

Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“___” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи гіперконвергентного
хмарного рішення на основі технології HaaS”

Виконав здобувач вищої освіти
IV курсу, групи KI-22-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Масний В.С.
«___» _____ 2026 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Марченко К.М.
«___» _____ 2026 р.
Рецензент _____

Центральноукраїнський національний технічний університет
Факультет Механіко-технологічний
Кафедра Кібербезпеки та програмного забезпечення
Освітній ступінь бакалавр
Галузь знань 12 "Інформаційні технології"
Спеціальність 123 "Комп'ютерна інженерія"
Освітньо-професійна (освітньо-наукова) програма "Комп'ютерна інженерія"

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 27 » лютого 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Масному Владиславу Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи гіперконвергентного хмарного рішення на основі технології HaaS*

2. Керівник роботи *Марченко Костянтин Миколайович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 136-02 від 27.02.2026 року

3. Строк подання студентом роботи до захисту 23.05.2026 р.

4. Мета та завдання випускної кваліфікаційної роботи: *Метою роботи є розробка програмного забезпечення системи гіперконвергентного хмарного рішення на основі технології HaaS*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2. Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи 1 аркуш

Функціональна схема системи 1 аркуш

Діаграма процесів 1 аркуш

Блок-схема алгоритму роботи додатку 2 аркуша

7. Дата видачі завдання « 27 » лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	17.04.2026 р.	
8.	Попередній захист роботи	23.05.2026 р.	

Дата видачі завдання
« 27 » лютого 2026 р.

Підпис керівника

Марченко К.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 27 » лютого 2026 р.

Підпис здобувача

Масний В.С.
(прізвище та ініціали)

АНОТАЦІЯ

Масний В.С. Програмне забезпечення системи гіперконвергентного хмарного рішення на основі технології NaaS. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи гіперконвергентного хмарного рішення на основі технології NaaS.

Метою розробки є програмне забезпечення системи гіперконвергентного хмарного рішення на основі технології NaaS.

Результат роботи – програмна реалізація системи гіперконвергентного хмарного рішення на основі технології NaaS.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, NaaS

ABSTRACT

Masnyi V.S. Software for a hyperconverged cloud solution system based on HaaS technology. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a hyperconverged cloud solution system based on HaaS technology.

The purpose of the development is software for a hyperconverged cloud solution system based on HaaS technology.

The result of the work is a software implementation of a hyperconverged cloud solution system based on HaaS technology.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the Python environment.

Keywords: computer engineering, HaaS

3MIST

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ	2
ВСТУП.....	3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ	5
1.1 Призначення системи.....	5
1.2 Область застосування.....	6
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	8
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.....	8
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування.....	13
2.3 Розгорнута постановка завдання	15
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ	17
3.1 Опис функціонування системи	17
3.2 Розробка структурної схеми.....	22
3.3 Розробка функціональної схеми	27
3.4 Розробка діаграми процесів.....	31
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ.....	33
4.1 Розробка блок-схем та опис алгоритмів функціонування системи.....	33
4.2 Захист розробленого програмного забезпечення.....	52
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ	55
6 ОСНОВНІ ВИСНОВКИ.....	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	63

					ВКРБ-123.26.0049.00.00.ПЗ			
Вим	Арк.	№ докум.	Підп.	Дата				
Розроб.	Масний В.С.				Програмне забезпечення системи гіперконвергентного хмарного рішення на основі технології HaaS	Літ.	Аркуш	Аркушів
Перев.	Марченко К.М.					Б	1	70
						ЦНТУ КІ-22-2		
Н.контр.	Коваленко А.С.							
Затв.	Смірнов О.А.							

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	—	база даних
ЛОМ	—	локальна обчислювальна мережа
ASP		Active Server Pages – активні серверні сторінки
DHCP	—	Dynamic Host Configuration Protocol – протокол динамічної конфігурації вузла
HTTP	—	HyperText Transfer Protocol – протокол передачі гіпер тексту
IMAP	—	Internet Message Access Protocol – протокол доступу до електронної пошти Інтернету
ICMP	—	Internet Control Message Protocol – міжмережний протокол керуючих повідомлень
MMC	—	Microsoft Management Console
POP3	—	Post Office Protocol Version 3 – протокол поштового відділення, версія 3
SQL	—	Structured Query Language – мова структурованих запитів
SMTP	—	Simple Mail Transfer Protocol – простий протокол передачі пошти
SNMP	—	Simple Network Management Protocol – простий протокол керування мережею
Syslog	—	стандарт відправки повідомлень про зміни які відбуваються в мережі
UDP	—	User Datagram Protocol – протокол користувальницьких дейтаграм

ВСТУП

Актуальність теми. Центри обробки даних існують не самі по собі, а для рішення реальних завдань. Перелік проблем, з якими доводиться зіштовхуватися ІТ-професіоналам, не обмежується межами ЦОД.

