації. Як результат, створюється нова Vagrantfile конфігурація, сервер перезавантажується командою до модуля віртуалізації та в момент запуску, він читає вже новий, модифікований Vagrantfile та налаштовується вже згідно з новою конфігурацією та з новим об'ємом наданих системних ресурсів.

Оператор також може запустити модуль з оновлення інформації про стан модуля контейнеризації. До інформації, яку такий модуль збирає, належать:

- кількість запущених контейнерів;
- кількість образів контейнерів;
- кількість сервісів;
- кількість задач докеру;
- кількість серверів у кластері;
- стан докер сервісу.

Останньою дією, можливою з точки зору оператора є редагування списку доступних образів. Оператор може переглянути весь список образів. При цьому відбувається запит до API модуля контейнеризації та як результат отримується список усіх образів. Далі такий список фільтрується, прибираються порожні та системні образи і оператору відображається лише список доступних. Він може переглянути детальну інформацію про вибраний образ, а також видалити його на тій самій сторінці. Також, альтернативно, оператор може видалити образ не переглядаючи інформацію про нього. В обох випадках вхідними даними є назва образу.

В момент видалення відбувається запит до API модуля контейнеризації та надсилається параметр примусу – для безпрецедентного видалення не зважаючи

на те, чи використовується образ на якому-небудь із запущених сервісів чи контейнерів чи ні.

В процесі роботи застосунок постійно змінює свій стан. Як саме відбувається така зміна та від чого вона залежить відображено на рис. А.4.

Починається все із запуску застосунку. В такий момент його стан є невизначеним. Проводиться перевірка стану docker-сервісу. При цьому грає роль наявність файлів в базі даних про попередні підключення, конфігурація, шлях інсталяції docker-сервісу, наявність підключення Інтернет та інші. В результаті перевірки, docker-сервіс повертає інформацію про підключення чи завершується помилкою та не надсилає ніякого результату. У першому випадку, docker-сервіс вважається запущеним та програма переходить у стан «Працює», що означає, що вона запущена та готова до виконання.

Інакше, якщо жодного результату отримано не було, програма встановлюється у стан «Не працює», що означає, що вона не готова до виконання та сервіс docker не працює. В такому випадку, програма з певним інтервалом робить повторні запити на отримання стану сервісу доки оператор не завершить роботу з програмою або docker-сервіс не буде розпізнано.

Незалежно від стану docker-сервісу, оператор може спробувати авторизуватись у сервісі-репозиторії. В результаті, проходить перевірка стану системи. Якщо програма знаходиться в стані «Працює», робиться запит та перевіряються надані дані для авторизації в обліковому записі оператора. У разі підтвердження правильності даних, програма переходить у стан «Авторизовано», що означає, що оператор успішно отримав необхідні дані для роботи з приватними репозиторіями образів docker-контейнерів.

У випадку, коли результатом запиту на перевірку даних є помилка, авторизація оператора була проведена невдало та програма переводиться у стан «Не авторизовано», що означає, що авторизацію проведено не було

У випадку, якщо програма знаходиться в стані «Не працює», запит на авторизацію не проводиться та програма одразу переходить у стан «Не авторизовано».

Після перевірки авторизації, в залежності від вибору оператором наступної дії – розгортання інфраструктури чи побудова нового образу, відбувається наступний алгоритм зміни стану. У випадку побудови нового образу, перевіряється чи шаблон, на базі якого буде створено образ, є приватним. Якщо результатом перевірки є те, що образ не є приватним, оператор може без перешкод побудувати новий образ. Після побудови програма переходить у стан «Чекати на дію», що означає, що вона очікує нових дій від користувача.

У випадку ж, коли образ визнано приватним, відбувається перевірка на авторизацію користувачем. Якщо авторизацію проведено не було, перевірка завершиться помилкою та застосунок перейде в стан «Чекати на дію» та буде очікувати на подальші дії з боку оператора. Інакше ж, якщо авторизацію було проведено, оператор без перешкод зможе побудувати новий образ. Після побудови, система переходить в стан «Чекати на дію».

У випадку, коли оператор обирає перевірку стану критичної інформаційної інфраструктури, відбувається запит до відповідного сервісу. Якщо сервіс не запущено – запит завершиться помилкою, тому стан критичної інформаційної інфраструктури вважається «Зупинено. При цьому можливий також сценарій, коли критичну інформаційну інфраструктуру було поставлено на паузу. Для цього здійснюється також перевірка на наявність Vagrantfile. Якщо файл існує – значить критична інформаційна інфраструктура знаходиться в стані «В стані паузи». Інакше, якщо файл видалено, застосунок переходить в стан «Чекати на дію».

Якщо ж в результаті перевірки сервісу критичної інформаційної інфраструктури, замість помилки отримано повідомлення ОК, застосунок переходить у стан «Запущено». Із цього стану застосунок може перейти до стану зупинки («Зупинено») та в стан паузи («В стані паузи»).

В першому випадку, запускається процес зупинки критичної інформаційної інфраструктури. Після цього, видаляється Vagrantfile. Програма переходить в стан «Зупинено».

В іншому випадку, запускається процес заморожування стану критичної інформаційної інфраструктури. При цьому всі дані з оперативної пам'яті переносяться на дисковий простір у вигляді снєпшоту образу. Після завершення процесу, застосунок переходить у стан «Зупинено». Зі стану «Зупинено» вона може перейти до стану «Запущено», якщо оператор запустить процес «Зняти паузу» та перевірка стану критичної інформаційної інфраструктури завершиться успіхом.

В результаті застосунок переходить до стану «Чекати на дію» та завершується оператором як результат закінчення роботи з застосунком.

6.4.2.2 Розробка серверної частини системи

Серверна частина представлена у вигляді віртуальних машин, об'єднаних у кластер за допомогою використання технології Docker Swarm.

Серверна інфраструктура представлена у вигляді Entity-Relationship діаграми на рис. 6.25.

Кожний користувач повинен мати обліковий запис у сервісі, який надає функціонал репозиторію, тому зв'язок між User та Registry один до одного.

Кожний сервіс будується на основі одного образу і лише одного. Проте один и той самий образ може бути використаний для створення багатьох сервісів. Тому, зв'язок між сервісами та образами багато до одного.

Сервіс також має лише одну конфігурацію. Конфігурація не може бути поширена між сервісами, тому зв'язок між сервісом та конфігурацією завжди один до одного.

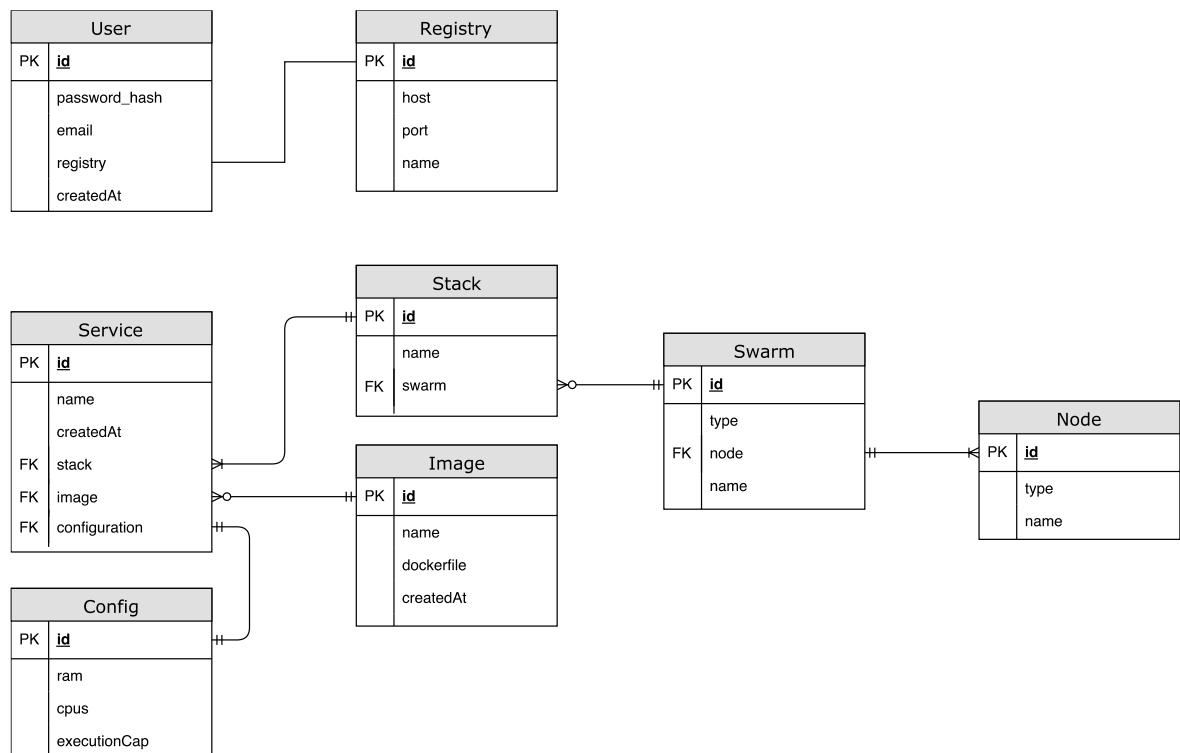


Рис. 6.25. Серверна інфраструктура у вигляді Entity-Relationship діаграми

Сервіс завжди належить до одного кластеру. Проте один и той самий кластер складається із багатьох сервісів. Тому, зв'язок між сервісом та кластером – багато до одного.

Один кластер складається з багатьох серверів. Але сервер не може бути включений у склад багатьох кластерів. Зв'язок між кластером та сервером – багато до одного.

Розглянемо частину алгоритму автоматичного розгортання – діаграму A0, представлену на рис. 6.26. На вхід надходить готовий проект – веб чи програмний застосунок, який необхідно представити у вигляді docker-контейнеру.

собою .yaml мапу розгортання кластеру. В разі помилки, оператору необхідно буде її виправити та побудувати новий файл.

Коли стекфайл готовий, оператор запускає процес розгортання. Вхідними параметрами є конфігурація, яку оператор задає на відповідній формі, а також коефіцієнти адаптації. В той самий момент запускається процес моніторингу програмою оператора стану критичної інформаційної інфраструктури. Створюється Vagrantfile, який заповнюється згідно з вхідною конфігурацією та за допомогою модуля віртуалізації (Vagrant) та VSphere створюється, налаштовується та розгортається інфраструктура, створюється docker-кластер та до нього під'єднуються усі сервери. Далі на серверах визначаються їх ролі в кластері та встановлюються сервіси.

Програма оператора з певним інтервалом збирає дані про стан інфраструктури. Якщо в результаті аналізу вирішено провести адаптацію серверів, вони перезавантажуються, Vagrantfile редагується відповідно до необхідних змін та сервери перезавантажуються, проте оскільки вони вже зконфігуровані та містять в собі певні сервіси та програми, вони проходять лише частину стандартного шляху розгортання – налаштування параметрів віртуальної машини та запуск вже існуючих сервісів.

На рис. А.5 представлено діаграму із зображенням архітектуру системи розгортання критичної інформаційної інфраструктури.

Отже, АРМ-О є лише відправною точкою для створення налаштувань та контейнеризації сервісу. Її задачею є створення одного з виду файлів конфігурацій:

- docker-compose.yaml;
- Dockerfile;
- stackfile.yaml.

Далі такий файл буде розгорнуто на сервері, що входить до складу кластеру. В складі кластеру є один лідер. Саме через такий сервер пересилаються команди до всіх інших серверів-працівників, які належать кластеру. Кластер може містити декілька менеджерів – серверів, які можуть стати заміною лідеру,

якщо той відмовить в роботі або виникне помилка. Менеджери також передають команди до працівників, яких в свою чергу може бути декілька, а може і не бути зовсім. Задачею працівників є виконання команд менеджерів.

Як видно із діаграми на рис. А.5, усі сервери працюють на основі операційної системи Linux та мають в своєму складі встановлений сервіс docker.

Результатом розгортання мікросервісної архітектури програми стане те, що доступ до неї може бути здійснено через будь-який сервер, який включено в кластер. Це дуже зручно, якщо необхідно зробити балансування навантаження відповідно до локації користувача. Все, що необхідно в такому випадку зробити – додати до кластеру новий сервер і змінити ДНС конфігурацію для пропуску користувачів лише з певного регіону. Такий сервер моментально стане доступний і через нього можна буде звертатись до будь-якого сервісу з інфраструктури.

6.4.2.3 Розробка функціоналу для створення образу контейнера

Файл образу контейнеру, інакше кажучи, Dockerfile – це текстовий файл з командними інструкціями, які розробник чи системний адміністратор мали б ввести вручну, щоб створити Docker Image – образ контейнеру. Таким чином, один раз описавши їх у вигляді файлу, розробник звільняє себе від задачі прописувати команди кожний раз як потрібно оновити Docker Image.

Дана програма містить інтерфейс, що дозволить оператору швидко створити Dockerfile, а з натисканням однієї кнопки – і Docker Image. Якщо ж Dockerfile в проекті вже існує, він буде використаний як шаблон. Його можна швидко змінити та замінити на новий. Процес створення файлу з образом контейнеру за допомогою інструментарію даної системи описаний в Додатку Б. В рамках дисертаційної роботи відповідні файли створюються автоматично керованою моделлю системою автоматичного розподілу ресурсів критичної інформаційної інфраструктури (див. розділ 6.5.1) .

6.4.2.4 Розробка функціоналу для створення Docker Stackfile

У разі, коли сервери об'єднані у кластер, створюють stackfile – yaml-мапу розгортання кластеру сервісів. Такий файл містить в собі детальну конфігурацію кожного окремого сервісу [286]:

- образу, на базі якого буде створено сервіс;
- змінні середовища;
- команда, яку буде виконано перед запуском сервісу;
- параметри конфігурацій;
- alias для імені сервісу;
- кількість реплік;
- стратегія перезавантаження у разі помилки;
- використання мереж для комунікації між сервісами в кластері;
- використання дискового простору для зберігання даних;
- порядок запуску серверів у кластері;
- зв'язок між сервісами у кластері;
- відкриті порти;
- драйвери та тип мереж;
- шлях до каталогу, куди будуть записуватись логи сервісу.

Процес створення yaml-мапи розгортання кластеру сервісів за допомогою інструментарію даної системи описаний в Додатку Б. В рамках дисертаційної роботи відповідні файли створюються автоматично керованою моделлю системою автоматичного розподілу ресурсів критичної інформаційної інфраструктури (див. розділ 6.4.1) .

6.4.2.5 Розробка інтерфейсу для роботи з Vagrant-сервісом

Vagrant – інструмент для конфігурації та запуску віртуальних машин. З точки зору конфігурації, він діє майже однаково з docker.

Основним файлом конфігурації є Vagrantfile. Синтаксис, на відміну від Stackfile – не yaml, а ruby.

Процес створення файлу конфігурації за допомогою інструментарію даної системи описаний в Додатку Б. В рамках дисертаційної роботи відповідні файли створюються автоматично керованою моделлю системою автоматичного розподілу ресурсів критичної інформаційної інфраструктури (див. розділ 6.4.1).

6.4.2.6 Розробка функціоналу модуля адаптації інфраструктури

Перед розгортанням інфраструктури та під час її роботи, оператор має змогу змінювати коефіцієнти адаптації. Зміна відбувається на панелі запуску процесу розгортання у вигляді бігунків – по одному для кожного регулюючого параметру – відсоток навантаження на ядро процесору та відсоток використаної оперативної пам'яті (рис. 6.27).

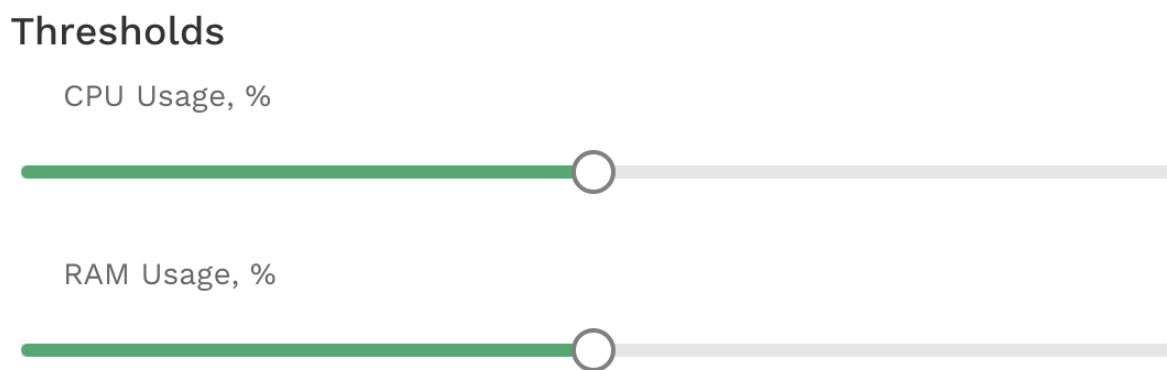


Рис. 6.27. Елемент регулювання коефіцієнтів адаптації

Логіка визначення моменту адаптивності системи до зміни процесорних ресурсів віртуальної машини є наступною. На вхід функції подається останнє значення, отримане від Grafana – модуля відображення даних. Серед них – відсоток навантаження на процесор та кількість задіяних ядер. Також, в якості

інших аргументів – коефіцієнт адаптивності навантаження процесору та вхідні дані – налаштування системи під час її першого запуску.

Далі перевіряється доступність кожного із прилеглих серверів, щоб бути впевненим у тому, що сервер не перезавантажується і дані, отримані від нього є актуальними. Якщо об'єм використовуваних даних на усіх підлеглих серверах є більшим за об'єм визначений коефіцієнтом адаптивності, перевіряється відсоток максимального навантаження на сервер. Якщо він менше за 100 відсотків, його буде підвищено. Інакше, перевіряється кількість доступних процесорів та у випадку, коли ще не всі ядра серверу використовуються, додається додаткове ядро.

У випадку, коли об'єм використовуваних даних є завеликим та необхідно його зменшити, процес перевірок є зворотнім. Перевіряється кількість ядер на кожному з серверів. Якщо вона більша за одиницю, віднімається одне ядро. Інакше, порівнюють максимальний відсоток навантаження на процесор. У випадку, якщо він є більшим за 10%, його зменшують, проте він не може бути нижчим за 10%, адже в такому випадку робота системи на сервері буде не якісною.

Відображення описаної логіки мовою Javascript ES6 приведено нижче:

```
shouldUpdateCPUUsage({ usage, cores }, threshold, config) {  
  const recommendedChanges = {};  
  // if one of nodes rebooting  
  if (!usage.worker1 || !usage.worker2) return recommendedChanges;  
  if (usage.worker1 >= threshold && usage.worker2 >= threshold) {  
    if (config.executionCap < 100) {  
      recommendedChanges.executionCap = config.executionCap + 20 > 100 ?  
100 : config.executionCap + 20;  
    } else if (cores.worker1 === cores.worker2 && cores.worker1 <  
os.cpus().length) {  
      recommendedChanges.cpus = config.cores + 1;  
    }  
  }  
}
```

```

    } else if (usage.worker1 < threshold && usage.worker2 < threshold) {
      if (cores.worker1 === cores.worker2 && cores.worker1 > 1) {
        recommendedChanges.cpus = config.cores - 1;
        // minimum 10% cpu usage
      } else if (config.executionCap > 10) {
        recommendedChanges.executionCap = config.executionCap - 20 > 10 ?
config.executionCap - 20 : 10;
      }
    }
    return recommendedChanges;
  }
}

```

Логіка визначення моменту адаптації системи до об'єму використаної оперативної пам'яті є схожою на логіку визначення такого моменту для ресурсів процесора.

Визначається об'єм доступної оперативної пам'яті у мегабайтах. Для кожного із підлеглих серверів визначається відсоток використовуваної оперативної пам'яті в залежності від максимальної, яка доступна серверу у даний момент часу. Якщо такий відсоток є вищим за встановлений коефіцієнт адаптивності, додають додаткові 256 Мб оперативної пам'яті, якщо система може її виділити.

Інакше, якщо об'єм використовуваної оперативної пам'яті занадто великий, перевіряють чи поточний розмір доступної серверу оперативної пам'яті перевищує мінімальний – 256 Мб, і якщо в такому разі він є більшим за мінімальний розмір – від доступного розміру оперативної пам'яті віднімають 256 Мб, однак результуюча величина не може бути меншою за мінімальну.