Сучасна цифрова інфраструктура повинна гнучко адаптуватися до вимог бізнесу, що змінюються, дозволяючи швидко нарощувати ресурси. Саме вона уможлиблює реалізацію передових моделей бізнесу. Підтримка традиційної ІТ-інфраструктури вимагає значних ресурсів. Ідея, що у числі іншого просуває й компанія Cisco, полягає в тому, щоб створити єдину легко керовану платформу, де були б об'єднані різні типи устаткування. Її втіленням є конвергентні й гіперконвергентні системи, а також типові валідовані дизайни на їхній основі.

На зміну традиційній інфраструктурі систем приходять конвергентна й гіперконвергентна. Остання за продуктивністю не уступає класичним рішенням, тому на її основі можна реалізувати всі типові сценарії: VDI, прикладні бізнес-системи, масштабовані сховища будь-якого обсягу й, природно, всі загальносистемні функції, такі як поштова система, портали й т.д. – практично 97–98% всіх ІС. При цьому гіперконвергентна архітектура зберігання й віртуалізації здатна забезпечити ефективне масштабування ІТ-інфраструктури.

У результаті застосування гіперконвергентних рішень не тільки спрощується розгортання обчислювальної інфраструктури, але й міняються принципи її планування й фінансового моделювання.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи гіперконвергентного хмарного рішення на основі технології HaaS.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем гіперконвергентного хмарного рішення на основі технології HaaS.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		3

– Дослідження системи гіперконвергентного хмарного рішення на основі технології NaaS.

– Програмна реалізація системи гіперконвергентного хмарного рішення на основі технології NaaS.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі гіперконвергентного хмарного рішення на основі технології NaaS.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи гіперконвергентного хмарного рішення на основі технології NaaS, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

К6ПЗ_2026

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		4

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Безпосередня мета трансформації ІТ-інфраструктури – більше ефективна підтримка застосунків. Якщо раніше застосунки розміщалися в ЦОД, що належать компаніям, то тепер вони розгортаються в публічній і приватній хмарах і надаються як сервіси SaaS, при цьому жоден варіант не виключає іншого, а найчастіше вони використовуються в різних комбінаціях. Відповідно, необхідно забезпечити як взаємодію всіх компонентів застосунків незалежно від їхнього місцезнаходження, так і безперебійний доступ до них кінцевих користувачів.

Ринок сервісів PaaS активно розвивається: на нього доводиться близько 20% загального обсягу публічних хмар, а за прогнозами західних аналітиків, ріст цього сегмента буде мати підривний характер. Використання PaaS дозволяє скоротити час і спростити розробку застосунків і тим самим прискорити вивід готових продуктів на ринок. Всі найбільші хмарні мегапровайдери, зокрема Amazon, Google і Microsoft, прагнуть дистанціюватися друг від друга за рахунок пропозиції додаткових сервісів PaaS для розроблювачів.

Хмарні мегаЦОД, здавалося б, мають незаперечні переваги. До того ж інвестиції в корпоративні ЦОД в усьому світі знижуються, а в хмарні і комерційні – ростуть. Однак уже зараз можна спостерігати ефект бумеранга, коли при досягненні певних параметрів підтримка власного локального ЦОД виявляється більше ефективною, ніж використання хмарного ЦОД. Це стосується насамперед великих підприємств, так що сегмент великих корпоративних ЦОД продовжує еволюціонувати.

Запити бізнесу приводять до розгортання багатоцільових центрів обробки даних, ускладненню й ущільненню корпоративних ЦОД. Разом з тим затребувані такі рішення, як колокейшн, хмарні/гіпермасштабовані (Cloud/Hyperscale) і

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		5

прикордонні (Edge) ЦОД, причому іноді одночасно. У результаті виникає попит на більше складні в керуванні ІТ-системи й відповідну інфраструктуру. Який тип ЦОД виявиться найбільш підходящим, залежить від безлічі умов, зокрема від вимог до часу затримки, рівню безпеки й т.д.

1.2 Область застосування

Hardware as a Service (HaaS) – альтернатива придбанню апаратного устаткування. При цьому устаткування визначається під потреби кожного конкретного замовника. Лізингодавачем виступає компанія.

Цільова аудиторія:

- Компанії, що не мають можливість виводити гроші на покупку устаткування з оборту.
- Держкомпанії, що мають обмеження на закупівлю імпортного ПЗ.
- Стартапи або дочірні компанії, що не мають можливість одержати лізинг.

Основні етапи проекту:

- Вибір устаткування й підписання договору.
- Поставка й установка устаткування. Налаштування устаткування перед запуском.
- Щомісячні платежі за оренду устаткування.

Характеристики:

- Кардинальне зниження капітальних витрат.
- Можливість розміщення устаткування як у ЦОДі компанії, так і в ЦОДі Замовника.
- Єдина точка входу при виникненні будь-якої проблеми: обслуговування й сервісна підтримка технічного й програмного складового устаткування службою технічної підтримки.
- Оперативна заміна устаткування (у рамках контрактів з вендором).

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		6

– Можливість викупу устаткування по залишковій вартості по закінченні договору оренди.

– Використання імпортного устаткування як послуги від української компанії.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи гіперконвергентного хмарного рішення на основі технології HaaS, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

К6ПЗ_2026

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		7

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Ринок гіперконвергентних (HCI) рішень продовжує рости. На ньому рідко з'являються нові гравці, тому високим попитом користуються ті продукти, які пропонують авторитетні вендори: Dell EMC, HPE, Cisco, Nutanix, VMware і Red Hat.

Як розвивається світовий ринок гіперконвергентних рішень

Тут відбувається приблизно те ж, що й у продажах серверів: ростуть апетити покупців, а великі гравці пропонують нові рішення, які краще відповідають потребам бізнесу. Тільки за перший квартал 2020-го продажу в HCI-сегменті виросли майже на 80 % у порівнянні з попереднім роком. Клієнти звертаються до гіперконвергентних рішень по різних причинах. Одні в такий спосіб зменшують складність IT-інфраструктури (а це «хвора мозоль» багатьох компаній!), інші використовують їх для розміщення критично важливих застосунків. Всі частіше сфера впливу HCI поширюється на суміжні області – гібридні й мультихмарні середовища. Давайте подивимося, які гіперконвергентні рішення виявилися кращими торік.

HPE SimpliVity

Компанія HPE вважається провідним постачальником гіперконвергентних рішень – її бізнес торік переборов оцінку в 1 млрд доларів США. HPE SimpliVity поєднує всі складові IT-інфраструктури на підприємстві, ПЗ сторонніх розроблювачів (наприклад, для віртуалізації) і засобу керування даними. На виході виходить інтегроване флеш-рішення, що дешевше, ніж традиційні інфраструктури. Сильні сторони SimpliVity – інструменти для створення бекапів і

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		8

відновлення даних, гнучке керування віртуальними машинами, оптимізація LAN-мережі, простої масштабування й висока ефективність роботи. Торік HPE купила Plexxi – розроблювача ПЗ для оптимізації мереж і підвищення продуктивності сервісів і застосунків. Це дозволило вендору впровадити в SimpliVity технологію програмно-конфігуруємих мереж. Також за останні кілька місяців HPE додала підтримку нових гіпервізорів і анонсувала моделі, які працюють із серверами й мережними рішеннями нового покоління.

Cisco HyperFlex

Гіперконвергентну систему HyperFlex від Cisco серйозно доповнили в 2019-м: виробник додав роботу з контейнерами в хмарі й підтримку популярного гіпервізора Hyper-V від Microsoft. Cisco пропонує це рішення у вигляді окремого пристрою, що інтегрується із серверами Cisco UCS, ПЗ Intersight для адміністрування в хмарі, гіпервізорами інших розроблювачів і платформою NX Data Platform. Зміни торкнулися й роботи з відео: з'явилася підтримка декількох сертифікованих GPU-продуктів. Клієнти компанії можуть управляти одночасно декількома кластерами HyperFlex, розташованими в різних ЦОД – це й ефективніше, і дешевше, ніж традиційні підходи. Ще одна гарна новина про Cisco стосується її співробітництва з Veeam – рішення останньої будуть інтегровані з HyperFlex.