Відображення описаної логіки мовою Javascript ES6 надана нижче:

```

shouldUpdateRamUsage(usage, threshold, ramPerConfig) {
  const recommendedChanges = {};
  const totalMemory = os.totalmem() / (1024 ** 2);
  // if one of nodes rebooting

```



```

if (!usage.worker1 || !usage.worker2) return recommendedChanges;
const worker1UsageMB = usage.worker1 / (1024 ** 2);
const worker2UsageMB = usage.worker2 / (1024 ** 2);
const worker1UsagePercent = (worker1UsageMB / ramPerConfig) * 100;
const worker2UsagePercent = (worker2UsageMB / ramPerConfig) * 100;
if (worker1UsagePercent >= threshold && worker2UsagePercent >=
threshold) {
  if (ramPerConfig < totalMemory) {
    recommendedChanges.ram = ramPerConfig + 256 > totalMemory ?
totalMemory : ramPerConfig + 256;
  }
  } else if (worker1UsagePercent < threshold && worker2UsagePercent <
threshold) {
    if (ramPerConfig > 256) {
      recommendedChanges.ram = ramPerConfig - 256 > 256 ? ramPerConfig -
256 : 256;
    }
  }
  return recommendedChanges;
}

```

В момент наступної адаптації системи до нових параметрів, створюється Customfile з інструкціями для зміни параметрів віртуальних машин. Цей файл також проходить через шаблонізатор handlebars і на виході маємо готовий файл з новими параметрами. Під час перезавантаження віртуальної машини, поправки будуть зчитані з цього файлу і задіяні до наступного запуску, який буде виконано автоматично.

Приклад шаблону інструкції з поправками для конфігурації віртуальної машини.

```

config.vm.provider "virtualbox" do |v|
  v.customize [

```

```
'modifyvm', :id,  
"--cpuexecutioncap", {{cpuexecutioncap}},  
"--memory", {{ram}},  
"--cpus", {{cpus}}  
]  
end
```

Оскільки рекомендації для адаптації ресурсів процесору та пам'яті можуть бути не надані, замість них шаблонізатор вписує дані, які отримує із поточної конфігурації віртуальної машини.

У разі, якщо жодних рекомендацій не надійшло, сервери залишаються без змін та не перезавантажуються.

6.5 Висновки до шостого розділу

У даному розділі представлений детальний опис концепції проектування критичної інформаційної інфраструктури, визначені її основні принципи та положення. Далі наведено опис узагальненої архітектури критичної інформаційної інфраструктури, визначені її основні рівні функціонування та наведено її життєвий цикл. Запропонована загальна схема пошуку проектних рішень на базі компенсаційно-декомпенсаційного підходу до проектування та обґрунтування проектних рішень щодо критичної інформаційної інфраструктури та відповідна візуалізація для відображення стратегій обґрунтування рішень, стратегій прийняття рішень стосовно архітектури критичної інформаційної інфраструктури. Запропонована концепція інтегрує в єдину систему проектування критичної інформаційної інфраструктури сам процес дизайну архітектурних проектів критичної інформаційної інфраструктури, моделювання, верифікацію та обґрунтований вибір кращого архітектурного рішення за визначеними критеріями та його представлення у вигляді виробничої структури з певним визначеним життєвим циклом та функціональними задачами.

ВИСНОВКИ

Дисертаційна робота становить собою закінчене наукове дослідження, присвячене вирішенню актуальної науково-прикладної проблеми, яка полягає в удосконаленні процесів вибору і обґрунтування проектних рішень щодо критичної інформаційної інфраструктури на підставі: використання розширених UML-моделей; розроблених методів представлення процесу прийняття рішень щодо вибору та обґрунтування архітектури критичної інформаційної інфраструктури, оцінки альтернативних проектних рішень, вибору оптимальної конфігурації компонент критичної інформаційної інфраструктури; методів розвитку критичної інформаційної інфраструктури; подальшого розвитку концепції проектування критичної інформаційної інфраструктури; створення та застосування єдиної системи проектування критичної інформаційної інфраструктури. Проведені дослідження дозволили отримати наступні наукові та практичні результати:

1. Досліджено потенційні можливості сучасних фрейворків опису архітектури підприємства та підходів до обґрунтування рішень щодо архітектури при проектуванні КІІ, що дозволило визначити критерій та показники оптимальності проектування та функціонування критичних інформаційних інфраструктур.

2. Розроблено модель перетворення розширених UML-діаграм, які відображають процес прийняття рішення щодо вибору та обґрунтування архітектури критичної інформаційної інфраструктури, в розмічені транзиційні системи. Використання даної моделі дозволяє архітектору застосовувати математичні методи на базі розмічених транзиційних систем для дослідження властивостей проектних рішень, представлених у вигляді розширених UML-діаграм.

3. Сформульовано постановку задачі верифікації параметризованих моделей архітектурних рішень критичної інформаційної інфраструктури в термінах теорій розмічених транзиційних систем та темпоральної логіки, що дозволяє використати в подальшому весь спектр методів та алгоритмів, розроблених для цих теорій, для верифікації/генерації проектних рішень критичної інформаційної інфраструктури.

4. Розроблено метод обґрунтування проектних рішень щодо архітектури критичної інформаційної інфраструктури, в якому для порівняння альтернативних проектних рішень застосовуються множина з трьох показників ризику, що враховують досяжність цілей проектного рішення, можливість його імплементації та дотримання вимог щодо критичності, та ентропійний підхід для оцінювання їх взаємного впливу для задачі проектування критичної інформаційної інфраструктур. Метод дозволяє архітектору обирати найкращий варіант проектного рішення та оцінювати вплив окремих проектних рішень або елементів на інші проектні рішення щодо архітектури критичної інформаційної інфраструктури або на весь дизайн архітектури в цілому. Використання даного методу дозволило врахувати вплив окремих проектних рішень або елементів на інші проектні рішення щодо архітектури при проектуванні єдиної інформаційної системи Міністерства внутрішніх справ України.

5. Розроблено метод проектування критичних інформаційних інфраструктур на базі розширених відкритих гібридних автоматів, який поширює застосування відкритих гібридних автоматів для задачі проектування критичної інформаційної інфраструктури. Застосування методу дозволило при проектуванні ЦОД ЄІС МВС врахувати проблему взаємозалежності телекомунікаційної системи від систем охолодження та електроживлення, і таким чином, якісно продумати резервування відповідних систем.

6. Розроблено метод структурної оптимізації нейронних мереж прямого поширення, який дозволяє отримувати оптимальну для вхідних даних

структуру нейронної мережі, що значно підвищує можливості адаптаційного вибору моделей нейронних мереж для розв'язання задач функціонування критичних інформаційних інфраструктур. Метод перевірено на різних типах вхідних даних та показано його ефективність для розв'язання окремих класів задач класифікації та розпізнавання.

7. Розроблено та програмно реалізовано метод верифікації даних в інформаційних ресурсах критичної інформаційної інфраструктури, який використовує процеси хешування і може застосовуватися для даних в знеособленому вигляді, що дозволило значно пришвидшити процес верифікації та скоротити кількість невизначених даних в національних електронних інформаційних ресурсах України при проведенні державного експерименту з верифікації даних.

8. Розроблено та програмно реалізовано керовану моделлю систему розподілу ресурсів критичної інформаційної інфраструктури, яка використовує методи оптимізації структури нейронних мереж та оптимізації навантаження критичної інформаційної інфраструктури, що дозволило в автоматичному режимі розподіляти ресурси критичної інформаційної інфраструктури з метою їх оптимального використання та задоволення потреб сервісів та компонент, що їх використовують.

9. Розроблено метод представлення та обґрунтування архітектури критичної інформаційної інфраструктури на основі розширених UML-діаграм, який дозволив спростити та пришвидшити пошук оптимальної архітектури ЄІС МВС, а також надав можливість зберегти всю історію проведених міркувань та обґрунтувань при її виборі.

10. Розроблено та програмно реалізовано метод оцінки та вибору оптимальної конфігурації компонент критичної інформаційної інфраструктури на базі Марківського процесу прийняття рішень, який дозволив оцінити та вибрати оптимальну конфігурацію компонент при проектуванні ЦОД ЄІС МВС

України, що дало можливість значно скоротити час на розробку відповідного технічного завдання.

11. Розроблено та програмно реалізовано метод розподілу ресурсів критичної інформаційної інфраструктури на базі генетичного алгоритму з чіткими параметрами, що надає можливість швидко визначити оптимальну схему розподілу ресурсів в центрах обробки даних з врахуванням критичності окремих процесів та сервісів. Даний метод використано для оптимізації розподілення ресурсів в ЦОД ЄІС МВС.

12. Розроблено та програмно реалізовано метод оптимізації навантаження критичної інформаційної інфраструктури, який використовує метод рою часток, що надав можливість швидко визначити оптимальну схему розміщення віртуальних машин та критичних сервісів на них з врахуванням вимог щодо критичності окремих сервісів та процесів, і відповідно, оптимізувати розподіл ресурсів в ЦОД ЄІС МВС.

13. Отримали подальший розвиток концепція проектування критичної інформаційної інфраструктури, типова виробнича архітектура ІТ-інфраструктури, життєвий цикл критичної інформаційної інфраструктури та метод верифікації параметризованих моделей архітектурних рішень критичної інформаційної інфраструктури, що дало можливість створити єдину систему проектування критичної інформаційної інфраструктури та набір інструментарію, який дозволяє значно прискорити та підвищити ефективність процесів проектування критичної інформаційної інфраструктури.

14. Розроблена єдина система проектування критичної інформаційної інфраструктури впроваджена на підприємстві ДП «ІНФОТЕХ», що дозволило скоротити час проектування ЦОД ЄІС МВС України на 20%.

15. Результати дисертаційного дослідження використані Міністерством внутрішніх справ України при формуванні технічних завдань на ЦОД ЄІС МВС України та функціональних підсистем ЄІС МВС, що дозволило скоротити час на

визначення відповідних специфікацій обладнання та програмних компонент на 18%.

16. Впровадження методу пошуку сумнівних записів при виконанні завдання верифікації даних в національних електронних інформаційних реєстрах дозволило Міністерству внутрішніх справ України скоротити час на виконання верифікації на 90% та скоротити кількість невизначених записів в ресурсах до 2%.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] Я. Дорогий and В. Цуркан, “Методи представлення та обґрунтування архітектури критичної ІТ-інфраструктури,” in *Engineering sciences: development prospects in countries of Europe at the beginning of the third millennium: Collective monograph*, vol. 1, Riga: Izdevnieciba “Baltija Publishing,” 2018, pp. 197–237.
- [2] Y. Y. Dorogyu, “Моделі і методи представлення архітектурних рішень критичної ІТ-інфраструктури,” in *Modern engineering research: topical problems, challenges and modernity*, vol. 1, Riga: Izdevnieciba “Baltija Publishing,” 2020, pp. 127–155.
- [3] Y. Y. Dorogyu, “Management of Critical It-Infrastructures,” *Inf. Telecommun. Sci.*, vol. 0, no. 1, pp. 10–15, Aug. 2014, doi: 10.20535/2411-2976.12014.10-15.
- [4] Я. Дорогий, “Життєвий цикл критичної ІТ-інфраструктури,” *Електроніка та зв’язок*, vol. 20, no. 4, pp. 100–105, 2015.
- [5] Я. Дорогий, “Порівняльний аналіз методів представлення та обґрунтування архітектури критичної ІТ-інфраструктури,” *Вісник НТУ «ХПІ». Серія Механіко-технологічні системи та комплекси*, vol. 33, no. 1255, pp. 42–54, 2017.
- [6] Я. Дорогий, “Мета-модель компенсаційно-декомпенсаційного підходу до проектування архітектури критичної ІТ-інфраструктури,” *Вісник університету України Інформатика, обчислювальна техніка та кібернетика*, vol. 2, no. 17, pp. 14–22, 2016.
- [7] Я. Дорогий, “Розробка підходу до проектування, моделювання та дослідження критичної ІТ-інфраструктури,” *Технологічний аудит та резерви виробництва*, vol. 5, no. 2(37), pp. 34–41, 2017.
- [8] Y. Dorogyu, “Development of a model for optimal configuration components selection for architecture of critical IT infrastructure at its designing,” *Технологічний аудит та резерви виробництва*, no. 6/2(38), pp. 19–27, 2017.
- [9] Я. Дорогий, “Алгоритм структурної оптимізації нейронної мережі,” *Вісник НТУУ «КПІ», «Інформатика, управління та обчислювальна техніка»*, no. 61, pp. 169–173, 2014.
- [10] Я. Дорогий, “ТС-сумісна архітектура критичної ІТ-інфраструктури,” *Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка збірник наукових праць*, vol. 65, pp. 123–126, 2017.
- [11] Я. Дорогий, “Розподіл ресурсів критичної ІТ-інфраструктури з використанням хмарних технологій,” *Електроніка та зв’язок*, vol. 1, no. 90, pp. 42–49, 2016.
- [12] Я. Дорогий, “Технологія пошуку сумнівних записів при створенні єдиного реєстру ідентифікації фізичних осіб України,” *Inf. Technol. Secur.*, vol. 7, no.

2, pp. 114–125, 2019, doi: 10.20535/2411-1031.2019.7.2.190539.

- [13] Y. Dorohyi, V. Tsurkan, S. Telenyk, and O. Doroha-Ivaniuk, “A comparison enterprise architecture frameworks for critical it infrastructure design,” *Inf. Technol. Secur.*, vol. 5, no. 2, pp. 90–118, 2017, doi: 10.20535/2411-1031.2017.5.2.136976.
- [14] I. Bondarenko, Y. Dorohyi, S. Stirenko, and T. Shemsedynov, “Analysis of critical IT infrastructure designing problems for a ministry,” *Collect. "Information Technol. Secur."*, vol. 6, no. 1, pp. 96–107, Jun. 2018, doi: 10.20535/2411-1031.2018.6.1.153207.
- [15] Y. Dorogyu, S. Telenyk, V. Tsurkan, and D. Halushko, “Structural optimization of neural network in qualitative evaluation method of IT-infrastructure functioning,” *Inf. Telecommun. Sci.*, vol. 6, no. 2, pp. 36–43, 2015.
- [16] Я. Дорогий, О. Дорога-Іванюк, В. Цуркан, and Д. Ференс, “Застосування алгоритму структурної оптимізації нейронної мережі в задачах класифікації даних,” *Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка збірник наукових праць*, no. 62, pp. 100–104, 2015.
- [17] О. Дорога-Іванюк, Я. Дорогий, and Д. Ференс, “Реалізація алгоритму структурної оптимізації нейронної мережі,” *Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка збірник наукових праць*, no. 63, pp. 101–106, 2015.
- [18] G. Nowakowski, Y. Dorogyu, and O. Doroga-Ivaniuk, “Neural network structure optimization algorithm,” *J. Autom. Mob. Robot. Intell. Syst.*, vol. 12, no. 1, pp. 5–13, 2018, doi: 10.14313/JAMRIS_1-2018/1.
- [19] Я. Дорогий, Є. Максименко, О. Крук, and В. Цуркан, “Оцінювання ризику безпеки інформації на основі спектрального підходу,” *Inf. Technol. Secur.*, vol. 3, no. 2(5), pp. 138–146, 2015.
- [20] Я. Дорогий, В. Мохор, В. Цуркан, and О. Крук, “Функціональне моделювання системи керування ризиком безпеки інформації,” *Захист інформації*, vol. 18, no. 1, pp. 74–80, 2016.
- [21] Я. Дорогий, О. Бакалинський, О. Богданов, С. Михайлов, В. Цуркан, and В. Мохор, “Використання ентропійного підходу для оцінювання ризиків безпеки інформації,” *Inf. Technol. Secur.*, vol. 4, no. 2(7), pp. 255–261, 2016.
- [22] Y. Dorohyi, O. Doroha-Ivaniuk, and D. Ferens, “Resources distribution model of critical IT infrastructure with clear parameters based on the genetic algorithm,” *Collect. "Information Technol. Secur."*, vol. 6, no. 2, pp. 124–144, Dec. 2018, doi: 10.20535/2411-1031.2018.6.2.153497.
- [23] Я. Дорогий, Д. Ференс, and О. Дорога-Іванюк, “Модель розподілу ресурсів критичної ІТ-інфраструктури з чіткими параметрами на основі методу рою часток,” *Електронне моделювання*, vol. 41, no. 2, pp. 1–15, 2019.

- [24] А. Йовенко and Я. Дорогий, “Вибір методу інтеграції розподілених гетерогенних модулів для розробки корпоративних систем,” *Вісник університету України Інформатика, обчислювальна техніка та кібернетика*, vol. 1, no. 18, pp. 14–22, 2016.
- [25] Я. Дорогий, В. Мохор, and В. Цуркан, “Концептуальна модель описання архітектури системи управління інформаційною безпекою,” *Безпека інформації*, vol. 25, no. 3, pp. 162–166, 2019.
- [26] Y. Y. Dorogyu and V. V. Tsurkan, “A survey of parametric model verification methods,” *Collect. Sci. Publ. NUS*, vol. 12, no. 1(479), pp. 82–90, 2020, doi: 10.15589/znp2020.1(479).10.
- [27] V.V. Mohor, V.V. Tsurkan, Y.Y. Dorogyu, and S.Y.M., “Information security management systems architecture frameworks,” *Informatics Math. methods Simul.*, vol. 9, no. 4, pp. 209–221, 2019, doi: 10.15276/imms.v9.no4.209.
- [28] G. Nowakowski, Y. Dorogyu, and O. Doroga-Ivaniuk, “The realisation of neural network structural optimization algorithm,” in *Proceedings of the 2017 Federated Conference on Computer Science and Information Systems, FedCSIS 2017*, vol. 11, 2017, pp. 1365–1371.
- [29] Д. Ференс and Я. Дорогий, “Оптимізація розміщення віртуальних машин у хмарних інфраструктурах за допомогою гнучкого алгоритму на базі генетичного програмування,” *Інфокомунікаційні системи та технології*, vol. 1, no. 1, 2017.
- [30] К. Левченко and Я. Дорогий, “Порівняння часу виконання базових операцій фреймворків машинного навчання,” *Інфокомунікаційні системи та технології*, vol. 2, no. 2, pp. 27–31, 2018.
- [31] S. Telenyk, O. Rolick, M. Bukasov, Y. Dorogiy, D. Halushko, and A. Pysarenko, “Qualitative evaluation method of IT-infrastructure elements functioning,” in *2014 IEEE International Black Sea Conference on Communications and Networking, BlackSeaCom 2014*, 2014, pp. 165–169, doi: 10.1109/BlackSeaCom.2014.6849031.
- [32] Y. Dorogyu and K. Valchuk, “Comparison between serialization formats,” in *Collection of Conference Papers of International Scientific-Practical Conference “Problems and Prospects of Intergration of Science and Technology,”* 2015.
- [33] Я. Дорогий, І. Козлюк, В. Цуркан, and В. Мохор, “Критична інфраструктура: вразливості, загрози, ризики,” in *II міжнародна науково-практична конференція «Інформаційні технології та взаємодії»*, 2015.
- [34] О. Дорога-Іванюк and Я. Дорогий, “Критерій оптимальності побудови критичної ІТ-інфраструктури,” in *Актуальні проблеми розвитку науки і техніки: Матеріали другої міжнародної науково-технічної конференції, 20 грудня*, 2015.