Dell EMC VxRail

Компанія Dell EMC теж регулярно обновляє свої гіперконвергентні рішення. Її платформа VxRail – одна із самих продаваних у світі за підсумками минулого року. Цей HCI-продукт базується на серверах популярної лінійки PowerEdge і ПЗ vSAN від VMware. Від конкурентів його відрізняє комплексний підхід до роботи з віртуалізацією. VxRail підтримує гнучке масштабування в одному вузлі, оплату в міру нарощування обсягів сховища і є найшвидшим і простим інструментом для розширення середовища VMware.

Лінійка серверів PowerEdge, доступних для VxRail, регулярно поповнюється новими моделями. Також вендор прискорив цикл випуску свого

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		9

HCI-рішення й включив у нього актуальну версію vSAN. З'явилася можливість автоматизувати конфігурування мереж завдяки Networking SmartFabric, додана підтримка VMware Cloud Foundation.

Пристрою VxRail на серверних платформах Dell EMC PowerEdge забезпечують готовність вашої інфраструктури до майбутнього й глибоку інтеграцію з екосистемою VMware завдяки використанню технологій нового покоління. Розширені можливості автоматизації й інтеграції з гібридною хмарою VMware спрощують розгортання захищеної хмарної інфраструктури VxRail.

– Консолідація обчислювальних ресурсів, ресурсів зберігання й засобів віртуалізації, а також комплексне автоматизоване керування їхнім життєвим циклом. Єдина точка підтримки для програмного забезпечення й устаткування.

– VMware Cloud Foundation на базі VxRail служить платформою для Dell Technologies Cloud, пропонуючи інтеграцію повного стека рішень і автоматизоване розгортання доменів робочого навантаження для Kubernetes і Horizon VDI.

– Автоматизація налаштування мережі й керування наступними операціями за допомогою функції SmartFabric Services для VxRail, що дозволяє значно прискорити розгортання й спростити роботу.

– VxRail Analytical Consulting Engine (ACE) хмарна платформа аналітики на базі машинного навчання, що надає кластерам VxRail функції моніторингу, створення звітів і прогнозування споживання ресурсів.

– Гнучкі варіанти фінансування, доступні в рамках моделі Flex on Demand.

VxRail забезпечує швидке й зручне спрощення й розширення середовища VMware за рахунок простого переходу до гібридної хмари. Система VxRail створена на базі ведучих у галузі спеціалізованих серверів Dell EMC PowerEdge, розроблених для підтримки самих ресурсномістких робочих навантажень. Доступні конфігурації VxRail класу All-Flash, що забезпечують поліпшену продуктивність із можливістю налаштування устаткування перед замовленням

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		10

(включаючи накопичувачі кеш-пам'яті NVMe і мережні підключення 25 Gb для будь-яких сценаріїв використання гіперконвергентної інфраструктури). Перед поставкою ваше рішення VxRail попередньо інтегрується, налаштовується й тестується фахівцями Dell EMC відповідно до обраного варіанта розгортання (пристрій або інтеграція в стійці, з мережними компонентами або без них). Це дозволяє вам вибрати оптимальну систему VxRail для задоволення ваших потреб.

Основні характеристики:

- Консолідація ресурсів обчислень, зберігання, віртуалізації й керування.
- Автоматизація налаштування мережі за допомогою функції SmartFabric Services, що дозволяє значно спростити й прискорити розгортання.
- Неперевершена продуктивність і ефективність завдяки VMware vSAN і гіпервізора vSphere на рівні ядра.
- Надає сервіси керування даними, забезпечує відказостійкість і якість обслуговування корпоративного рівня.
- Єдина точка підтримки для програмного й апаратного забезпечення.

Nutanix Enterprise Cloud OS

Enterprise Cloud можна назвати найвідомішою й продаваною позицією в портфелі Nutanix. Компанія однієї з перші початки пропонувати корпоративним клієнтам гіперконвергентні рішення. Вони випускаються у вигляді окремих пристроїв з інтегрованим ПЗ, як хмарний сервіс управляють гіпервізорами, інструментами безпеки, ресурсами зберігання й ін. В 2019-м з'явилися додаткові програмні можливості й підтримка устаткування для організації периферійних обчислень у територіально-розподілених структурах. Самі значимі доробки – Calm для автоматизації ПЗ, Ersch для керування продуктивністю й Flow для організації SDN-мереж.

Red Hat Hyper-Converged Infrastructure for Cloud

Хмарна HCI-система від Red Hat поєднує OpenStack і Ceph Storage в одне гнучко масштабоване рішення для високих робочих навантажень. Виробник використовує гіпервізор KVM власної розробки й розподілену файлову систему

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		11

Gluster для віртуалізації ресурсів зберігання. Відкрита платформа дозволяє домогтися більше високої гнучкості при переносі застосунків між дата-центрами й периферією. Гіперконвергентна система забезпечує плавну взаємодію із замовниками й підвищеною функціональністю на границі мережі. Наприкінці минулого року Red Hat – ведучий постачальник відкритого ПЗ на Linux-системах – купила компанію NooBaaS (Ізраїль). Ця угода допоможе американському вендору доповнити своє гіперконвергентне рішення новими технологіями керування в хмарних сховищах.

VMware vSAN 6.7

Сховище vSAN – основне рішення для побудови гіперконвергентної інфраструктури від VMware. В 2019-м цей виробник випустив для нього важливе відновлення. Клієнти одержали дороблений інтерфейс на базі HTML 5, інструменти для моніторингу продуктивності і ємності зберігання, більше тісну інтеграцію з vCenter. VMware vSAN 6.7 тепер справляється з більше складними робочими навантаженнями, у тому числі з обслуговуванням хмарного ПЗ. У гіпервізорі VMware досить активувати опцію «Гіперконвергентна інфраструктура», щоб одержати потужну платформу для VDI, територіально-розподілених структур і критично важливих застосунків. Також vSAN можна замовити як послугу на платформі Amazon Web Services.

Бізнес хоче одержувати послуги з керування дата-центрами максимально оперативно й без ускладнення IT-інфраструктури. Йому потрібні недорогі, але надійні рішення, які можна швидко розгорнути й у вилученому офісі, і в приватній хмарі. Лідери ринку успішно справляються зі зрослим інтересом до гіперконвергентності й пропонують ще більш гнучкі, добре укомплектовані й найважливіше – більше ощадливі рішення, ніж використовуються в традиційних інфраструктурах.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		12

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Як мова програмування обрана Python. Python – високорівнева мова програмування загального призначення з акцентом на продуктивність розроблювача й читаність коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій.

Python підтримує кілька парадигм програмування, у тому числі структурне, об'єктно-орієнтоване, функціональне, імперативне й аспектно-орієнтоване. Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточні обчислень і зручні високорівневі структури даних. Код у Python організовується у функції й класи, які можуть поєднуватися в модулі (які у свою чергу можуть бути об'єднані в пакети).

Еталонною реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ. Він поширюється вільно під дуже ліберальною ліцензією, що дозволяє використовувати його без обмежень у будь-яких застосунках, включаючи пропрієтарні. Є реалізації інтерпретаторів для JVM (з можливістю компіляції), MSIL (з можливістю компіляції), LLVM і інших. Проект PyPy пропонує реалізацію Python на самому Python, що зменшує витрати на зміни мови й постановку експериментів над новими можливостями.

Python – мова програмування, що активно розвивається, нові версії (з додаванням/зміною мовних властивостей) виходять приблизно раз у два з половиною року. Внаслідок цього й деяких інших причин на Python відсутні ANSI, ISO або інші офіційні стандарти, їхня роль виконує CPython.

Python портований і працює майже на всіх відомих платформах – від КПК до мейнфреймів. Існують порти під Microsoft Windows, практично всі варіанти UNIX (включаючи FreeBSD і Linux), Plan 9, Mac OS і Mac OS X, iPhone OS 2.0 і вище, Palm OS, OS/2, Amiga, AS/400 і навіть OS/390, Symbian і Android.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		13

При цьому, на відміну від багатьох портуємих систем, для всіх основних платформ Python має підтримку характерних для даної платформи технологій (наприклад, Microsoft COM/DCOM). Більше того, існує спеціальна версія Python для віртуальної машини Java – Jython, що дозволяє інтерпретаторові виконуватися на будь-якій системі, що підтримує Java, при цьому класи Java можуть безпосередньо використовуватися з Python й навіть бути написаними на Python. Також кілька проектів забезпечують інтеграцію із платформою Microsoft .NET, основні з яких – IronPython і Python.Net.