- [35] Я. Дорогий, Є. Максименко, В. Цуркан, and В. Мохор, “Аналітика кібербезпеки: поняття, структура, задачі,” in *Международная научная конференция ИТБ-2015, 21 октября, 2015*.
- [36] О. Дорога-Іванюк and Я. Дорогий, “Умови створення критичних процесів або сервісів критичної ІТ-інфраструктури,” in *V міжнародна науково-технічна конференція «Проблеми інформатизації», 10-11 грудня, 2015*.
- [37] Y. Dorogyu, O. Doroga-Ivaniuk, B. Mart, and V. Tsurkan, “Life cycle of IT-infrastructure,” in *Proceedings of The Congress on Information Technology, Computational and Experimental Physics 2015 (CITCEP 2015), 2015*.
- [38] Y. Dorogyu, O. Doroga-Ivaniuk, S. Telenik, and D. Halushko, “Qualitative evaluation method of IT-infrastructure functioning based on structural optimization of neural network,” in *2015 2nd International Scientific-Practical Conference Problems of Infocommunications Science and Technology, PIC S and T 2015 - Conference Proceedings, 2015, pp. 1–4, doi: 10.1109/INFOCOMMST.2015.7357252*.
- [39] Д. Ференс and Я. Дорогий, “Структурна оптимізація штучних нейронних мереж,” in *Summer Infocom 2016: Матеріали 2-ї Міжнародної науково-практичної конференції, 2016*.
- [40] Я. Дорогий, О. Крук, В. Цуркан, and В. Мохор, “Процесна модель системи керування ризиком безпеки інформації,” in *V Міжнар. на-ук.-техн. конф. “Захист інформації і безпека інформаційних систем,” 2016*.
- [41] Я. Дорогий, О. Крук, В. Цуркан, and В. Мохор, “Дерево вузлів функціональної моделі системи керування ризиком безпеки інформації,” in *Міжнар. наук.-практ. конф. “Безпека інформації в інформаційно-телекомунікаційних системах”, 25-26 травня, 2016*.
- [42] Я. Дорогий, О. Крук, В. Мохор, and В. Цуркан, “Модель потоків даних системи керування ризиком безпеки інформації,” in *V-а міжнародна науково-практична конференція “Кібербезпека інформаційних технологій /// Фізико-технологічні проблеми передавання, оброблення та зберігання інформації в інфокомунікаційних системах, 2016*.
- [43] Я. Дорогий, О. Крук, В. Цуркан, and В. Мохор, “Функціональна модель системи керування ризиком безпеки інформації,” in *II-а міжнародна науково-практична конференція “Актуальні проблеми моделювання ризиків і загроз виникнення надзвичайних ситуацій на об'єктах критичної інфраструктури,” 2016*.
- [44] Y. Y. Dorogyu, O. O. Doroha-Ivaniuk, U. Dzelendzyak, and K. Tomczyk, “Usage of structural optimization algorithm of neural nets in problems of data classification,” in *Proceedings of the 2017 IEEE 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2017, 2017, vol. 2, pp. 983–987, doi:*

10.1109/IDAACS.2017.8095234.

- [45] Я. Дорогий, “Проектування критичних ІТ-інфраструктур з використанням розмічених транзиційних систем,” in *V заочная научная конференция «Фундаментальные и прикладные исследования в современной науке»*, 31 октября, 2017.
- [46] Я. Дорогий, “Моделювання критичних ІТ-інфраструктур з використанням розмічених транзиційних систем,” in *VI Міжнародна науково-практична конференція (I Міжнародний симпозіум «Практичне застосування нелінійних динамічних систем в інфокомунікаціях», PREDT-2017*, 9-11 листопада, 2017.
- [47] Я. Дорогий, О. Дорога-Іванюк, and В. М. В. Цуркан, “Cyber resilience: the concept of information security in cyberspace,” in *III міжнародна науково-практична конференція «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації»*, 2017.
- [48] Y. Dorogyu, “Meta model of compensatory-decompensational approach for architecture of critical it infrastructure designing,” in *14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, TCSET 2018 - Proceedings*, 2018, vol. 2018-April, pp. 223–228, doi: 10.1109/TCSET.2018.8336191.
- [49] Я. Дорогий, В. Мохор, and В. Ц. Ю. Штифурак, “Моделі оцінювання ризику кібербезпеки,” in *VII Міжнародна науково-практична конференція «Фізико-технологічні проблеми передавання, оброблення та зберігання інформації в інфокомунікаційних системах (PREDT-2018)»*, 8-10 листопада, 2018.
- [50] А. Горносталь and Я. Дорогий, “Мурашиний алгоритм кластеризації,” in *XI Міжнародна науково-практична конференція "ІНТЕРНЕТ-ОСВІТА-2018*, 2018.
- [51] Я. Дорогий and К. Левченко, “Порівняння фреймворків машинного навчання,” in *Summer InfoCom 2018: Матеріали VI Міжнародної науково-практичної конференції з інформаційних систем та технологій*, 2018.
- [52] В. Масарік, “Застосування новітніх технологій у місцевих органах влади,” *Аспекти самоврядування*, vol. 5, no. 37, pp. 53–60, 2006.
- [53] Н. Г. Топольский and В. С. Ватагин, “Интеллектуальные системы безопасности промышленных объектов,” *Системы безопасности*, vol. 2, no. 50, pp. 100–103, 2003.
- [54] В. Б. Кисельов, “Впровадження сучасних інформаційно-телекомунікаційних технологій та вдосконалення інформаційного забезпечення в системах державного управління як нові стратегії адміністративних перетворень,” *Наукові розвідки з державного та муніципального управління*, no. 1, pp. 348–361, 2015.

- [55] Г. Ю. Мотков, “Обеспечение безопасности объектов транспортного комплекса - наша приоритетная задача,” *Системы безопасности*, vol. 6, no. 48, pp. 8–9, 2003.
- [56] Л. О. Нікіфорова, А. А. Шиян, and Ю. Яремчук, “Моделювання вибору оптимального методу протидії загрозам інформаційній безпеці,” *Реєстрація, зберігання і обробка даних*, vol. 16, no. 4, pp. 28–33, 2014.
- [57] П. В. Сундеев and В. С. Симанков, *Системный анализ функциональной стабильности критичных информационных систем: Монография*. Краснодар: КубГТУ; ИСТЭК, 2004.
- [58] В. І. Пожусь, “Становлення і розвиток інформаційного глобального суспільства,” *Гуманітарний вісник ЗДІА*, no. 51, pp. 5–13, 2012.
- [59] Я. В. Бон, Д. Пондаман, and Г. Кеммерлинг, *Сервис-менеджмент. Введение*. Ред., М: IT Expert, 2003.
- [60] П. Брукс, *Метрики для управления ИТ-услугами*. М: Альпина Бизнес Букс, 2008.
- [61] Н. Дубова, “ITSM - новая идеология управления ИТ.” [Online]. Available: <http://www.osp.ru/os/2000/10/178254/>. [Accessed: 28-Jun-2018].
- [62] О. И. Шелухин, *Моделирование информационных систем: учебное пособие*. М: Радиотехника, 2005.
- [63] Т. Л. Бірюкова and В. Г. Спрінсян, *Ресурси та технології інформаційного менеджменту*. Одеса: ОНПУ, 2012.
- [64] В. К. Батоврін, “Сучасний стан міжнародних стандартів системної та програмної інженерії,” 2018. [Online]. Available: <http://www.klubok.net/article2529.html>. [Accessed: 30-Jul-2018].
- [65] Г. Н. Исаев, *Информационный менеджмент. Ч. 2: Управление качеством информационных систем*. М: МГУС, 2005.
- [66] Г. Н. Исаев, *Моделирование оценки качества информационных систем*. М: ИМСГС, 2006.
- [67] В. Г. Елиферов and В. В. Репин, *Процессный подход к управлению. Моделирование бизнес-процессов*. М: РИА “Стандарты и качество,” 2004.
- [68] В. Б. Школьный, “Деякі причини виникнення і розвитку злочинності у сфері використання ЕОМ,” *Право і суспільство*, no. 2, pp. 222–227, 2012.
- [69] С. В. Якимова, “Кримінологічне забезпечення оперативно-розшукової діяльності.” [Online]. Available: <http://science.lpnu.ua/sites/default/files/journal-paper/2017/aug/5784/vnulpurn201685084.pdf>. [Accessed: 30-Oct-2018].
- [70] М. А. Борисов and Ю. В. Кузнецов, “Основные положения национального

плана защиты информационных систем США,” in *II межвузовская научно-техническая конференция*, 2001.

- [71] В. Хорошко and М. Прокофьев, “Проблеми захисту інформації в Україні,” *Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні*, vol. 2, no. 30, pp. 9–14, 2015.
- [72] О. С. Виговська, “Система забезпечення інформаційної безпеки України в період глобальної кризи,” *Актуальні проблеми міжнародних відносин*, vol. 93, no. 1, pp. 108–113, 2010.
- [73] В. Ю. Степанов, “Інформаційна безпека як складова державної інформаційної політики,” *Державне будівництво*, no. 2, pp. 1–9, 2016.
- [74] S. Russo, “Workload Characterization for Software Aging Analysis.” [Online]. Available: http://www.academia.edu/4963481/Workload_Characterization_for_Software_Aging_Analysis. [Accessed: 07-Nov-2018].
- [75] M. Fusani, “Examining software engineering requirements in safety-related standarts,” *Радиоэлектронные и компьютерные системы*, vol. 7, no. 49, pp. 268–274, 2009.
- [76] Г. В. Певцов, “Досвід і концепції ведення інформаційної боротьби у провідних країнах світу,” *Наука і техніка Повітряних Сил Збройних Сил України*, vol. 1, no. 18, pp. 12–16, 2015.
- [77] О. М. Хошаба, “Частина 13. Захист інформації в системах електронного урядування,” in *Електронне урядування та електронна демократія: навч. посібник*, К: ФОП Москаленко О.М., 2017, p. 72.
- [78] В. С. Цимбалюк, “Забезпечення безпеки інформаційних систем цивільної авіації.” [Online]. Available: <http://www.pravnuk.info/2013-12-27-15-13-45/534-zabezpechennya-bezpeki-informacijnix-sistem-u-civilnij-aviacii.html>. [Accessed: 10-Dec-2018].
- [79] Д. Н. Хомяков and Н. Н. Хомяков, “Анализ безопасности АЭС при террористических актах,” *Системы безопасности*, vol. 2, no. 44, pp. 74–76, 2002.
- [80] А. В. Лукацкий, “Неофициальный взгляд на трагедию в США,” *Системы безопасности*, no. 41, pp. 10–11, 2001.
- [81] Е. А. Золотарева and С. В. Скрыль, “Задача оценки эффективности противодействия угрозам безопасности в критически важных сегментах информационной сферы,” in *Технологии безопасности: Материалы VIII международного форума*, 2003.
- [82] А. Н. Агапов, “Ядерная и радиационная безопасность. Готовность к ЧС,” *Системы безопасности*, vol. 2, no. 50, pp. 8–10, 2003.

- [83] О. М. Маковецький, “Підходи до удосконалення методики оцінки ефективності комплексної системи захисту інформації,” *Сучасні інформаційні технології у сфері безпеки та оборони*, vol. 2, no. 26, pp. 54–58, 2016.
- [84] Д. І. Рабчун, “Оцінка ефективності інформаційної безпеки з урахуванням економічних показників,” *Сучасний захист інформації*, no. 4, pp. 91–96, 2015.
- [85] Ф. Г. Хисамов, “Современная информационная борьба и методы защиты информации в системах управления критическими объектами государства,” *Автоматизированные системы управления и средства автоматики в пищевой промышленности*, vol. 3, pp. 43–56, 2015.
- [86] Верховна Рада України, “Закон України №2163-VIII ‘Про основні засади забезпечення кібербезпеки України’ від 05.10.2017 р.,” 2017. [Online]. Available: <http://zakon2.rada.gov.ua/laws/show/2163-19>. [Accessed: 10-Oct-2018].
- [87] А. В. Крупенин, “Системный подход к синтезу функционирования программно модулирующего комплекса автоматизированных информационных систем критически важных объектов связи,” in *Межвуз. сборник научных трудов No7*, 2008.
- [88] Р. Н. Кветний, О. Р. Бойко, and І. В. Богач, *Комп’ютерне моделювання систем і процесів: навчальний посібник*. Вінниця: ВНТУ, 2013.
- [89] Ю. М. Щєбланін and А. О. Гресько, “Загальний, комплексний опис проблем інформаційної безпеки в ‘Інтернеті речей,’” *Сучасний захист інформації*, no. 1, pp. 69–73, 2016.
- [90] А. Ролик, “Концепция управления корпоративной ИТ-инфраструктурой,” *Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка Зб. наук. пр.*, no. 56, p. 170, 2012.
- [91] А. Ролик, “Тенденции и перспективы развития управления информационными технологиями,” *Вестник НТУУ «КПІ» Інформатика, управление и вычислительная техника*, no. 55, pp. 81–109, 2012.
- [92] С. Кондратов and Д. Бірюков, *Захист критичної інфраструктури: проблеми та перспективи впровадження в Україні*. К: НІСД, 2012.
- [93] С. Кондратов and Д. Бірюков, “Концепція захисту критичної інфраструктури: стан, проблеми та перспективи її впровадження в Україні,” in *зб. матеріалів міжнар. наук.-практ. конф.*, 2013.
- [94] С. Кондратов and Д. Бірюков, *Зелена книга з питань захисту критичної інфраструктури в Україні*. К: НІСД, 2015.
- [95] С. С. Бучик and О. К. Юдін, *Державні інформаційні ресурси. Методологія побудови класифікатора загроз: монографія*. К: НАУ, 2015.

- [96] В. І. Єсін, Л. С. Сорока, and О. О. Кузнецов, *Безпека інформаційних систем і технологій: навчальний посібник*. Х: ХНУ імені В.Н.Каразіна, 2013.
- [97] К. М. Носенко, Т. А. Ліхоузова, and О. І. Півторак, “Огляд систем виявлення атак в мережевому трафіку,” *Міжсвідомчий науково-технічний збірник “Адаптивні системи автоматичного управління,”* vol. 1, no. 24, pp. 67–75, 2014.
- [98] Е. Золотарева, “Правила структуризации частных показателей системы противодействия (СП) угрозам безопасности в критически важных сегментах информационной сферы,” in *Технологии безопасности: Материалы VIII Международного форума*, 2003.
- [99] В. Асеев and Е. Золотарева, “Имитационная модель для оценки временных характеристик средств противодействия угрозам безопасности элементов информационной сферы,” *Информация и безопасность*, no. 2, pp. 147–149, 2003.
- [100] О. Корченко, О. Романенко, and Ю. Дрейс, “Формування множини ідентифікаторів для класифікації об’єктів критичної інформаційної інфраструктури,” in *Актуальні проблеми забезпечення кібербезпеки та захисту інформації: тези доповідей учасників IV Міжнародної науково-практичної конференції, 21-24 лютого, 2018*, 2018.
- [101] О. Корченко, О. Романенко, and Ю. Дрейс, “Критична інформаційна інфраструктура України: терміни, сектори і наслідки,” *Захист інформації*, vol. 19, no. 4, pp. 303–309, 2017.
- [102] С. Гнатюк, В. Лядовська, and М. Рябий, “Визначення критичної інформаційної інфраструктури та її захисту: аналіз підходів,” *Зв’язок*, no. 4, pp. 3–7, 2014.
- [103] С. Гнатюк, Н. Сейлова, and В. Сидоренко, “Універсальна модель даних для формування переліку об’єктів критичної інформаційної інфраструктури держави,” *Безпека інформації*, vol. 23, no. 2, pp. 87–91, 2017.
- [104] В. Харченко, *Безопасность критических инфраструктур: математические и инженерные методы анализа и обеспечения*. Х: , Х: Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», 2011.
- [105] В. Харченко, “Гарантоспособность и гарантоспособные системы: элементы методологии,” *Радиоэлектронные и компьютерные системы*, vol. 5, no. 17, pp. 7–19, 2006.
- [106] В. Харченко, В. Жихарев, Н. Нечипорук, and В. Илюшко, *Многоверсионные системы, технологии, проекты*. Х: Нац. ерокосм. ун-т «Харьк. авиац. ин-т», 2003.
- [107] А. Становский, Т. Бирик, and В. Тонконогий, “Автоматизация управлением

объектами повышенной опасности,” in *Материалы международной научно-технической конференции «Автоматизация: проблемы, идеи, решения»*, 2009.

- [108] М. Ястребенецкий, О. Бутова, and В. Инюшев, “Оценка уровня безопасности информационных и управляющих систем АЭС,” *Радиоэлектронные и компьютерные системы*, vol. 8, no. 27, pp. 96–103, 2008.
- [109] М. Ястребенецкий, *Информационно-управляющие системы АЭС: проблемы безопасности*. К: Техника, 2004.
- [110] А. Ролик, А. Волошин, Д. Галушко, А. Покотило, and П. Можаровский, “Система управления корпоративной информационно-телекоммуникационной инфраструктурой на основе агентского подхода,” *Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техніка*, no. 52, pp. 39–52, 2010.
- [111] С. Ф. Теленик, О. І. Ролік, А. Ю. Лабунський, and М. М. Букасов, “Моделі управління віртуальними машинами при серверній віртуалізації,” *Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техніка*, no. 51, pp. 147–152, 2009.
- [112] С. Ф. Теленик *et al.*, “Методы исследования свойств высокопроизводительных инфраструктур. Обзор,” *Фундаментальные и прикладные проблемы информатики и информационных технологий*, vol. 1, pp. 3–13, 2015.
- [113] Верховна Рада України, “Проект Закону України ‘Про внесення змін до деяких законів України щодо забезпечення кібернетичної безпеки України’ від 31.08.2012.” [Online]. Available: http://w1.c1.rada.gov.ua/pls/zweb2/webproc4_1?id=&pf3511=44208. [Accessed: 10-Dec-2018].
- [114] *COBIT 5.0. Российское издание*. М: ISACA.
- [115] В. Шевченко and Е. Мащенко, “Исследование процессов сервисного обслуживания критических ИТ-инфраструктур на основе полумарковской модели,” *Радіоелектронні і комп’ютерні системи*, no. 5, pp. 57–63, 2013.
- [116] А. Скатков, А. Скидан, Н. Кузнецова, and Д. Воронин, “Математическая модель для решения задачи параметрического синтеза в критических ИТ-инфраструктурах,” *Збірник наукових праць СХУЯЕтаП*, pp. 229–233, 2013.
- [117] В. Шевченко and Е. Мащенко, “Исследование критических ситуаций в ИТ-инфраструктурах методами кластерного анализа,” *Радіоелектронні і комп’ютерні системи*, no. 5, pp. 191–196, 2013.
- [118] Кабінет Міністрів України, “Постанова №563 ‘Про затвердження Порядку формування переліку інформаційно-телекомунікаційних систем об’єктів

критичної інфраструктури держави' від 23.08.2016," 2016. [Online]. Available: <http://zakon5.rada.gov.ua/laws/show/563-2016-п>. [Accessed: 20-Dec-2018].

- [119] D. GARLAN and M. SHAW, "an Introduction To Software Architecture," *Adv. Softw. Eng. Knowl. Eng.*, vol. 2, pp. 1–39, 1993, doi: 10.1142/9789812798039_0001.
- [120] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture," *ACM SIGSOFT Softw. Eng. Notes*, vol. 17, no. 4, pp. 40–52, 1992, doi: 10.1145/141874.141884.
- [121] R. T. Monroe, A. Kompanek, R. Melton, and D. Garla, "Architectural Styles, Design Patterns, and Objects," *IEEE Softw.*, vol. 14, pp. 43–52, 1997.
- [122] I.E.E.E., "IEEE Recommended Practice for Architecture Description of Software-Intensive System (IEEE Std 1471-2000)," 2000.
- [123] A. Eden and R. Kazman, "Architecture, Design, Implementation," in *International Conference for Software Engineering*, 2003.
- [124] P. Hinssen and J. Derynck, "Business/IT Fusion: How to Move Beyond Alignment and Transform IT in Your Organization," *Across Technol.*, p. 276, 2009.
- [125] US Office of Management and Budget, "The Common Approach to Federal Enterprise Architecture," *Off. Manag. Budg.*, p. 52, 2012.
- [126] J. Schekkerman, *How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating or Choosing an Enterprise Architecture Framework*, vol. 480017. Trafford Publishing, 2006.
- [127] J. A. Zachman, "John Zachman's Concise Definition of the Zachman Framework," *The SIM Guide to Enterprise Architecture*. pp. 61–65, 2017, doi: 10.1201/9781439811146-9.
- [128] J. W. Ross, "Creating a Strategic IT Architecture Competency: Learning in Stages," *SSRN Electron. J.*, p. 18, 2005, doi: 10.2139/ssrn.416180.
- [129] J. Zachman, "The Zachman Framework For Enterprise Architecture," *CA Magazine*, vol. 128, no. 9. p. 15, 2003, doi: 10.1109/CSIE.2009.478.
- [130] "Layered Enterprise Architecture | LEADing Practice." [Online]. Available: <http://www.leadingpractice.com/enterprise-standards/enterprise-architecture/layered-enterprise-architecture/>. [Accessed: 18-Oct-2017].
- [131] J. Han and P. Chen, "Architecture support for system-of-systems evolution," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2002, vol. 2480, pp. 332–346, doi: 10.1007/3-540-45785-2_26.
- [132] A. Tang, J. Han, and P. Chen, "A comparative analysis of architecture

frameworks,” in *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, 2004, pp. 640–647, doi: 10.1109/APSEC.2004.2.

- [133] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, Second Edition. Boston: Addison Wesley, 2003.
- [134] R. Lagerström, U. Franke, P. Johnson, and J. Ullberg, “A Method for creating enterprise architecture metamodels - applied to systems modifiability analysis,” *Int. J. Comput. Sci. Appl.*, vol. 6, no. 5, pp. 89–120, 2009.
- [135] R. Winter and R. Fischer, “Essential layers, artifacts, and dependencies of enterprise architecture,” *Proc. - 2006 10th IEEE Int. Enterp. Distrib. Object Comput. Conf. Work. EDOCW2006*, vol. 3, no. 2, pp. 7–18, 2006, doi: 10.1109/EDOCW.2006.33.
- [136] J. Sembiring, E. T. Nuryatno, and Y. S. Gondokaryono, “Analyzing the Indicators and Requirements in Main Components Of Enterprise Architecture Methodology Development Using Grounded Theory in Qualitative Methods,” in *SSRN Electronic Journal*, 2012, doi: 10.2139/ssrn.1867875.
- [137] A. Ramadhan, “Enterprise Architecture in University.” 2014.
- [138] F. Hald, “Enterprise Architecture in Denmark,” vol. December, no. December. pp. 1–28, 2006.
- [139] R. Ravenhorst, “Developing a project architecture and evaluating the Enterprise Architecture initiative at Tata Steel,” vol. February. pp. 1–75, 2012.
- [140] S. a Bernard, “The Strategic Use of Enterprise Architecture,” *IT Strateg. IT Gov. StatoilHydro*, vol. June, 2008.
- [141] M. A. Mohamed, G. H. Galal-Edeen, H. A. Hassan, and E. E. Hasanien, “An evaluation of enterprise architecture frameworks for E-government,” *Proceedings - ICCES 2012: 2012 International Conference on Computer Engineering and Systems*. pp. 255–260, 2012, doi: 10.1109/ICCES.2012.6408524.
- [142] “TOGAF 9.1.” [Online]. Available: <http://pubs.opengroup.org/architecture/togaf9-doc/arch/>. [Accessed: 18-Oct-2019].
- [143] The Open Group, “The TOGAF Architecture Development Method (ADM).” [Online]. Available: <http://pubs.opengroup.org/architecture/togaf9-doc/arch/Figures/adm.png>. [Accessed: 18-Oct-2017].
- [144] “Basics of the Federal Enterprise Architecture Framework,” *InnovateGov*, 2012. [Online]. Available: <http://innovategov.org/2015/04/08/basics-of-the-federal-enterprise-architecture-framework/>. [Accessed: 19-Oct-2017].
- [145] “The whole structure of FEAF.” [Online]. Available: <http://pubs.opengroup.org/architecture/togaf8-doc/arch/Figures/feaf.gif>.

[Accessed: 19-Oct-2019].

- [146] “Volume I. Overview and Concepts Manager’s Guide,” in *DoD Architecture Framework*, Overview and Concepts Manager’s Guide, 2015.
- [147] Department of Defense Deputy Chief Information Officer, “The DoDAF Architecture Framework Version 2 . 02,” *DoDAF Journal*, 2011. [Online]. Available: http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_v2-02_web.pdf. [Accessed: 19-Oct-2019].
- [148] “DoDAF V2.0 Viewpoints.” [Online]. Available: http://dodcio.defense.gov/portals/0/Images/dodaf20/clip_image002_0011.gif. [Accessed: 20-Oct-2019].
- [149] “MODAF (UK Ministry of Defence Architectural Framework) Training - Tonex Training.” [Online]. Available: <https://www.tonex.com/training-courses/modaf-training/>. [Accessed: 20-Oct-2017].
- [150] D. T. Centre, “The Human View Handbook for MODAF : Part II – Technical Description,” 2011. [Online]. Available: [https://www.defencehumancapability.com/DOCS/HFI/DTC/Legacy/HFI/Process/Phase 3/MoDAF Part 2.pdf](https://www.defencehumancapability.com/DOCS/HFI/DTC/Legacy/HFI/Process/Phase%203/MoDAF/Part%202.pdf). [Accessed: 20-Oct-2017].
- [151] “IBM Knowledge Center - MODAF overview.” [Online]. Available: https://www.ibm.com/support/knowledgecenter/SS6RBX_11.4.3/com.ibm.sa.modaf.doc/topics/c_MoDAF_Overview.html. [Accessed: 20-Oct-2019].
- [152] “The British Ministry of Defence Architecture Framework Views.” [Online]. Available: http://incoseonline.org.uk/Images/zGuides/z8_framework.jpg. [Accessed: 21-Oct-2019].
- [153] British Ministry of Defense, “MOD Architectural Framework. Technical Handbook Version 1.0,” 2005. [Online]. Available: http://www.modaf.com/files/MODAF_Technical_Handbook_V1.0.pdf. [Accessed: 21-Oct-2019].
- [154] “TOGAF TO MODAF MAPPING,” 2010. [Online]. Available: <http://docplayer.net/20752419-Togaf-to-modaf-mapping.html>. [Accessed: 22-Oct-2019].
- [155] NATO Architecture Capability Team, “NATO Architecture Framework v4.0 Methodology Documentation (draft),” 2014. [Online]. Available: <http://nafdocs.org/methodology/>. [Accessed: 22-Oct-2019].
- [156] P. Gorman, K. van Haperen, and I. Bailey, “An Update on the Convergence of MOD and NATO Architecture Frameworks,” *IntegratedEA*, 2014. [Online]. Available: <http://nafdocs.org/wp-content/uploads/2013/07/Gorman-Bailey-Van-Haperen-IEA14-v2.pdf>. [Accessed: 22-Oct-2019].
- [157] “Treasury Enterprise Architecture Framework (TEAF).” [Online]. Available:

<https://www.slideshare.net/Aamir97/treasurey-enterprise-architecture-framework-teaf>. [Accessed: 25-Oct-2017].

- [158] R. S. Bittler and G. Kreizman, “Gartner Enterprise Architecture Process: Evolution 2005,” *Gartner Enterprise Architecture Process*, 2005. [Online]. Available: <https://www.gartner.com/doc/486246/gartner-enterprise-architecture-process-evolution>. [Accessed: 25-Oct-2017].
- [159] ISO/IEC, “RM-ODP. Reference Model for Open Distributed Processing,” *ISO/IEC 10746, ITU-T Rec. X.901-X.904*, 1997.
- [160] P. R. Chelliah, “Elucidating the cloud enterprise architecture for smarter enterprises,” *IT Prof.*, vol. 16, no. 6, pp. 33–37, 2014, doi: 10.1109/MITP.2014.90.
- [161] “Service-Oriented Architecture,” in *Services Computing*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 89–113.
- [162] A. Arsanjani and A. Allam, “Service-oriented modeling and architecture for realization of an SOA,” *Proceedings - 2006 IEEE International Conference on Services Computing, SCC 2006*, 2006. [Online]. Available: <https://www.ibm.com/developerworks/library/ws-soa-design1/>. [Accessed: 25-Oct-2017].
- [163] “The Layers of Service Oriented Architecture.” [Online]. Available: <https://www.ibm.com/developerworks/library/ws-soa-design1/figure3.gif>. [Accessed: 25-Oct-2017].
- [164] L. Urbaczewski and S. Mrdalj, “a Comparison of Enterprise Architecture Frameworks,” *Issues in Information Systems*, vol. 7, no. 2, pp. 18–23, 2006.
- [165] N. A. A. Bakar, S. Harihodin, and N. Kama, “Assessment of Enterprise Architecture Implementation Capability and Priority in Public Sector Agency,” *Procedia Comput. Sci.*, vol. 100, pp. 198–206, 2016, doi: 10.1016/j.procs.2016.09.141.
- [166] S. Toulmin, *The Uses of Argument*, 2nd ed. Cambridge University Press, 2003.
- [167] W. Kunz and H. W. J. Rittel, *Issues as elements of information systems*, vol. 9780203851. Berkeley: University of California, 2010.
- [168] J. Lee, “Extending the Potts and Bruns model for recording design rationale,” in *Proceedings - International Conference on Software Engineering*, 1991, pp. 114–125, doi: 10.1109/icse.1991.130629.
- [169] J. Tyree and A. Akerman, “Architecture decisions: Demystifying architecture,” *IEEE Softw.*, vol. 22, no. 2, pp. 19–27, 2005, doi: 10.1109/MS.2005.27.
- [170] J. Asundi, R. Kazman, and M. Klein, *Using Economic Considerations to Choose Among Architecture Design Alternatives*, no. December. Carnegie Mellon University, 2001.

- [171] G. Fischer, A. C. Lemke, R. McCall, and A. I. Morch, "Making Argumentation Serve Design," in *Human-Computer Interaction*, vol. 6, no. 3–4, 1991, pp. 393–419.
- [172] R. J. McCall, "PHI: a conceptual foundation for design hypermedia," *Des. Stud.*, vol. 12, no. 1, pp. 30–41, 1991, doi: 10.1016/0142-694X(91)90006-I.
- [173] S. T. Redwine and W. E. Riddle, "Software Technology Maturation," in *Proceedings - International Conference on Software Engineering*, 1985, pp. 189–200.
- [174] J. Conklin and M. L. Begeman, "gIBIS: A Hypertext Tool for Exploratory Policy Discussion," in *ACM Transactions on Information Systems (TOIS)*, 1988, vol. 6, no. 4, pp. 303–331, doi: 10.1145/58566.59297.
- [175] B. Ramesh and V. Dhar, "Supporting systems development by capturing deliberations during requirements engineering," *IEEE Trans. Softw. Eng.*, vol. 18, no. 6, pp. 498–510, 1992, doi: 10.1109/32.142872.
- [176] J. Lee and K. Y. Lai, "What's in Design Rationale?," *Human-Computer Interact.*, vol. 6, no. 3–4, pp. 251–280, 1991, doi: 10.1080/07370024.1991.9667169.
- [177] J. Lee, "SIBYL: A tool for managing group decision rationale," in *Proceedings of the 1990 ACM Conference on Computer-Supported Cooperative Work, CSCW 1990*, 1990, pp. 79–92, doi: 10.1145/99332.99344.
- [178] T. Maclean, A. Young, R.M., Bellotti, V. & Moran, "Questions_Options_and_Criteria_Elements_of_Design.pdf," in *Design Rationale: Concepts, Techniques and Use*, Lawrence Erlbaum Associates, 1991, pp. 53–106.
- [179] A. H. Dutoit and B. Paech, "Rationale-based use case specification," *Requir. Eng.*, vol. 7, no. 1, pp. 3–19, 2002, doi: 10.1007/s007660200001.
- [180] J. E. Burge and D. C. Brown, "Software Engineering Using RAtionale," *Journal of Systems and Software*, vol. 81, no. 3, pp. 395–413, 2008, doi: 10.1016/j.jss.2007.05.004.
- [181] G. Tel, *Introduction to Distributed Algorithms*. Cambridge University, 2000.
- [182] S. Tanenbaum, Andres and V. Steen, Marteen, "Distributed Computing Principles and Paradigms," *Creat. Indep. Publ. Platf.*, p. 702, 2016.
- [183] M. Steen and A. S. Tanenbaum, "A brief introduction to distributed systems," *Computing*, vol. 98, no. 10, pp. 967–1009, 2016.
- [184] A. Н. Максимец and С. Л. Крытый, "Верификация программ: состояние, проблемы, результаты," *Кибернетика и системный анализ*, vol. 49, pp. 3–14, 2013.
- [185] I. V Konnov, "Parameterized Model Checking by Network Invariants: the

Asynchronous Case,” in *Aiss*, vol. 23, Dubrovnik, Croatia, 2012, pp. 1–4.

- [186] B. Liu, *Theory and Practice of Uncertain Programming*. UTLAB, 2009.
- [187] В. В. Цуркан and В. В. Мохор, “Энтропийный подход к определению понятия «риск информационной безопасности» (information security risk),” in *XXVIII наук.-техн. конф. Моделирования*, 2009.
- [188] В. В. Мохор, С. М. Михайлов, and В. В. Цуркан, “Ентропійний підхід до оцінювання ризику безпеки інформації в кіберпросторі,” in *IV міжн. наук.-практ. конф. ITSEC*, 2014, p. 43.
- [189] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly, “Identifying, understanding, and analyzing critical infrastructure interdependencies,” *IEEE Control Syst. Mag.*, vol. 21, no. 6, pp. 11–25, 2001, doi: 10.1109/37.969131.
- [190] M. Ouyang, “Review on modeling and simulation of interdependent critical infrastructure systems,” *Reliab. Eng. Syst. Saf.*, vol. 121, pp. 43–60, 2014, doi: 10.1016/j.res.2013.06.040.
- [191] G. Satumtira and L. Dueñas-Orsorio, *Synthesis of modeling and simulation methods on critical infrastructure interdependencies research*. Berlin Heidelberg: Springer, 2010.
- [192] S. Hasan and G. Foliente, “Modeling infrastructure system interdependencies and socioeconomic impacts of failure in extreme events: emerging R&D challenges,” *Nat. Hazards*, vol. 78, no. 3, pp. 2143–2168, 2015, doi: 10.1007/s11069-015-1814-7.
- [193] G. Oliva, S. Panzieri, and R. Setola, “Fuzzy dynamic input-output inoperability model,” *Int. J. Crit. Infrastruct. Prot.*, vol. 4, no. 3–4, pp. 165–175, 2011, doi: 10.1016/j.ijcip.2011.09.003.
- [194] M. Kaegi, R. Mock, and W. Kröger, “Analyzing maintenance strategies by agent-based simulations: A feasibility study,” *Reliab. Eng. Syst. Saf.*, vol. 94, no. 9, pp. 1416–1421, 2009, doi: 10.1016/j.res.2009.02.002.
- [195] N. K. Svendsen and S. D. Wolthusen, *Graph models of critical infrastructure interdependencies*, vol. 4543 LNCS. Berlin Heidelberg: Springer, 2007.
- [196] S. Wang, L. Hong, M. Ouyang, J. Zhang, and X. Chen, “Vulnerability analysis of interdependent infrastructure systems under edge attack strategies,” *Saf. Sci.*, vol. 51, no. 1, pp. 328–337, 2013, doi: 10.1016/j.ssci.2012.07.003.
- [197] O. Gursesli and A. A. Desrochers, “Modeling infrastructure interdependencies using Petri nets,” *Proc. IEEE Int. Conf. Syst. Man Cybern.*, vol. 2, pp. 1506–1512, 2003, doi: 10.1109/icsmc.2003.1244625.
- [198] M. Beccuti, S. Chiaradonna, F. Di Giandomenico, S. Donatelli, G. Dondossola, and G. Franceschinis, “Quantification of dependencies between electrical and information infrastructures,” *Int. J. Crit. Infrastruct. Prot.*, vol. 5, no. 1, pp. 14–

27, 2012, doi: 10.1016/j.ijcip.2012.01.003.

- [199] A. Di Giorgio and F. Liberati, "A Bayesian network-based approach to the critical infrastructure interdependencies analysis," *IEEE Syst. J.*, vol. 6, no. 3, pp. 510–519, 2012, doi: 10.1109/JSYST.2012.2190695.
- [200] Я. Ю. Дорогий, "Ускоренный алгоритм обучения сверточных нейронных сетей," *Вісник НТУУ «КПІ», «Інформатика, управління та обчислювальна техніка»*, vol. 57, pp. 150–154, 2012.
- [201] Л. С. Глоба, О. М. Дяденко, and М. А. Скуліш, *Математичні основи побудови інформаційно-телекомунікаційних систем*. К: Норіта-плюс, 2007.
- [202] R. Kaur and A. Kaur, "A Review Paper on Evolution of Cloud Computing , its Approaches and Comparison with Grid Computing," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 5, pp. 6060–6063, 2014.
- [203] A. Gupta, D. Milojevic, and L. V. Kalé, "Optimizing VM placement for HPC in the cloud," *Fed. - Proc. 2012 Work. Cloud Serv. Fed. 8th Open Cirrus Summit, Co-located with ICAC'12*, vol. 16, pp. 1–6, 2012, doi: 10.1145/2378975.2378977.
- [204] R. Buyya, J. Broberg, and A. M. Goscinski, *Cloud Computing: Principles and Paradigms*. John Wiley & Sons, 2010.
- [205] X. Hu and R. Eberhart, "Tracking dynamic systems with PSO: where's the cheese?," in *Proc. of the Workshop on Particle Swarm Optimization*, 2001, pp. 80–83.
- [206] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, 1999, vol. 3, pp. 1945–1950, doi: 10.1109/CEC.1999.785511.
- [207] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the IEEE Conference on Evolutionary Computation, ICEC*, 2001, vol. 1, pp. 94–100, doi: 10.1109/cec.2001.934376.
- [208] A. Carlisle and G. Dozier, "Tracking changing extreme with adaptive particle swarm optimizer," in *Multimedia, Image Processing and Soft Computing: Trends, Principles and Applications - Proceedings of the 5th Biannual World Automation Congress, WAC 2002, ISSCI 2002 and IFMIP 2002*, 2002, vol. 13, pp. 265–270, doi: 10.1109/wac.2002.1049555.
- [209] W. Du and B. Li, "Multi-strategy ensemble particle swarm optimization for dynamic optimization," *Inf. Sci. (Ny)*, vol. 178, no. 15, pp. 3096–3109, 2008, doi: 10.1016/j.ins.2008.01.020.
- [210] M. F. Tasgetiren, M. Sevkli, Y. C. Liang, and G. Gencyilmaz, "Particle swarm optimization algorithm for permutation flowshop sequencing problem," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2004, vol. 3172

LNCS, pp. 382–389, doi: 10.1007/978-3-540-28646-2_38.

- [211] M. F. Tasgetiren, Y. C. Liang, M. Sevkli, and G. Gencyilmaz, “Particle swarm optimization algorithm for single machine total weighted tardiness problem,” in *Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004*, 2004, vol. 2, pp. 1412–1419, doi: 10.1109/cec.2004.1331062.
- [212] Кабінет Міністрів України, “Постанова №1024 ‘Про затвердження Положення про єдину інформаційну систему Міністерства внутрішніх справ та переліку її пріоритетних інформаційних ресурсів’ від 14.11.2018 р.” [Online]. Available: <https://zakon.rada.gov.ua/laws/show/1024-2018-п>. [Accessed: 01-Mar-2020].
- [213] J. Lee, S. Bohacek, J. P. Hespanha, and K. Obraczka, “Modeling communication networks with hybrid systems,” *IEEE/ACM Trans. Netw.*, vol. 15, no. 3, pp. 630–643, 2007, doi: 10.1109/TNET.2007.893090.
- [214] ISO/IEC JTC 1, “ISO/IEC 17789:2014 Information Technology — Cloud Computing — Reference Architecture,” *ISO/IEC Information Technology Task Force (ITTF) web site*, 2014. [Online]. Available: <https://www.iso.org/obp/ui/#iso:std:iso-iec:17789:ed-1:v1:en>. [Accessed: 09-Nov-2017].
- [215] “NIST SP 500-291 Cloud Computing Standards Roadmap, 2011-07-05.pdf.” [Online]. Available: http://www.nist.gov/itl/cloud/upload/NIST_SP-500-291_Version-2_2013_June18_FINAL.pdf. [Accessed: 09-Nov-2017].
- [216] F. Liu *et al.*, “NIST Special Publication 500-292: Cloud Computing Reference Architecture,” 2011. [Online]. Available: http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=909505. [Accessed: 09-Nov-2017].
- [217] IsecT, “ISO / IEC 27017 - Information technology - Security techniques - Code of practice for information security controls based on ISO / IEC 27002 for cloud services (DRAFT),” *IsecT*, 2014. [Online]. Available: <http://www.iso27001security.com/html/27017.html>. [Accessed: 10-Oct-2017].
- [218] ISO/IEC, “ISO/IEC 27018:2019 Information technology — Security techniques — Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors,” 2019. [Online]. Available: <https://www.iso.org/obp/ui/#iso:std:iso-iec:27018:ed-2:v1:en>. [Accessed: 02-Feb-2020].
- [219] “BSI Standards Publication Information technology — Security techniques — Information security for supplier relationships Part 4 : Guidelines for security of cloud services,” 2016. [Online]. Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=59689. [Accessed: 15-Oct-2018].
- [220] “ISO/IEC 27040 Information technology – Security techniques – Storage

security.” [Online]. Available: http://www.iso.org/iso/iso_catalogue/. [Accessed: 20-Oct-2017].