Python підтримує динамічну типізацію, тобто тип змінної визначається тільки під час виконання. Тому замість «присвоювання значення змінної» краще говорити про «зв'язування значення з деяким ім'ям». У Python є убудовані типи: бульові, рядки, Unicode-рядки, цілі числа довільної точності, числа із плаваючою комою, комплексні числа й деякі інші. З колекцій Python підтримує кортежі (*tuples*), списки, словники (асоціативні масиви) і, починаючи з версії 2.4, безлічі. Всі значення в Python є об'єктами, у тому числі функції, методи, модулі, класи.

Додати новий тип можна або написавши клас (*class*), або визначивши новий тип у модулі розширення (наприклад, написаному мовою C). Система класів підтримує спадкування (одиначне й множинне) і метапрограмування. Можливе спадкування від більшості убудованих типів і типів розширень.

Всі об'єкти діляться на посилальні й атомарні. До атомарного ставляться *int*, *long*, *complex* і деякі інші. При присвоюванні атомарних об'єктів копіюється їхнє значення, у той час як для посилальних копіюється тільки покажчик на об'єкт, таким чином, обидві змінні після присвоювання використовують те саме значення. Посилальні об'єкти бувають змінювані й незмінні. Наприклад, рядки й кортежі є незмінними, а списки, словники й багато інших об'єктів – змінюваними. Кортеж у Python є, по суті, незмінним списком. У багатьох випадках кортежі працюють швидше списків, тому якщо ви не плануєте змінювати послідовність, то краще використовувати саме їх.

Мова має чіткий і послідовний синтаксис, продуману модульність й масштабованість, завдяки чому вихідний код написаних на Python програм легко читаємий.

Python – стабільна й розповсюджена мова. Він використовується в багатьох проектах і в різних якостях: як основна мова програмування або для створення розширень і інтеграції застосунків. На Python реалізоване велика кількість проектів, також він активно використовується для створення прототипів майбутніх програм. Python використовується в багатьох великих компаніях.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускню кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи гіперконвергентного хмарного рішення на основі технології HaaS.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		15

забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформулювати висновки про виконаний обсяг робіт та одержані результати.

К6ПЗ-2026

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		16

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

У міру розвитку ринку хмарних послуг кількість аббревіатур, що кінчаються на -aaS збільшується, і різні види рішень зливаються в одне. Так, відносно новим сервісом є послуга HaaS – as-a-Service, у рамках якої апаратна серверна інфраструктура надається із хмари.

Як працює HaaS

HaaS коріннями йде в SaaS – as-a-Service, програмне рішення, до якого клієнт підключається через браузер або мобільний/десктопний застосунок. HaaS як би пішов далі – постачальник рішення надає готові пакети, куди включене програмне й апаратне забезпечення, впровадження й подальша підтримка. Фактично, це найбільш широкий варіант IT-аутсорсингу, у рамках якого оптимізована інфраструктура замовника розвертається або на його території, або в дата-центрі, або в хмарі.

Сервіс надається по підписці – тобто користувач платить певну суму щомісяця. При цьому, на відміну від лізингу, замовникові не потрібно викуповувати устаткування – воно залишається в активі компанії-постачальника рішення. Єдиною точкою входу для замовника при виникненні будь-якої проблеми є постачальник HaaS послуги, сервіс виявляється відповідно до підписаного SLA (Service Level Agreement – гарантуємі параметри якості).

Вибір HaaS, у першу чергу обумовлений зниженням витрат і економією IT-бюджетів – постачальник рішення оптимізує парк техніки, кількість пристроїв знижується, а витрати стають передбачуваними. Для українського замовника ключовим моментом є відсутність витрат на закупівлю устаткування – перехід з капітальних витрат на операційні. Економія досягається й за рахунок регулярного профілактичного обслуговування, проактивного моніторингу й своєчасної заміни

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		17

необхідних вузлів – замовникові не доводиться витрачатися на амортизацію, переживати простої через виходи техніки з ладу, витрачати час- і ресурси ІТ-персоналу на неперіоритетні завдання.

Ризики переходу на НaaS зв'язані в основному з неправильним вибором постачальника: компанія може зштовхнутися з низьким якістю послуг. На жаль, у більшості випадків, по даним Ovum, найбільш важливим критерієм вибору вендора дотепер є ціна – для майже 60% опитаних дослідниками компаній. У той час як практика показує, що якість і тривалість контракту на НaaS послуги прямо залежить від досвіду й експертизи постачальника рішення. Компанії з розгалуженою мережею філій і регіональних представництв повинні працювати з постачальником, що має більшу систему регіональних складів і сервісної підтримки.