- [221] Deepak K. Vij and David Bernstein, “P2302 Draft Standard for Intercloud Interoperability and Federation (SIIF),” *IEEE Standards Draft*, 2012. [Online]. Available: <https://www.oasis-open.org/committees/>. [Accessed: 25-Oct-2017].
- [222] C. DiMinico, “Telecommunications Infrastructure Standard for Data Centers,” in *MC Communications*, no. August, Arlington, 2010, p. 36.
- [223] GlobalSpec, “CENELEC - EN 50173-5 - Information technology - Generic cabling systems - Part 5: Data centres | Engineering360,” in *Comité Européen de Normalisation Électrotechnique (CENELEC)*, Brussels, 2016.
- [224] IEC, “ISO/IEC 24764 - Information technology – Generic cabling systems for data centres,” 2010. [Online]. Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=. [Accessed: 25-Oct-2017].
- [225] Ж. В. Шелиманова, А. А. Фурманов, and О. В. Яновская, “Таксономическая схема архитектур облачных систем,” *Радіоелектронні і комп’ютерні системи*, vol. 74, no. 4, pp. 51–55, 2015.
- [226] Enterasys, “Data Center Networking – Connectivity and Topology Design Guide,” 2011. [Online]. Available: <https://www.cdigroup.co.uk/wp-content/pdf-documents/data-centers/Enterasys-Data-Center-Design-Guide.pdf%0Ahttp://www.enterasys.com/company/literature/datacenter-design-guide-wp.pdf>. [Accessed: 20-Oct-2019].
- [227] Microsoft, “SLA for App Service,” *Microsoft Azure Website*, 2016. [Online]. Available: https://azure.microsoft.com/en-gb/support/legal/sla/app-service/v1_4/. [Accessed: 30-Oct-2017].
- [228] P. Murray and El, “Cloud Networking Architecture Description,” *Scalable Adapt. Internet Solut.*, pp. 14–69, 2012.
- [229] S. B. Thrun *et al.*, “The monk’s problems: a performance comparison of different learning algorithms,” 1991.
- [230] K. J. Lang and M. J. Witbrock, “Larning to Tell Two Spirals Apart,” in *Connectionist Models Summer School*, 1988, pp. 52–59.
- [231] A. S. Georghiades, “The Yale Face Database B,” 2001. [Online]. Available: http://vision.ucsd.edu/~iskwak/ExtYaleDatabase/Yale_Face_Database.htm. [Accessed: 01-Mar-2020].
- [232] D. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, vol. 27, no. 02. Addison-Wesley, 1989.
- [233] R. Sookhtsaraei, M. Madani, and A. Kavian, “A multi objective virtual machine placement method for reduce operational costs in cloud computing by genetic,”

- [234] G. Wu, M. Tang, Y. C. Tian, and W. Li, “Energy-efficient virtual machine placement in data centers by genetic algorithm,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, vol. 7665 LNCS, no. PART 3, pp. 315–323, doi: 10.1007/978-3-642-34487-9_39.
- [235] G. Hinton and L. V. Maaten, “Visualizing High-Dimensional Data Using t-SNE,” *J. Mach. Learn. Res.*, vol. 9, no. Nov, pp. 2579–2605, 2008.
- [236] Верховна Рада України, “Закон №5492-VI ‘Про Єдиний державний демографічний реєстр та документи, що підтверджують громадянство України, посвідчують особу чи її спеціальний статус’ від 20.11.2012 р.” [Online]. Available: <https://zakon.rada.gov.ua/laws/card/5492-1>. [Accessed: 16-Sep-2019].
- [237] Кабінет Міністрів України, “Постанова №606 ‘Деякі питання електронної взаємодії державних електронних інформаційних ресурсів’ від 08.09.2016 р.” [Online]. Available: <https://zakon.rada.gov.ua/laws/show/606-2016-п>. [Accessed: 16-Sep-2019].
- [238] The Office for Civil Rights (OCR) and B. Malin, “Guidance Regarding Methods for de-identification of protected health information in accordance with the Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule,” *Health Information Privacy*, 2012. [Online]. Available: <https://www.hhs.gov/hipaa/for-professionals/privacy/special-topics/de-identification/index.html>. [Accessed: 19-Sep-2019].
- [239] J. B. Freymann, J. S. Kirby, J. H. Perry, D. A. Clunie, and C. C. Jaffe, “Image data sharing for biomedical research - Meeting HIPAA requirements for de-identification,” *J. Digit. Imaging*, vol. 25, no. 1, pp. 14–24, 2012, doi: 10.1007/s10278-011-9422-x.
- [240] Ö. Uzuner, Y. Luo, and P. Szolovits, “Evaluating the State-of-the-Art in Automatic De-identification,” *J. Am. Med. Informatics Assoc.*, vol. 14, no. 5, pp. 550–563, 2007, doi: 10.1197/jamia.M2444.
- [241] K. Emam *et al.*, “De-identification methods for open health data: The case of the heritage health prize claims dataset,” *J. Med. Internet Res.*, vol. 14, no. 1, 2012, doi: 10.2196/jmir.2001.
- [242] B. S. Elger *et al.*, “Strategies for health data exchange for secondary, cross-institutional clinical research,” *Comput. Methods Programs Biomed.*, vol. 99, no. 3, pp. 230–251, 2010, doi: 10.1016/j.cmpb.2009.12.001.
- [243] National Institutes of Health, “Clinical Research and the HIPAA Privacy Rule,” *NIH Publication Number 04-5495*, 2004. [Online]. Available: <https://privacyruleandresearch.nih.gov/>. [Accessed: 16-Sep-2019].

- [244] L. Ohno-Machado and etc, “iDASH: integrating data for analysis, anonymization, and sharing,” *Am Med Inf. Assoc.*, vol. 19, no. 2, pp. 196–201, 2012.
- [245] K. Benitez and B. Malin, “Evaluating re-identification risks with respect to the HIPAA privacy rule,” *J. Am. Med. Informatics Assoc.*, vol. 17, no. 2, pp. 169–177, 2010, doi: 10.1136/jamia.2009.000026.
- [246] C. Quantin, M. Fassa, G. Coatrieux, B. Riandey, G. Trouessin, and F. A. Allaert, “Linking anonymous databases for national and international multicenter epidemiological studies: A cryptographic algorithm,” *Rev. Epidemiol. Sante Publique*, vol. 57, no. 1, pp. 33–39, 2009, doi: 10.1016/j.respe.2008.11.002.
- [247] S. B. Johnson *et al.*, “Using global unique identifiers to link autism collections,” *J. Am. Med. Informatics Assoc.*, vol. 17, no. 6, pp. 689–695, 2010, doi: 10.1136/jamia.2009.002063.
- [248] Кабінет Міністрів України, “Постанова №1078 ‘Про реалізацію експериментального проекту верифікації даних про фізичних осіб, що обробляються в деяких національних електронних інформаційних ресурсах’ від 04.12.2019 р.” [Online]. Available: <https://zakon.rada.gov.ua/laws/show/1078-2019-п>. [Accessed: 10-Jan-2020].
- [249] R. Boutaba, J. Xiao, and I. Aib, “CyberPlanner: A comprehensive toolkit for network service providers,” in *NOMS 2008 - IEEE/IFIP Network Operations and Management Symposium: Pervasive Management for Ubiquitous Networks and Services*, 2008, pp. 379–386, doi: 10.1109/NOMS.2008.4575158.
- [250] International Organization for Standardization (ISO), “Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model,” *Int. Stand. ISO/IEC 7498-1*, vol. 1, no. 1994, p. 59, 1996.
- [251] ITU-T, “Principles for a telecommunications management network,” *ITU-T Recomm. M3010*, vol. 3, no. 1, pp. 39–68, 2000.
- [252] С. Катышев, “Об одной концепции управления распределенным ресурсами,” *Открытые системы*, no. 3, 1998.
- [253] ISO/IEC, “Service management,” in *Information technology. Service management. Part 2: Code of practice: ISO/IEC 20000-1:2005*, 2005, p. 34.
- [254] E. Hubbert, J. Garbani, G. O’Donnell, S. Mann, and J. Rakowski, *TechRadar™ For I&O Professionals: IT Service Management Processes*. Forrester Research, Inc, 2012.
- [255] I. Jeffreys, “The use of compensatory and non-compensatory multi-criteria analysis for small-scale forestry,” *Small-scale For. Econ. Manag. Policy*, vol. 3, no. 1, pp. 99–117, 2004, doi: 10.1007/s11842-004-0007-0.
- [256] T. Elrod, R. D. Johnson, and J. White, “A new integrated model of noncompensatory and compensatory decision strategies,” *Organ. Behav. Hum.*

Decis. Process., vol. 95, no. 1, pp. 1–19, Sep. 2004, doi: 10.1016/j.obhdp.2004.06.002.

- [257] L. Rothrock and J. Yin, “Integrating Compensatory and Noncompensatory Decision-Making Strategies in Dynamic Task Environments,” *Decis. Model. Behav. Complex Uncertain Environ.*, pp. 125–141, 2008, doi: 10.1007/978-0-387-77131-1_6.
- [258] B. Alenljung and A. Persson, “Portraying the practice of decision-making in requirements engineering: A case of large scale bespoke development,” *Requir. Eng.*, vol. 13, no. 4, pp. 257–279, 2008, doi: 10.1007/s00766-008-0068-2.
- [259] J. Orasanu and T. Connolly, “The Reinvention of Decision Making,” *Decis. Mak. action Model. Methods*, pp. 3–20, 1993.
- [260] The Open Group, “ArchiMate® 3.0.1 Specification,” 2017. [Online]. Available: <https://pubs.opengroup.org/architecture/archimate301-doc/>. [Accessed: 29-Oct-2017].
- [261] A. Cimatti *et al.*, “NuSMV 2: An opensource tool for symbolic model checking,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2002, vol. 2404, pp. 359–364, doi: 10.1007/3-540-45657-0_29.
- [262] “PyNUSMV.” [Online]. Available: <https://github.com/sbusard/pynusmv>. [Accessed: 01-Mar-2020].
- [263] S. Izza, “Integration of industrial information systems: From syntactic to semantic integration approaches,” *Enterp. Inf. Syst.*, vol. 3, no. 1, pp. 1–57, 2009, doi: 10.1080/17517570802521163.
- [264] J. Roy and A. Ramanujan, “Understanding web services,” *IT Prof.*, vol. 3, no. 6, pp. 69–73, 2001, doi: 10.1109/6294.977775.
- [265] M. D. Hanes, S. C. Ahalt, and A. K. Krishnamurthy, “A Comparison of Java RMI , CORBA , and Web Services Technologies for Distributed SIP Applications Department of Electrical Engineering The Ohio State University,” in *High Performance Embedded Computing Annual Workshop (HPEC)*, 2002, pp. 2–3.
- [266] S. Vinoski, “Integration with Web Services,” *IEEE Internet Comput.*, vol. 7, no. 6, pp. 75–77, 2003, doi: 10.1109/MIC.2003.1250587.
- [267] W. Emmerich, “Software engineering and middleware: A roadmap,” *Proc. Conf. Futur. Softw. Eng. ICSE 2000*, pp. 117–129, 2000, doi: 10.1145/336512.336542.
- [268] ISO/IEC, “Open Distributed Processing Reference Model (RM-ODP) Part 3.” Architecture, ISO/IEC, 1995.
- [269] D. Project, “DEAP Documentation,” 2013. [Online]. Available: <https://deap.readthedocs.io/en/master/>. [Accessed: 01-Mar-2020].
- [270] Django Software Foundation, “The Web framework for perfectionists with

- deadlines | Django,” *Django Software Foundation*, 2018. [Online]. Available: <https://www.djangoproject.com/>. [Accessed: 01-Feb-2020].
- [271] “Swarm mode overview,” 2019. [Online]. Available: <https://docs.docker.com/engine/swarm/>. [Accessed: 01-Mar-2020].
- [272] “vSphere.” [Online]. Available: <https://www.vmware.com/products/vsphere.html>. [Accessed: 01-Mar-2020].
- [273] “Node.js.” [Online]. Available: <https://nodejs.org/uk/>. [Accessed: 01-Mar-2020].
- [274] “Electron | Создавайте кросс-платформенные приложения при помощи JavaScript, HTML и CSS,” *Electron*. [Online]. Available: <https://www.electronjs.org/>. [Accessed: 01-Mar-2020].
- [275] Chromium, “Chromium - The Chromium Projects,” 2019. [Online]. Available: <https://www.chromium.org/Home>. [Accessed: 01-Mar-2020].
- [276] Docker Inc, “Empowering App Development for Developers | Docker.” [Online]. Available: <https://www.docker.com/>. [Accessed: 01-Mar-2020].
- [277] HashiCorp, “Vagrant.” [Online]. Available: <https://www.vagrantup.com/>. [Accessed: 01-Mar-2020].
- [278] Grafana Labs, “The open observability platform.” [Online]. Available: <https://grafana.com/>. [Accessed: 01-Mar-2020].
- [279] “Prometheus.” [Online]. Available: <https://prometheus.io/>. [Accessed: 01-Mar-2020].
- [280] “Consul.” [Online]. Available: <https://www.consul.io/>. [Accessed: 01-Mar-2020].
- [281] M. DeHaan, “Ansible is Simple IT Automation.,” *Sponsored by Red Hat*, 2012. [Online]. Available: <https://www.ansible.com/>. [Accessed: 01-Mar-2020].
- [282] Redux.js, “A predictable state container for JavaScript apps.,” 2019. [Online]. Available: <https://redux.js.org/>. [Accessed: 01-Mar-2020].
- [283] “Store.” [Online]. Available: <https://redux.js.org/api/store/>. [Accessed: 01-Mar-2020].
- [284] “Grommet.” [Online]. Available: <https://v2.grommet.io/>. [Accessed: 01-Mar-2020].
- [285] “Flexbox - изучение веб-разработки.” [Online]. Available: https://developer.mozilla.org/ru/docs/Learn/CSS/CSS_layout/Flexbox. [Accessed: 01-Mar-2020].
- [286] “docker stack deploy.” [Online]. Available: https://docs.docker.com/engine/reference/commandline/stack_deploy/. [Accessed: 01-Mar-2020].

ДОДАТОК А

ІЛЮСТРАЦІЙНІ МАТЕРІАЛИ

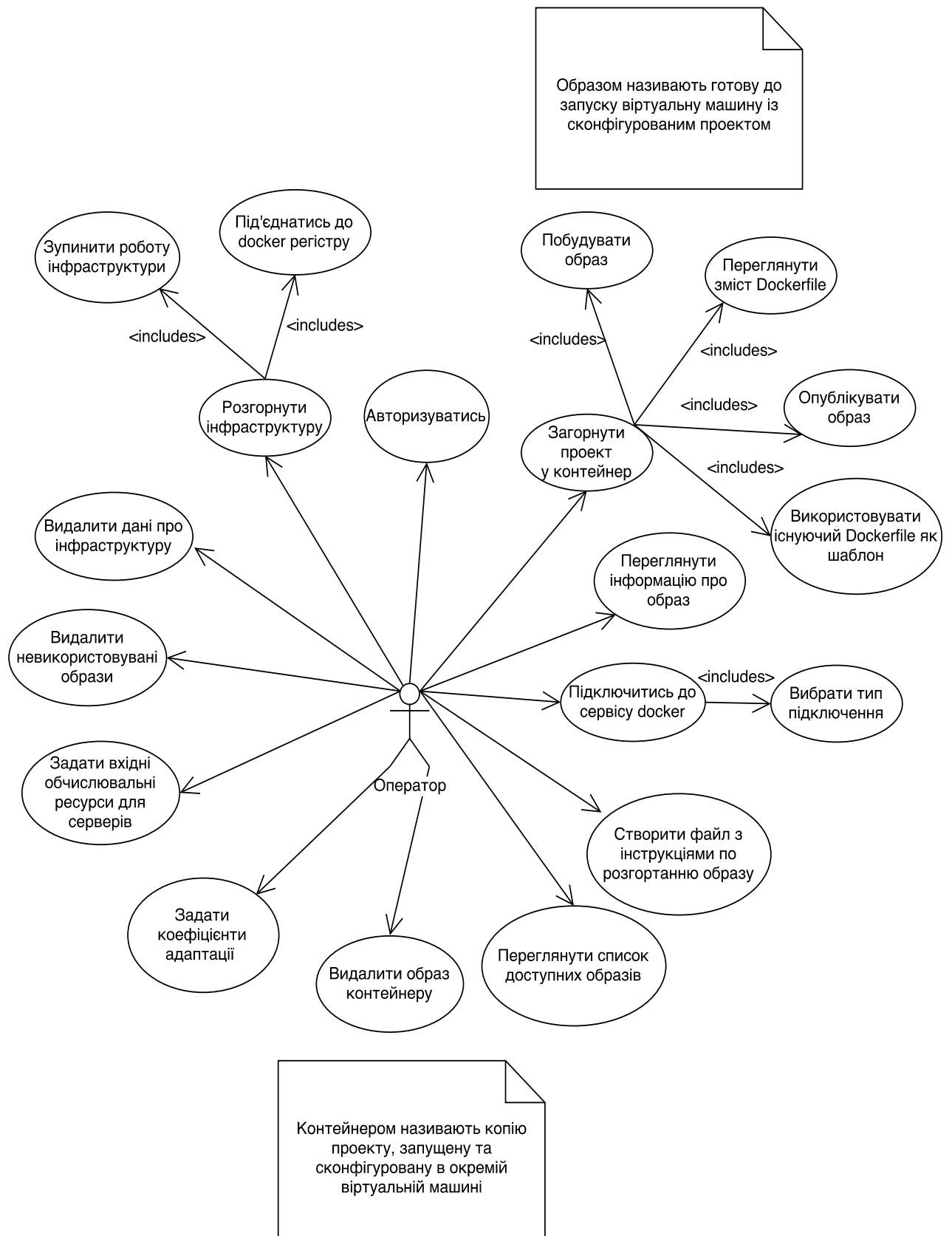


Рис. А.1. Діаграма юз кейсів оператора

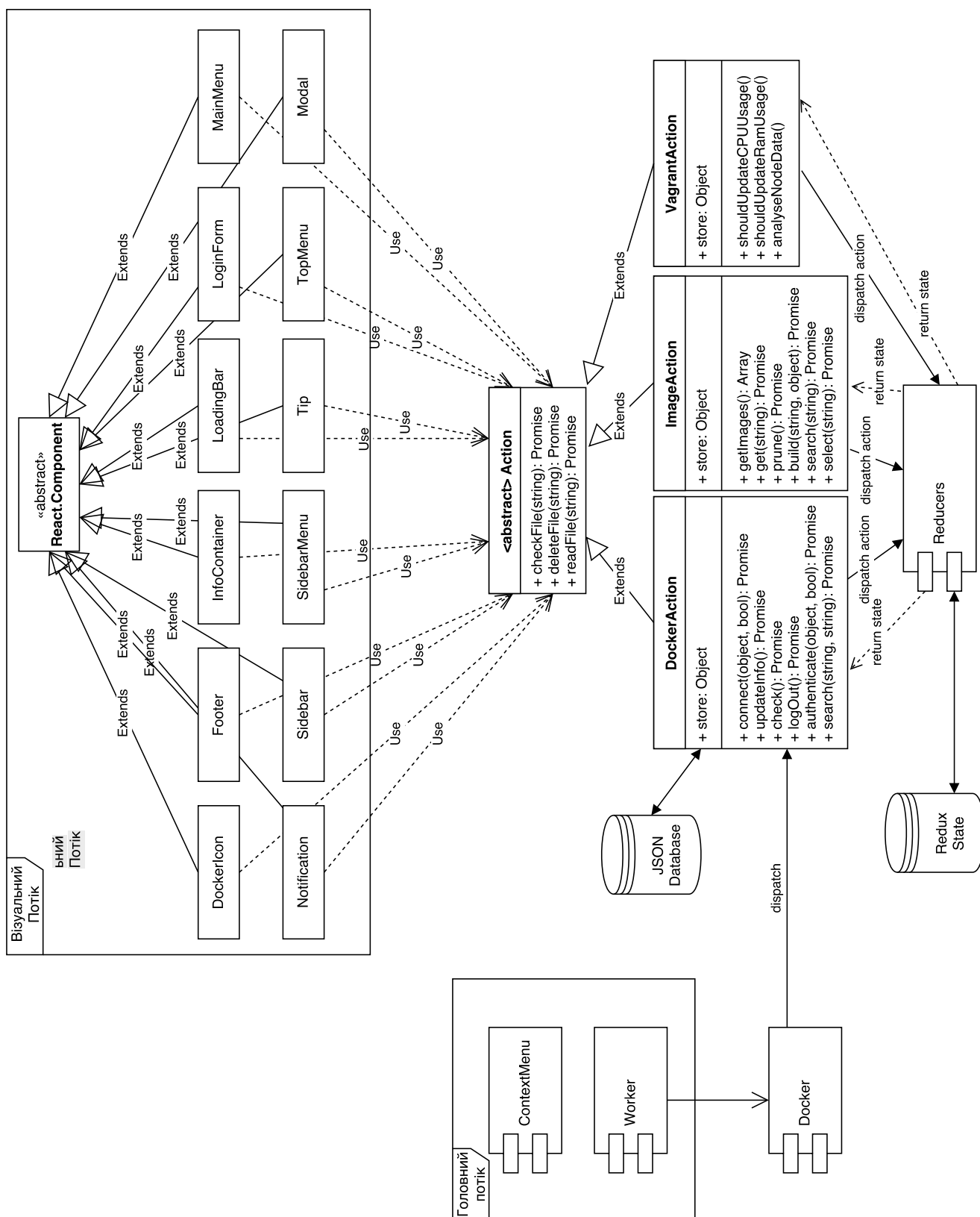


Рис. А.2. Діаграма класів АРМ-О

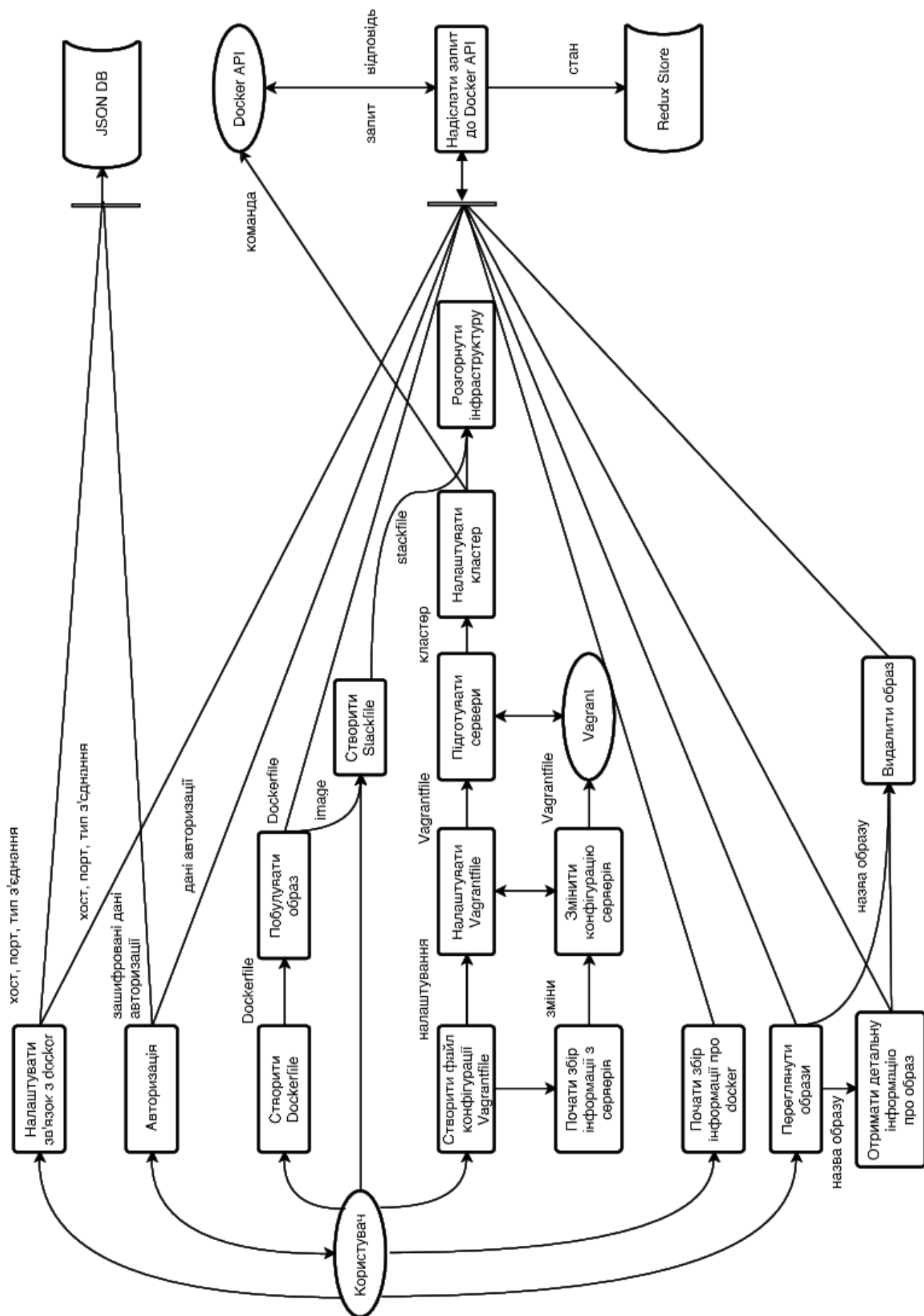


Рис. А.3. Діаграма потоку

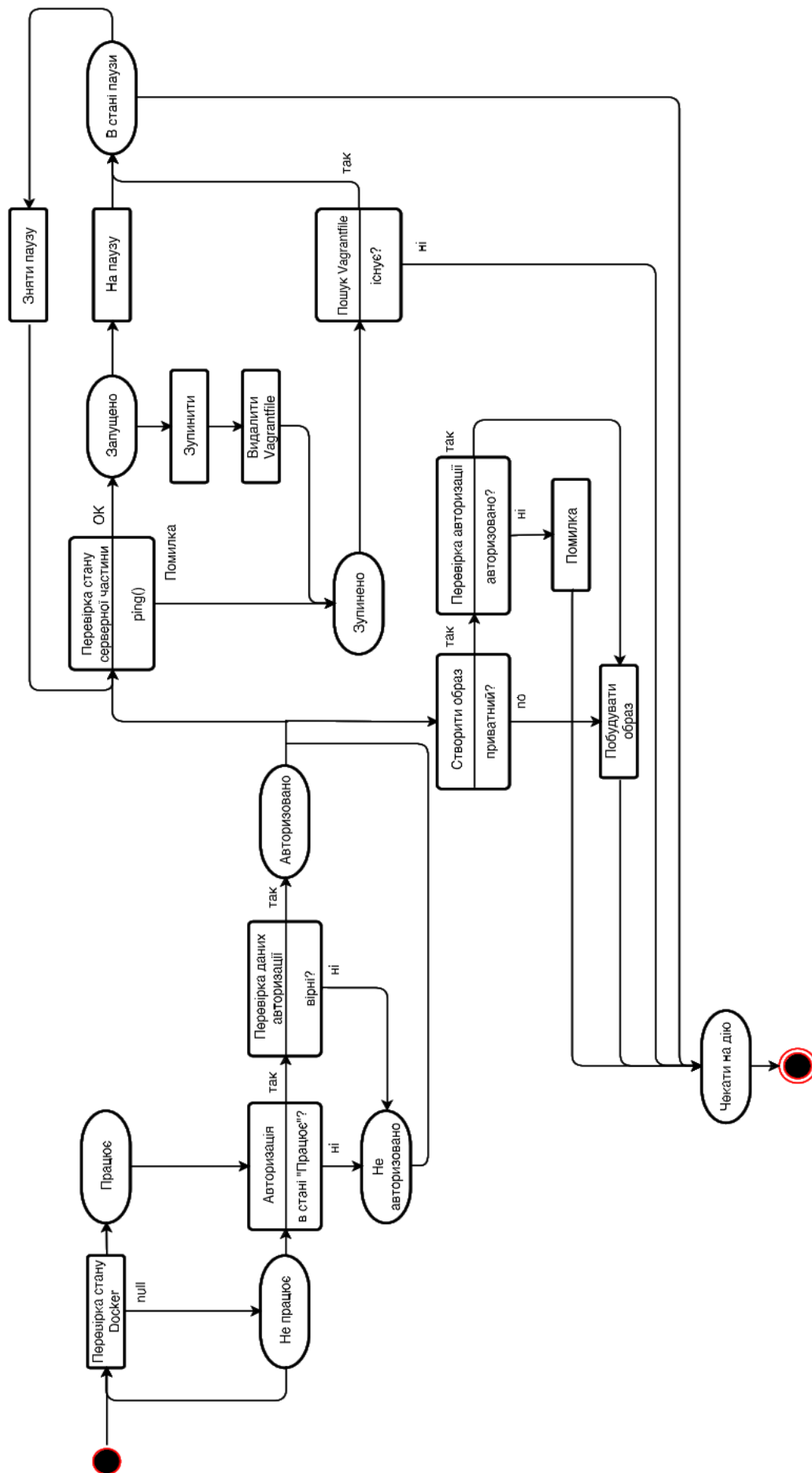


Рис. А.4. Діаграма станів

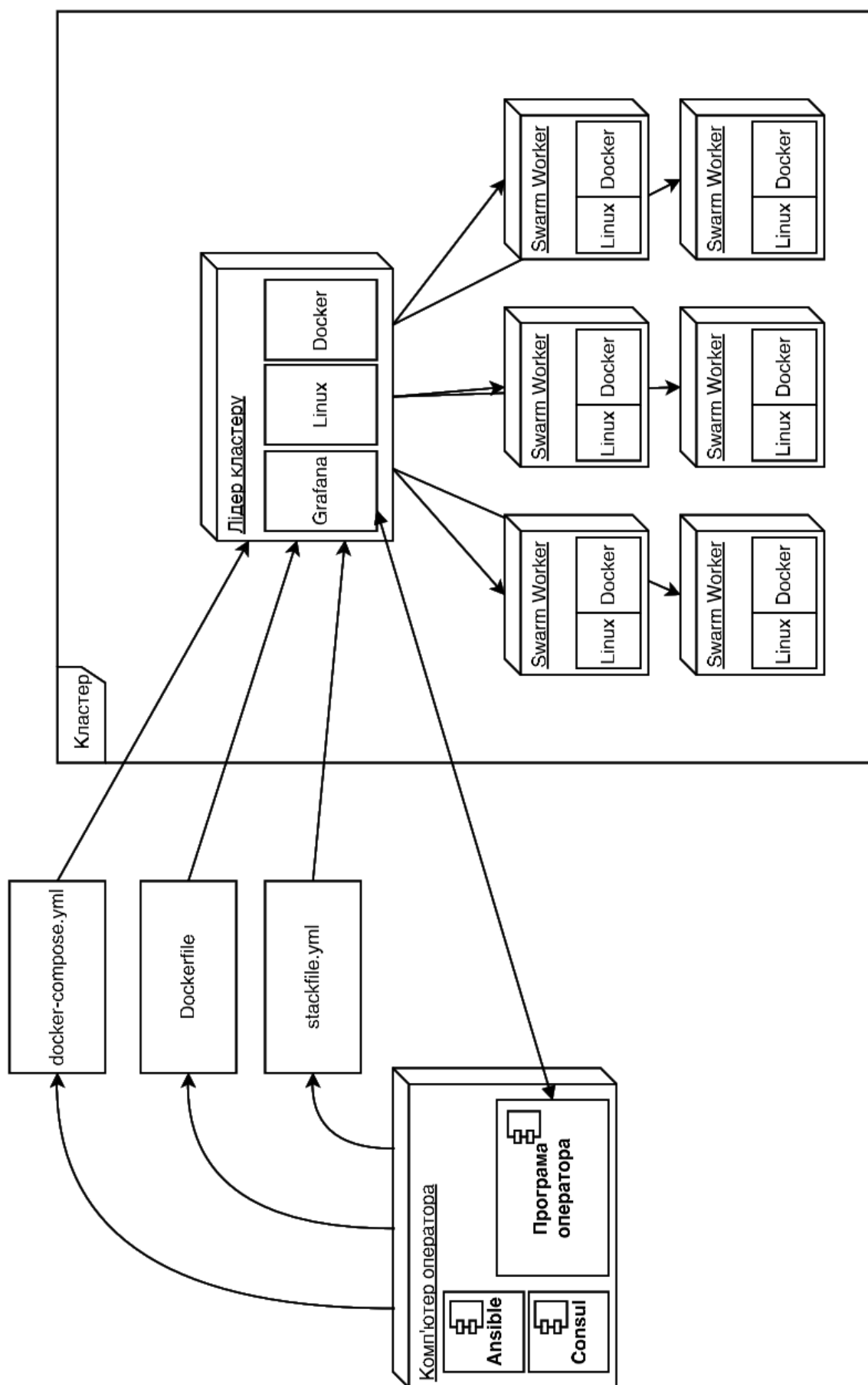


Рис. А.5. Інфраструктура системи розгортання критичної інформаційної інфраструктури

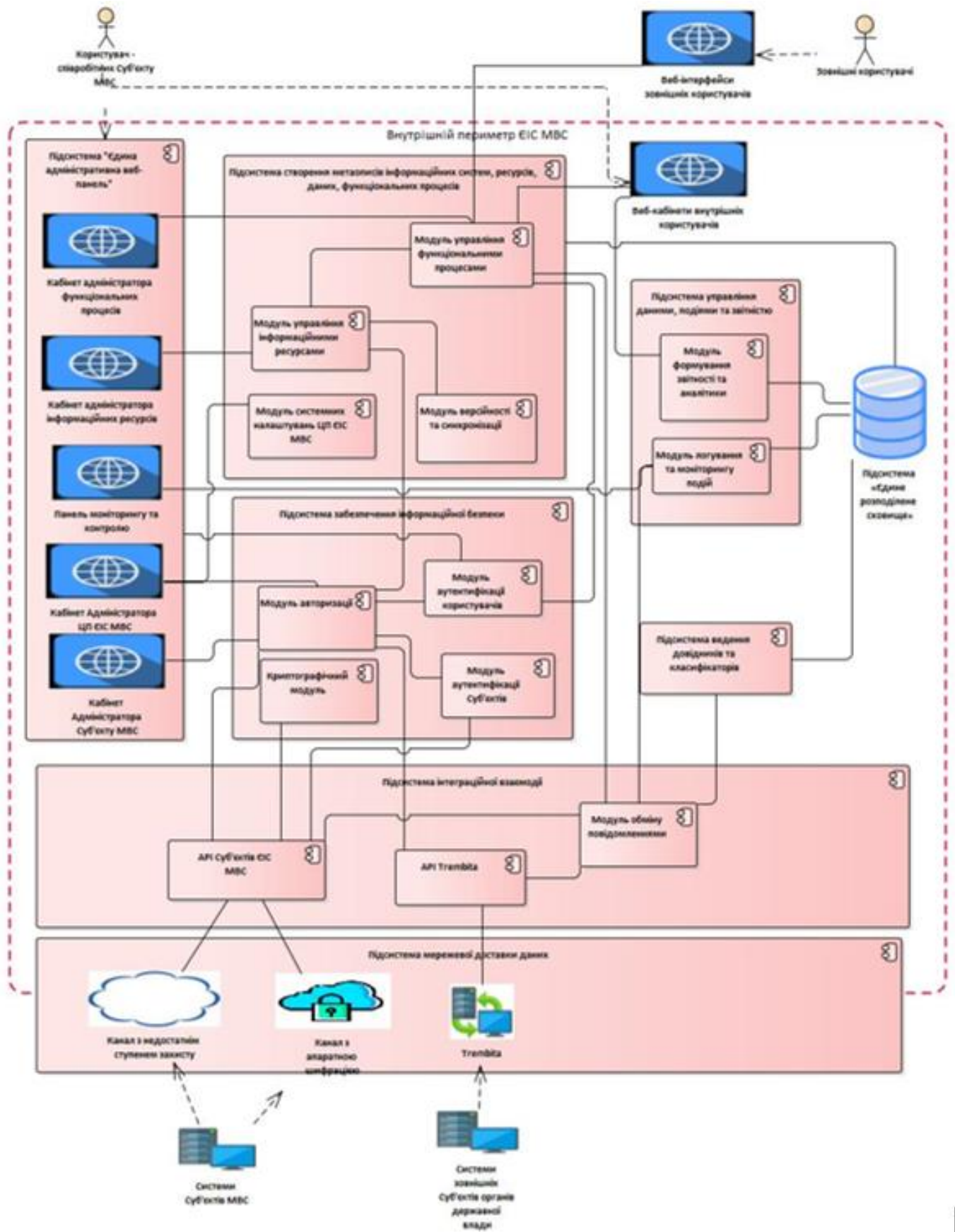


Рис. А.6. Функціональна структура ЦП ЄС MBC

ДОДАТОК Б

ФУНКЦІОНАЛ СИСТЕМИ РОЗГОРТАННЯ КРИТИЧНОЇ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ

Б.1 Розробка функціоналу для створення образу контейнера

Для того, щоб створити новий Dockerfile, оператор вибирає відповідне меню на головній сторінці програми (рис. Б.1). Далі, відмічаючи команди, які оператору необхідно включити в Dockerfile, він створює структуру майбутнього файлу.

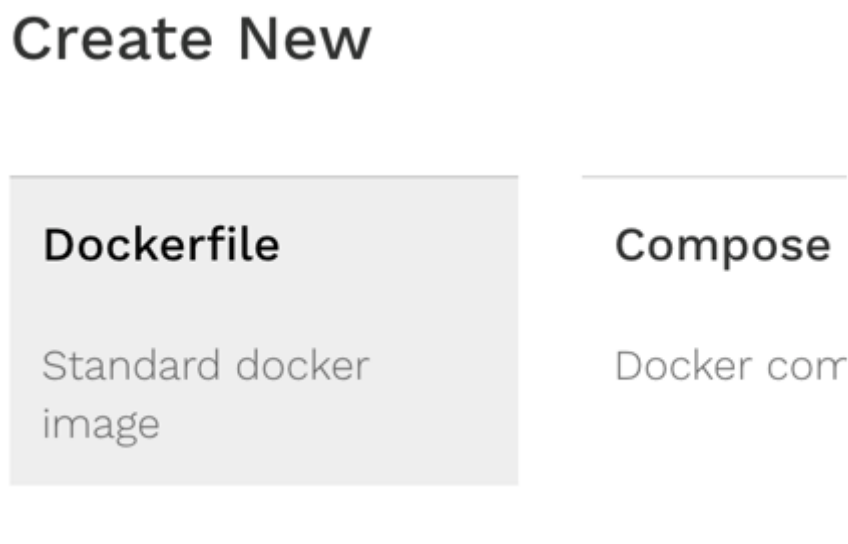


Рис. Б.1. Меню вибору режиму створення образу контейнеру

Доступні параметри до налаштування:

- FOM – вибір образу-шаблону – Docker Image, на основі якого буде базуватись сервіс. Заздалегідь використовується образ, створений розробниками мови програмування, яку було використано при розробці програми. Так, для node.js це буде node:latest, для java 8 – java:8-alpine відповідно;
- RUN – команда, яку буде виконано у терміналі контейнеру;
- EXPOSE – відкриті порти;
- ENV – змінні середовища;

- ADD – скопіювати файли у контейнер за їх шляхом у файловій системі або за посиланням;
- COPY – скопіювати локальні файли у контейнер;
- VOLUME – вказання директорії для збереження файлів, необхідних для запуску;
- WORKDIR – вказує кореневий каталог для виконання команд;
- ARG – передача параметрів у контейнер;
- SHELL – виконання скрипту;
- CMD – кінцева інструкція – команда для запуску сервісу чи програми всередині контейнеру.

The screenshot shows the 'Docker Image' configuration window. At the top, there are icons for navigation and a title bar. Below the title bar is a toolbar with icons for Destination, Save, Preview, Delete, Build, and Push. The main area is divided into two sections: 'Contents' on the left and a list of build options on the right.

Contents:

- FROM:** dialonce/nodejs
- LABEL:** test-app
- EXPOSE:** 8000
- ENV:** NODE_ENV=production
- VOLUME:** /data

Build Options (Right Side):

- ☐ RUN
- ☒ LABEL
- ☐ ADD
- ☐ COPY
- ☐ ENTRY POINT
- ☒ VOLUME
- ☐ USER
- ☐ WORKDIR
- ☐ ARG
- ☐ ONBUILD
- ☐ STOP SIGNAL
- ☐ HEALTHCHECK
- ☐ SHELL

Рис. Б.2. Форма створення образу докер контейнеру

Заповнивши необхідні текстові поля (рис. Б.2) є можливість побудувати файл. Візуальний потік нагромаджує усі дані, які оператор заповнює у формі, а

коли він натискає кнопку «Побудувати», візуальний потік надсилає запит у IPC-канал на побудову нового Dockerfile та передає дані з форми. Основний потік проводить валідацію на відповідність формату Dockerfile і коригує їх у разі необхідності. В результаті в кінцевій папці з'являється валідний Dockerfile (рис. Б.3). Основний потік надсилає відповідь про успішну побудову файлу до каналу зв'язку, візуальний потік розблоковує кнопку побудови Docker Image.

```
FROM node:latest

RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

COPY package.json /usr/src/app
RUN npm install

COPY . /usr/src/app

EXPOSE 8080

CMD ["npm", "start"]
```

Рис. Б.3. Зразок інструкцій Dockerfile

Після натискання кнопки візуальний інтерфейс за допомогою модуля `dockerode` надсилає запит до `docker` сервісу про побудову нового образу, передаючи в якості параметру інструкції з Dockerfile. В залежності від наявності доступу у оператора до образу-шаблону та правильності вводу команд, буде створений новий образ контейнеру. Для того, щоб образ став доступний публічно, його можна опублікувати у репозиторії образів – `Dockerhub`. Для цього оператор натискає відповідну підсвічену кнопку і комунікація між двома потоками проходить за тим же самим алгоритмом.

Б.2 Розробка функціоналу для створення Docker Stackfile

Дана підсистема містить функціонал для створення файлу конфігурації кластеру. Оператор вибирає відповідне меню на головній сторінці сервісу (рис. Б.4), далі створює необхідну кількість та необхідний тип мереж, розмір дискового простору для збереження важливих даних, а потім створює необхідні сервіси, встановлює залежність між ними, вибирає які мережі із раніше встановлених будуть їм доступними, також вибирає шляхи до дискового простору, куди вони будуть зберігати свої дані (рис. Б.5).

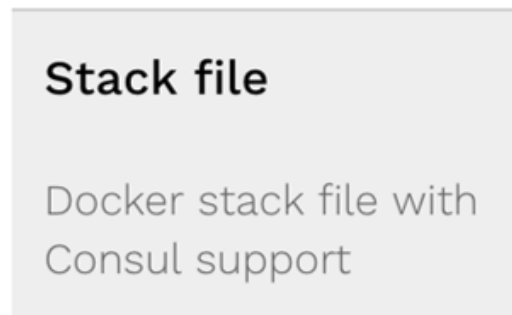


Рис. Б.4. Меню вибору режиму побудови Stackfile

Коли оператор натискає кнопку «Побудувати», візуальний потік надсилає дані з форми до основного потоку, де вони за допомогою парсеру `yaml` переводяться із об'єкту мови Javascript у валідний `yaml` файл та зберігається у вибраному каталозі проекту. Якщо на момент вибору каталогу знайдено існуючий Stackfile, він буде взятий за шаблон, оператор зможе його відредагувати та внести необхідні зміни, після чого його буде перезаписано.

Destination Save Preview Delete

Configuration

+ Service + Network + Volume

Networks

proxy ☒ external	stack ☐ external	log ☐ external

Volumes

volume1	volume2

Services

service_test	service_dependency

Рис. Б.5. Форма створення Stackfile

Зразок файлу stackfile наведено нижче:

Вказується, що синтаксис файлу відповідає 3й версії синтаксису. Версія синтаксису впливає на версію докеру, який необхідний для розпізнавання змісту, а також на список підтримуваних команд.

version: "3"

Вказуються мережі, які стануть доступні для використання сервісами
networks:

proxy:

external: true

stack:

alias: s

log:

Резервується дисковий простір для сервісів

volumes:

volume1:

volume2:

Прописуються налаштування для кожного з сервісів. Вказується назва, команда для запуску, режим запуску, змінні мережі, образ, на основі якого буде

створено сервіс, список мереж, які будуть доступні сервісу, відкриті порти, зарезервований дисковий простір.

services:

service_test:

command: '-cookie-secure=false'

deploy_mode:

replicated:

replicas: '2'

environment:

- NODE_ENV=production

image: couchbase

networks:

- proxy

- stack

ports:

- '1111:1111'

volumes:

- 'volume1:volume1'

Налаштування записуються для кожного сервісу окремо

service_dependency:

command: '-test=false'

depends_on:

- service_test

deploy_mode:

replicated:

replicas: '3'

environment:

- TEST=false

- ELSE=false

- WOW=true

image: dialonce/nodejs

networks:

- proxy

ports:

- '2222:2222'

- '1111:1111'

volumes:

- 'volume2:volume2'

Б.3 Розробка інтерфейсу для роботи з Vagrant-сервісом

Розглянемо процес конфігурації. Оператор відкриває форму створення Vagrant інфраструктури. На формі доступне меню для запуску, перезапуску, паузи та видалення інфраструктури, а також компоненти візуального інтерфейсу для налаштування виділених обчислювальних ресурсів, які будуть використані кожним із серверів.

Оператор вказує необхідні величини для об'єму доступної оперативної пам'яті, кількості ядер, відсотку максимального навантаження на ядро. Після цього, інфраструктура стає готовою для розгортання. В момент розгортання, візуальний потік передає управління на основний потік, адже vagrant працює лише через термінал (рис. Б.6), а візуальний потік не має до нього доступу.

```

Bringing machine 'manager' up with 'virtualbox' provider...
Bringing machine 'worker1' up with 'virtualbox' provider...
Bringing machine 'worker2' up with 'virtualbox' provider...
==> manager: Importing base box 'ubuntu/trusty64'...
==> manager: Matching MAC address for NAT networking...
==> manager: Checking if box 'ubuntu/trusty64' is up to date...
==> manager: Setting the name of the VM: templates_manager_1511828174983_28614
==> manager: Clearing any previously set forwarded ports...
==> manager: Clearing any previously set network interfaces...
==> manager: Preparing network interfaces based on configuration...
manager: Adapter 1: nat
manager: Adapter 2: hostonly
==> manager: Forwarding ports...
manager: 22 (guest) => 2222 (host) (adapter 1)
==> manager: Running 'pre-boot' VM customizations...
==> manager: Booting VM...
==> manager: Waiting for machine to boot. This may take a few minutes...
manager: SSH address: 127.0.0.1:2222
manager: SSH username: vagrant
manager: SSH auth method: private key
manager:
manager: Vagrant insecure key detected. Vagrant will automatically replace
manager: this with a newly generated keypair for better security.
manager:
manager: Inserting generated public key within guest...
manager: Removing insecure key from the guest if it's present...
manager: Key inserted! Disconnecting and reconnecting using new SSH key...
==> manager: Machine booted and ready!

```

Рис. Б.6. Процес автоматичного запуску та налаштування кластеру

Разом із повідомленням передаються встановлені бажані параметри для кожного із серверів. За основу конфігурації береться шаблонний Vagrantfile із наступними налаштуваннями:

- за провайдер віртуальної машини взято Virtualbox;
- за замовчуванням змінна середовища AUTO_START_SWARM отримує значення false, що свідчить про те, що конфігурацію docker swarm буде пропущено. Щоб задіяти автоматичну конфігурацію достатньо лише ініціалізувати таку змінну середовища;
- у файл hosts записуються ip-адреси приватної мережі кожного із серверів;
- перевіряється наявність файлу Customfile – даний файл містить поправки щодо розміру використаних ресурсів кожним із серверів інфраструктури;

- так само як і в Dockerfile, у Vagrantfile обирається образ-шаблон, так званий vagrant box. В даному прикладі сервери піднімаються на базі vagrant box ubuntu/trusty64;

- запускаються bash-скрипти з інструкціями по встановленню docker, його оновленням, встановленням docker swarm;

- запускається load-balancer haproxy;

- IP адреси із локального файла hosts записуються у /etc/hosts в середині контейнера для того, щоб сервери розпізнавали один одного та мали змогу комунікувати;

- далі сервер-менеджер приєднується до кластеру під видом менеджера, а сервери-worker-и під видом виконавців;

- останньою дією є розгортання Prometheus stack.

Prometheus – сервіс для збору даних через SNMP – Simple Network Management Protocol – простий протокол мережевого управління. Він складається з ряду підлеглих сервісів колекторів, кожен з яких відповідає за збір інформації з того чи іншого типу сервісу. Стек також містить сервіс Grafana, який налаштовано на відображення накопиченої інформації з Прометея у вигляді графіків.

Зразок Vagrantfile наведено нижче:

```
# -*- mode: ruby -*-
```

```
# vi: set ft=ruby :
```

```
# All Vagrant configuration is done below. The "2" in Vagrant.configure  
# configures the configuration version (we support older styles for  
# backwards compatibility). Please don't change it unless you know what  
# you're doing.
```

```
auto = ENV['AUTO_START_SWARM'] || false
```

```
vagrant_dir = File.expand_path(File.dirname(__FILE__))
```

```
# Increase numworkers if you want more than 3 nodes
```

```
numworkers = 2
```

```
# VirtualBox settings
```

```

# Increase vmmemory if you want more than 512mb memory in the vm's
vmmemory = 1024

# Increase numcpu if you want more cpu's per vm
numcpu = 1

cpuexecutioncap = 100

instances = []

(1..numworkers).each do |n|
  instances.push({:name => "worker#{n}", :ip => "192.168.10.#{n+2}"})
end

manager_ip = "192.168.10.2"

File.open("./hosts", 'w') { |file|
  instances.each do |i|
    file.write("#{i[:ip]} #{i[:name]} #{i[:name]}\n")
  end
}

Vagrant.configure("2") do |config|
  config.vm.provider "virtualbox" do |v|
    v.memory = vmmemory
    v.cpus = numcpu
    v.customize ["modifyvm", :id, "--cpuexecutioncap", cpuexecutioncap]
    if File.exists?(File.join(vagrant_dir, 'Customfile')) then
      eval(IO.read(File.join(vagrant_dir, 'Customfile')), binding)
    end
  end

  config.vm.define "manager" do |i|
    i.vm.box = "ubuntu/trusty64"
    i.vm.hostname = "manager"
    i.vm.network "private_network", ip: "#{manager_ip}"
    i.vm.provision "shell", path: "./provision.sh"
    i.vm.provision "shell", path: "./haproxy.sh"
  end
end

```

```

    if File.file?("./hosts")
      i.vm.provision "file", source: "hosts", destination: "/tmp/hosts"
      i.vm.provision "shell", inline: "cat /tmp/hosts >> /etc/hosts", privileged:
true
    end
    if auto
      i.vm.provision "shell", inline: "docker swarm init --advertise-addr
#{manager_ip}"
      i.vm.provision "shell", inline: "docker swarm join-token -q worker >
/vagrant/token"
    end
    i.vm.provision "shell", path: "./prometheus.sh"
  end
  instances.each do |instance|
    config.vm.define instance[:name] do |i|
      i.vm.box = "ubuntu/trusty64"
      i.vm.hostname = instance[:name]
      i.vm.network "private_network", ip: "#{instance[:ip]}"
      i.vm.provision "shell", path: "./provision.sh"
      if File.file?("./hosts")
        i.vm.provision "file", source: "hosts", destination: "/tmp/hosts"
        i.vm.provision "shell", inline: "cat /tmp/hosts >> /etc/hosts", privileged:
true
      end
      if auto
        i.vm.provision "shell", inline: "docker swarm join --advertise-addr
#{instance[:ip]} --listen-addr #{instance[:ip]}:2377 --token `cat /vagrant/token`
#{manager_ip}:2377"
      end
    end
  end
end

```

end

end

Основний потік змінює об'єм використовуваних системних ресурсів згідно з встановленими користувачем значеннями за допомогою шаблонізатора `handlebars` (рис. Б.7). Шаблонний файл копіюється, коренева директорія змінюється на директорію, в якій лежить `Vagrantfile` та виконується команда `vagrant up`.

```
13
14 # VirtualBox settings
15 # Increase vmmemory if you want more than 512mb memory in the vm's
16 vmmemory = {{ram}}
17 # Increase numcpu if you want more cpu's per vm
18 numcpu = {{cpus}}
19 cpuexecutioncap = {{cpuPercentage}}
```

Рис. Б.7. Формат шаблону `handlebars`

Команда виконується у терміналі за допомогою модуля `vagrant`, який запускає команди у вигляді `child_process` – окремого потоку виконання. В якості аргумента у функцію надходить команда та список аргументів. В результаті у паралельному режимі команда виконується напряму в терміналі. Завантажується образ – шаблон, проходить увесь ряд налаштувань послідовно та в результаті успішного розгортання, стає доступною за вказаним `ip`, який в даному випадку є `192.168.10.2`.

`Worker` з основного потоку помічає, що система розгорнута та починає збирати дані з `Grafana` через `HTTP`-запити до `Grafana API`. Якщо необхідно, проводить авторизацію та отримує токен, який і використовує на протязі своєї дії. Запит на дані відправляється раз в 10 секунд.

Дані відображаються у вигляді графіків безпосередньо через інтерфейс `Grafana`, доступний за адресою `192.168.10.2:3000`, а також основні графіки відображені у клієнтській програмі – графік використання оперативної пам'яті, графік навантаження на процесор та графік, показуючий загальну кількість ядер, які віртуальна машина може використовувати.

У вигляді діаграми розгортання на рис. Б.8 результуючу систему з прикладу зображено таким чином:

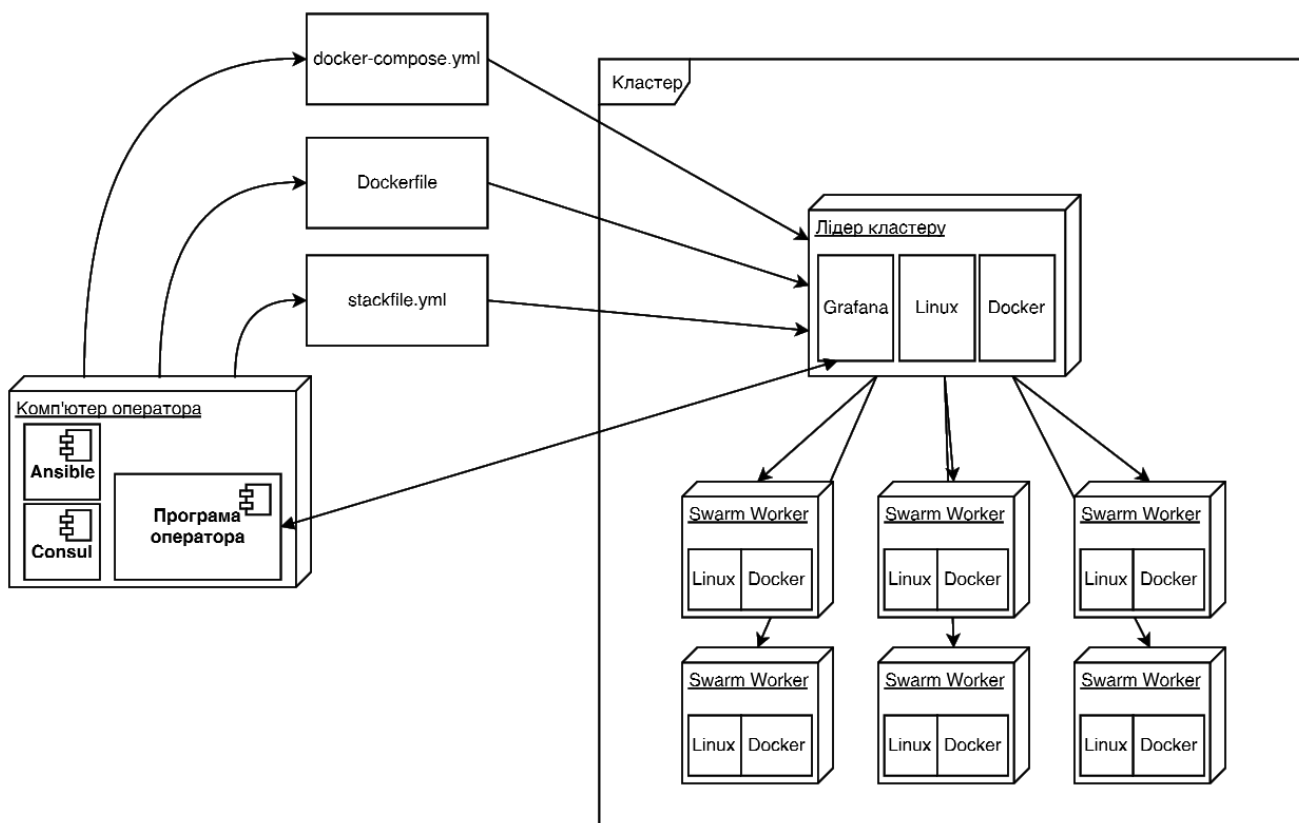


Рис. Б.8. Діаграма розгортання кластеру серверів ІТ-інфраструктури за допомогою модуля віртуалізації

Отже, серверний кластер представляє собою інфраструктуру, з серверів, які об'єднуються в кластер на основі технології Docker Swarm. Серед серверів розподіляються ролі – сервери можуть бути менеджерами, або виконавцями. Серед менеджерів обирається лідер. Краще, якщо кількість менеджерів у кластері буде не менше 3х, адже у випадку, коли лідер кластеру виходить із ладу, кластер не зможе обрати нового лідера. Вибори нового лідера відбуваються у вигляді голосування на основі пріоритетів. Безпечною мінімальною кількістю менеджерів є саме число 3, адже для успішної роботи кластеру, необхідно, щоб дві треті менеджерів було доступні. Таким чином, маючи 3 менеджери, один з них може безпечно вийти з ладу та замінений пізніше, коли у оператора буде на те час. В даному прикладі та на діаграмі зображено лише один менеджер, який також автоматично є лідером кластеру, через те, щоб зекономити системні ресурси, адже система тестується локально на комп'ютері у вигляді локального

кластеру та вірогідність випадку виходу із ладу лідера в таких умовах є дуже малою, а також, велика кількість серверів використовувала б занадто велику кількість ресурсів для нормальної демонстрації проекту.

В процесі розгортання інфраструктури, за допомогою Vagrant, комп'ютер оператора здійснює безпечне підключення через ssh протокол та встановлює на сервері сучасну версію docker. Важливим є завантаження саме найновішою версії, щоб підтримувати усі можливі версії .yaml docker stackfile. Після цього завантажується stackfile з архітектурою. Такий файл розпізнається docker сервісом та завантажується за допомогою команди [286].

`docker stack deploy -c назва-файлу.yaml`

Саме тому, на діаграмі зображено асоціативний зв'язок між stackfile та сервісами та композицію між docker та сервісами. Коли лідер кластеру готовий до роботи, він генерує спеціальний ключ – токен, який іншим серверам необхідно використати в команді підключення для того, щоб бути доданим до саме цього кластеру. Такий ключ записується у вигляді локального файлу на комп'ютері оператора та після підняття серверів працівників, завантажується до них разом з stackfile-ом.

Коли сервер працівник піднято, на ньому також оновлюється версія докеру до найсучаснішої. Далі за допомогою команди [286]:

`docker swarm join --token ключ --listen-addr ip_адреса_лідера`

він під'єднується до кластеру. При цьому, локальний сервіс докеру підключається до сервісу докер на сервері лідера кластеру та між ними створюється TCP-з'єднання.

Коли на сервері лідері кластеру розгортаються сервіси, Docker Platform дозволяє звертатись до них через будь-яку із IP-адрес серверів, які належать кластеру. Також, сервіси можуть бути завантажені на будь-який із серверів, які належать кластеру. Таким чином досягається логічне об'єднання серверів у кластер. За необхідності, docker swarm також проводить балансування навантаження на репліки сервісів, дозволяє масштабувати архітектуру за допомогою команди [286]:

`docker service scale id_сервісу=к-ть_реплік`

а через специфіку роботи `docker swarm`, новий сервер дуже легко під'єднується до кластеру – необхідно лише мати IP-адресу поточного лідера кластеру та згенерований ключ для з'єднання.

ДОДАТОК В

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Монографії, в яких опубліковані основні наукові результати дисертації:

- [1] Я. Дорогий, та В. Цуркан, “Методи представлення та обґрунтування архітектури критичної ІТ-інфраструктури”, in *Engineering sciences: development prospects in countries of Europe at the beginning of the third millennium: Collective monograph*, vol. 1, Riga: Izdevnieciba “Baltija Publishing,” 2018, pp. 197–237.
- [2] Я. Дорогий, “Моделі і методи представлення архітектурних рішень критичної ІТ-інфраструктури”, in *Modern engineering research: topical problems, challenges and modernity: Collective monograph*, Riga: Izdevnieciba “Baltija Publishing,” 2020, pp. 127–156, doi: 10.30525/978-9934-588-47-1.7.

Наукові праці, в яких опубліковані основні наукові результати дисертації:

- [3] Y. Y. Dorogyu, “Management of Critical It-Infrastructures”, *Inf. Telecommun. Sci.*, vol. 0, no. 1, pp. 10–15, Aug. 2014, doi: 10.20535/2411-2976.12014.10-15.
- [4] Я. Дорогий, “Життєвий цикл критичної ІТ-інфраструктури”, *Електроніка та зв’язок*, т. 20, №4, с. 100–105, 2015.
- [5] Я. Дорогий, “Порівняльний аналіз методів представлення та обґрунтування архітектури критичної ІТ-інфраструктури”, *Вісник НТУ «ХПІ». Серія Механіко-технологічні системи та комплекси*, т. 33, №1255, с. 42–54, 2017.
- [6] Я. Дорогий, “Мета-модель компенсаційно-декомпенсаційного підходу до проектування архітектури критичної ІТ-інфраструктури”, *Вісник університету України Інформатика, обчислювальна техніка та кібернетика*, т. 2, №17, с. 14–22, 2016.
- [7] Я. Дорогий, “Розробка підходу до проектування, моделювання та дослідження критичної ІТ-інфраструктури”, *Технологічний аудит та резерви виробництва*, т. 5, №2(37), с. 34–41, 2017.

- [8] Y. Dorogyu, “Development of a model for optimal configuration components selection for architecture of critical IT infrastructure at its designing”, *Технологічний аудит та резерви виробництва*, №6/2(38), с. 19–27, 2017.
- [9] Я. Дорогий, “Алгоритм структурної оптимізації нейронної мережі”, *Вісник НТУУ «КПІ», «Інформатика, управління та обчислювальна техніка»*, №61, с. 169–173, 2014.
- [10] Я. Дорогий, “ТС-сумісна архітектура критичної ІТ-інфраструктури”, *Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка збірник наукових праць*, т. 65, с. 123–126, 2017.
- [11] Я. Дорогий, “Розподіл ресурсів критичної ІТ-інфраструктури з використанням хмарних технологій”, *Електроніка та зв’язок*, т. 1, №90, с. 42–49, 2016.
- [12] Я. Дорогий, “Технологія пошуку сумнівних записів при створенні єдиного реєстру ідентифікації фізичних осіб України”, *Inf. Technol. Secur.*, т. 7, №2, с. 114–125, 2019.
- [13] Y. Dorogyu, O. Doroga-Ivaniuk, V. Tsurkan, and S. Telenyk, “Comparative analysis of the architecture designing platform for critical it infrastructure”, *Inf. Technol. Secur.*, vol. 5, no. 2, pp. 90–118, 2017.
- [14] Я. Дорогий, І. Бондаренко, Т. Шемседінов, та С. Стиренко, “Аналіз проблем побудови критичної ІТ-інфраструктури міністерства”, *Inf. Technol. Secur.*, т. 1, №10, с. 96–107, 2018.
- [15] Y. Dorogyu, S. Telenyk, V. Tsurkan, and D. Halushko, “Structural optimization of neural network in qualitative evaluation method of IT-infrastructure functioning”, *Inf. Telecommun. Sci.*, vol. 6, no. 2, pp. 36–43, 2015.
- [16] Я. Дорогий, О. Дорога-Іванюк, В. Цуркан, та Д. Ференс, “Застосування алгоритму структурної оптимізації нейронної мережі в задачах класифікації даних”, *Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка збірник наукових праць*, №62, с. 100–104, 2015.

- [17] О. Дорога-Іванюк, Я. Дорогий, та Д. Ференс, “Реалізація алгоритму структурної оптимізації нейронної мережі”, *Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка збірник наукових праць*, №63, с. 101–106, 2015.
- [18] G. Nowakowski, Y. Dorogyu, and O. Doroga-Ivaniuk, “Neural network structure optimization algorithm,” *J. Autom. Mob. Robot. Intell. Syst.*, vol. 12, no. 1, pp. 5–13, 2018, doi: 10.14313/JAMRIS_1-2018/1. **(Журнал індексується в Elsevier Scopus)**
- [19] Я. Дорогий, Є. Максименко, О. Крук, та В. Цуркан, “Оцінювання ризику безпеки інформації на основі спектрального підходу”, *Inf. Technol. Secur.*, т. 3, №2(5), с. 138–146, 2015.
- [20] Я. Дорогий, В. Мохор, В. Цуркан, та О. Крук, “Функціональне моделювання системи керування ризиком безпеки інформації”, *Захист інформації*, т. 18, №1, с. 74–80, 2016.
- [21] Я. Дорогий, О. Бакалинський, О. Богданов, С. Михайлов, В. Цуркан, та В. Мохор, “Використання ентропійного підходу для оцінювання ризиків безпеки інформації”, *Inf. Technol. Secur.*, т. 4, №2(7), с. 255–261, 2016.
- [22] Я. Дорогий, Д. Ференс, та О. Дорога-Іванюк, “Модель розподілу ресурсів критичної ІТ-інфраструктури з чіткими параметрами на основі генетичного алгоритму”, *Inf. Technol. Secur.*, т. 2, №11, с. 124–144, 2018.
- [23] Я. Дорогий, Д. Ференс, та О. Дорога-Іванюк, “Модель розподілу ресурсів критичної ІТ-інфраструктури з чіткими параметрами на основі методу рою часток”, *Електронне моделювання*, т. 41, №2, с. 1–15, 2019.
- [24] А. Йовенко, та Я. Дорогий, “Вибір методу інтеграції розподілених гетерогенних модулів для розробки корпоративних систем”, *Вісник університету України Інформатика, обчислювальна техніка та кібернетика*, т. 1, №18, с. 14–22, 2016.

- [25] Я. Дорогий, В. Мохор, та В. Цуркан, “Концептуальна модель описання архітектури системи управління інформаційною безпекою”, *Безпека інформації*, т. 25, №3, с. 162–166, 2019.
- [26] Я. Дорогий, та В. Цуркан, “Огляд методів верифікації параметризованих моделей”, *Збірник наукових праць Національного університету кораблебудування імені адмірала Макарова*, №1(479), с. 82–91, 2020, doi: [https://dpo.org/10.15589/znp2020.1\(479\).10](https://dpo.org/10.15589/znp2020.1(479).10).
- [27] Я. Ю. Дорогий, В. В. Мохор, В. В. Цуркан, та Ю. М. Штифурак, “Структури архітектури систем управління безпекою інформації”, *Інформатика та математичні методи в моделюванні*, т. 9, № 4, с. 209–221, 2019, doi: 10.15276/imms.v9.no4.209.
- [28] G. Nowakowski, Y. Dorogyu, and O. Doroga-Ivaniuk, “The realisation of neural network structural optimization algorithm,” in *Annals of Computer Science and Information Systems (ACSIS)*, vol. 11, 2017, pp. 1365–1371. **(Журнал індексується в Elsevier Scopus)**
- [29] Д. Ференс and Я. Дорогий, “Оптимізація розміщення віртуальних машин у хмарних інфраструктурах за допомогою гнучкого алгоритму на базі генетичного програмування”, *Інфокомунікаційні системи та технології*, т. 1, №1, 2017.
- [30] К. Левченко and Я. Дорогий, “Порівняння часу виконання базових операцій фреймворків машинного навчання”, *Інфокомунікаційні системи та технології*, т. 2, №2, с. 27–31, 2018.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

- [31] S. Telenyk, O. Rolick, M. Bukasov, Y. Dorogiy, D. Halushko, and A. Pysarenko, “Qualitative evaluation method of IT-infrastructure elements functioning”, in *Proc. 2014 IEEE International Black Sea Conference on Communications and Networking, BlackSeaCom 2014*, 2014, pp. 165–169, doi:

- [32] Y. Dorogyu and K. Valchuk, “Comparison between serialization formats”, in *Collection of Conference Papers of International Scientific-Practical Conference “Problems and Prospects of Intergration of Science and Technology,”* 2015.
- [33] Я. Дорогий, І. Козлюк, В. Цуркан, та В. Мохор, “Критична інфраструктура: вразливості, загрози, ризики,” на *II міжн. наук.-практ. конф. «Інформаційні технології та взаємодії»*, 2015.
- [34] Я. Дорогий, та О. Дорога-Іванюк, “Критерій оптимальності побудови критичної ІТ-інфраструктури”, на *2-й міжн. наук.-практ. конф. Актуальні проблеми розвитку науки і техніки, 20 грудня*, 2015.
- [35] Я. Дорогий, Є. Максименко, В. Цуркан, та В. Мохор, “Аналітика кібербезпеки: поняття, структура, задачі”, на *Межд. научн. конф. ИТБ-2015, 21 октября*, 2015.
- [36] О. Дорога-Іванюк, та Я. Дорогий, “Умови створення критичних процесів або сервісів критичної ІТ-інфраструктури”, на *V міжн. наук.-техн. конф. «Проблеми інформатизації»*, 10-11 грудня, 2015.
- [37] Y. Dorogyu, O. Doroga-Ivaniuk, B. Mart, and V. Tsurkan, “Life cycle of IT-infrastructure,” in *Proceedings of The Congress on Information Technology, Computational and Experimental Physics 2015 (CITCEP 2015)*, 2015.
- [38] Y. Dorogyu, O. Doroga-Ivaniuk, S. Telenik, and D. Halushko, “Qualitative evaluation method of IT-infrastructure functioning based on structural optimization of neural network,” in *Proc. 2015 2nd International Scientific-Practical Conference Problems of Infocommunications Science and Technology, PIC S&T 2015 - Conference Proceedings*, 2015, pp. 1–4, doi: 10.1109/INFOCOMMST.2015.7357252. (Матеріали конференції індексуються в Elsevier Scopus)

- [39] Д. Ференс, та Я. Дорогий, “Структурна оптимізація штучних нейронних мереж”, на 2-й міжн. наук.-практ. конф. *Summer Infocom 2016*, 2016.
- [40] Я. Дорогий, О. Крук, В. Цуркан, та В. Мохор, “Процесна модель системи керування ризиком безпеки інформації”, на V Міжнар. на-ук.-техн. конф. *“Захист інформації і безпека інформаційних систем,”* 2016.
- [41] Я. Дорогий, О. Крук, В. Цуркан, та В. Мохор, “Дерево вузлів функціональної моделі системи керування ризиком безпеки інформації”, на Міжнар. наук.-практ. конф. *“Безпека інформації в інформаційно-телекомунікаційних системах”*, 25-26 травня, 2016.
- [42] Я. Дорогий, О. Крук, В. Мохор, та В. Цуркан, “Модель потоків даних системи керування ризиком безпеки інформації”, на V-й міжн. наук.-практ. конф. *“Кібербезпека інформаційних технологій /// Фізико-технологічні проблеми передавання, оброблення та зберігання інформації в інфокомунікаційних системах*, 2016.
- [43] Я. Дорогий, О. Крук, В. Цуркан, та В. Мохор, “Функціональна модель системи керування ризиком безпеки інформації”, на II-й міжн. наук.-практ. конф. *“Актуальні проблеми моделювання ризиків і загроз виникнення надзвичайних ситуацій на об'єктах критичної інфраструктури,”* 2016.
- [44] Y. Y. Dorogyu, O. O. Doroha-Ivaniuk, U. Dzelendzyak, and K. Tomczyk, “Usage of structural optimization algorithm of neural nets in problems of data classification”, in *Proceedings of the 2017 IEEE 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2017*, 2017, vol. 2, pp. 983–987, doi: 10.1109/IDAACS.2017.8095234. **(Матеріали конференції індексуються в Elsevier Scopus)**
- [45] Я. Дорогий, “Проектування критичних ІТ-інфраструктур з використанням розмічених транзиційних систем”, на V заочн. научн. конф. *«Фундаментальные и прикладные исследования в современной науке»*, 31 октября, 2017.

- [46] Я. Дорогий, “Моделювання критичних ІТ-інфраструктур з використанням розмічених транзиційних систем”, на *VI Міжн. наук.-практ. конференція (I Міжнародний симпозиум «Практичне застосування нелінійних динамічних систем в інфокомунікаціях», PREDT-2017, 9-11 листопада, 2017.*
- [47] Я. Дорогий, О. Дорога-Іванюк, and В. М. В. Цуркан, “Cyber resilience: the concept of information security in cyberspace”, на *III міжнар. наук.-практ. конф. «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації», 2017.*
- [48] Y. Dorogyu, “Meta model of compensatory-decompensational approach for architecture of critical it infrastructure designing”, in *Proc. 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, TCSET 2018 - Proceedings, 2018, vol. 2018-April, pp. 223–228, doi: 10.1109/TCSET.2018.8336191. (Матеріали конференції індексуються в Elsevier Scopus)*
- [49] Я. Дорогий, В. Мохор, та В. Ц. Ю. Штифурак, “Моделі оцінювання ризику кібербезпеки”, на *VII Міжн. наук.-практ. конф. «Фізико-технологічні проблеми передавання, оброблення та зберігання інформації в інфокомунікаційних системах (PREDT-2018)», 8-10 листопада, 2018.*
- [50] А. Горносталь, та Я. Дорогий, “Мурашиний алгоритм кластеризації”, на *XI Міжнар. наук.-практ. конф. "ІНТЕРНЕТ-ОСВІТА-2018, 2018.*
- [51] Я. Дорогий, та К. Левченко, “Порівняння фреймворків машинного навчання”, на *VI-й міжн. наук.-практ. конф. з інформаційних систем та технологій Summer InfoCom 2018, 2018.*

ДОДАТОК Г
ДОВІДКИ ПРО ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОГО
ДОСЛІДЖЕННЯ

**МІНІСТЕРСТВО
ВНУТРІШНІХ СПРАВ
УКРАЇНИ**

ДЕПАРТАМЕНТ ІНФОРМАТИЗАЦІЇ

вул. Богомольця, 10, м. Київ, 01601
тел.: (044) 256-03-33, факс: (044) 253-64-04
E-mail: zmi@mvs.gov.ua,
сайт: www.mvs.gov.ua
код згідно з ЄДРПОУ 00032684

05 березня 2020 року № 7208/16

ДОВІДКА

**про впровадження результатів дисертаційної роботи
Дорогого Ярослава Юрійовича, поданої на здобуття наукового ступеня
доктора технічних наук, на тему «Методи і моделі підвищення
ефективності процесу проектування критичної ІТ-інфраструктури»
при розробленні сервісу ідентифікації ЄІС МВС**

За допомогою моделей та методів проектування критичної ІТ-інфраструктури, розроблених Дорогим Ярославом Юрійовичем в дисертаційній роботі «Методи і моделі підвищення ефективності процесу проектування критичної ІТ-інфраструктури» виконані роботи з створення сервісу ідентифікації єдиної інформаційної системи Міністерства внутрішніх справ України (далі – ЄІС МВС), а саме:

1. Розроблені здобувачем Дорогим Я.Ю. методи та моделі проектування критичної ІТ-інфраструктури, а також метод верифікації даних в інформаційних ресурсах критичної ІТ-інфраструктури використані при проектуванні та реалізації сервісу ідентифікації ЄІС МВС.

2. Використання сервісу ідентифікації ЄІС МВС дозволило Міністерству внутрішніх справ України скоротити час на виконання верифікації даних в національних електронних інформаційних реєстрах при виконанні завдань Постанови КМУ №1078 від 04 грудня 2019 року «Про реалізацію експериментального проекту верифікації даних про фізичних осіб, що обробляються в деяких національних електронних інформаційних ресурсах» на 90% та скоротити кількість записів з невизначеним статусом в ресурсах до 2%.

Директор Департаменту



Ігор БОНДАРЕНКО



ТОВ «ОЛЛІ ТРАНС»,
Свідоцтво платника ПДВ № 200146537, ЄДРПОУ 36049543
Індивідуальний податковий номер № 360495426594
Юридична адреса: 01135, м. Київ, вул. Полтавська, 10
тел./факс: (044) 390 17 75/(044) 390 17 76
www.ollie.com.ua

ДОВІДКА

**про впровадження результатів дисертаційної роботи
Дорогого Ярослава Юрійовича, поданої на здобуття наукового ступеня
доктора технічних наук, на тему «Методи і моделі підвищення
ефективності процесу проектування критичної ІТ-інфраструктури»
при проектуванні функціональної підсистеми ЄІС МВС «Система
автоматичної фіксації порушень правил дорожнього руху»**

Положення та результати дисертаційної роботи Дорогого Ярослава Юрійовича використані підприємством при реалізації функціональної підсистеми ЄІС МВС «Система автоматичної фіксації порушень правил дорожнього руху» (далі – СФАП ЄІС МВС), а саме:

1. Розробці техноробочого проекту СФАП ЄІС МВС, що дозволило отримати якісну програмно-апаратну платформу функціональної підсистеми.
2. Запропоновані методи проектування інформаційних систем та їх представлення, методи визначення оптимальних специфікацій компонент для створення функціональної підсистеми дозволило підприємству зменшити час на проектування та реалізацію підсистеми на 10%.

З повагою,

Генеральний директор

 **Ігнатовський О.В.**

Вик.: Онопрієнко Ю.

тел.: +380443901775,

mail@ollie.com.ua

**МІНІСТЕРСТВО ВНУТРІШНІХ
СПРАВ УКРАЇНИ
ДЕРЖАВНЕ ПІДПРИЄМСТВО
“ІНФОТЕХ”**

вул. Дегтярівська, 15 Б, м. Київ, 04050
e-mail: info@infotech.gov.ua
www.infotech.gov.ua
тел.: (044) 363-43-34
Код за ЄДРПОУ 34239034



ІНФОТЕХ
Мобільність Інновації Аналітика

**THE MINISTRY OF INTERNAL
AFFAIRS OF UKRAINE
STATE ENTERPRISE
“INFOTECH”**

Dehtiarivs'ka str., 15 B, Kyiv, 04050
e-mail: info@infotech.gov.ua
www.infotech.gov.ua
phone: (044) 363-43-34
EDRPOU 34239034

Вих. № 671/04 від «22» червня 2020 р.

ДОВІДКА
про впровадження при проектуванні ЄІС МВС
результатів дисертаційної роботи Дорогого Ярослава Юрійовича, поданої на здобуття
наукового ступеня доктора технічних наук, на тему: «Методи і моделі підвищення
ефективності процесу проектування критичної ІТ-інфраструктури»

Цим підтверджується, що наведені та реалізовані в дисертаційній роботі Дорогого Я.Ю. ідеї, моделі та методи проектування критичної ІТ-інфраструктури використані Державним підприємством «ІНФОТЕХ» – адміністратором Єдиної інформаційної системи Міністерства внутрішніх справ України (далі – ЄІС МВС) при реалізації таких проектів, що пов'язані з ЄІС МВС:

1. Розробка техноробочого проекту центральної підсистеми ЄІС МВС, що дозволило отримати якісну і економічно вигідну програмно-апаратну платформу, залучити до її розвитку управлінський і технічний персонал.
2. Проектування центру обробки даних МВС, що дозволило скоротити час на його проектування на 20%.

Директор

Андрій ТИМОШЕНКО

Максим Колісниченко 050 151 01 08

ДОВІДКА

про впровадження результатів дисертаційної роботи
Дорогого Ярослава Юрійовича, поданої на здобуття наукового ступеня
доктора технічних наук, на тему «Методи і моделі підвищення
ефективності процесу проектування критичної ІТ-інфраструктури»
при проектуванні функціональної підсистеми ЄІС МВС «Інформаційно-
аналітичної системи оперативного інформування про події»

Цим підтверджується, що основні положення та результати дисертаційної роботи Дорогого Ярослава Юрійовича використані підприємством при реалізації функціональної підсистеми єдиної інформаційної системи Міністерства внутрішніх справ «Інформаційно-аналітична система оперативного інформування про події».

Застосування пропонованих автором моделей та методів проектування та обґрунтування архітектури підсистеми, моделей оцінки та вибору оптимальної конфігурації компонент для її реалізації дозволило підприємству скоротити час на проектування та створення підсистеми на 18%.

Директор



Мар'їн Ю.Я.