Вигода бізнес-моделі для постачальника рішення полягає в тому, що клієнти підписують довгостроковий контракт, а не оплачують разово покупку устаткування. Наприклад, Vivint – виробник пристроїв для розумного будинку, пропонує за щомісячну плату від 60 доларів користуватися датчиками безпеки, інтелектуальними замками, термостатами й іншим устаткуванням – разом з установкою, обслуговуванням і цілодобовою підтримкою клієнтів. Більша частина пристроїв виробляється самою Vivint. Сьогодні в компанії більше 1 млн клієнтів і дохід – вище 650 млн доларів США. Цім шляхом пішла й компанія Hitachi: вона підписала контракт «поїзда як сервіс» з компанією Virgin у Великобританії, у рамках якого Hitachi зберігає право власності на 65 составів і одержує гроші за обслуговування й гарантує параметри доступності.

Очевидний недолік моделі НaaS для постачальника – це обсяг первісного капіталу для первісної покупки клієнтського устаткування. І, звичайно, ризики несплати з боку замовника.

Незважаючи на популярність ІТ-аутсорсингу й хмарних рішень за рубежом, в Україні подібні сервіси не показують стрімке зростання. Передача устаткування й тим більше інфраструктури в такому обсязі сприймається

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		18

українськими замовниками неоднозначно – це страхи, пов'язані з інформаційною безпекою, втрата контролю над закупівлями, зменшення ваги ІТ-менеджменту в компанії.

Проте, аутсорсингові моделі поступово приживаються в українських замовників: у зв'язку із зі зниженням ІТ-бюджетів і відмовою від тривалих проектів на користь короткострокових зі швидким поверненням інвестицій, ІТ-аутсорсинг і хмарні сервіси під час кризи стали одними з найбільш зростаючих сегментів.

Як зв'язати хмарну функціональність із гіперконвергентною інфраструктурою

Останнім часом у корпоративному сегменті відзначається зростаючий попит на гіперконвергентну інфраструктуру (hyper-converged infrastructure, HCI) – горизонтально-горизонтально-масштабовані системи (scale-out), які поєднують сервери й сховища в єдині вузли, а також на хмарні сервіси, що сполучають технології гібридних і мультихмарних середовищ. Реагуючи на інтерес клієнтів до цих технологій, хмарні постачальники почали комбінувати HCI-продукти із хмарною функціональністю. Перевага таких продуктів полягає в тому, що вони дозволяють здійснювати окремі операції як локально, так і в публічній хмарі.

Крім цього гіперконвергентні продукти можуть застосовуватися як платформа для розгортання приватних хмарних служб. Опції зберігання в них можуть бути реалізовані або у вигляді контролера системи зберігання віртуальної машини (це характерно для продуктів Nutanix), або ж вони убудовані безпосередньо в базовий гіпервізор, що не вимагає розгортання додаткових сервісів і службових ВМ (наприклад, розподілені програмні СЗД VMware vSAN або Scale Computing Scribe). HCI управляються засобами API, центральним об'єктом керування яких є віртуальний екземпляр. Гіперконвергентним платформам не потрібне ручне керування логічними розділами або томами – вони роблять це в автоматичному режимі. Їх також легко масштабувати: сервери

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		19

або вузли додаються в кластер програмними засобами, при необхідності вони дозволяють проводити балансування ресурсів.

Модель HSI вирішує багато проблем ефективного розгортання й керування фізичною інфраструктурою, що робить її ідеальним супутником хмарних служб. Щоб зрозуміти, які HSI-рішення для розгортання приватних хмар шукати на ринку, потрібно знати, що необхідно для забезпечення їхньої роботи.

Мінімальні вимоги для хмарно-сумісного HSI-рішення

– **Автоматизація.** Вона передбачає автоматичне налаштування платформних ресурсів, що здобуваються в хмарі, (наприклад, VM) і виведення їх з експлуатації, коли вони більше не потрібні. Автоматичне підстроювання компонентів платформи для створення необхідних об'єктів необхідні на рівні компонентів (наприклад, сховища) і повинна забезпечувати роботу потоку завдань у процесі й пов'язаних з ним підпроцесах.

– **Магазин застосунків.** Для конфігурації платформи потрібен магазин з набором застосунків і сервісів. Якщо на підприємстві він представлений у вигляді автономного каталогу, то в хмарі це скоріше набір опцій. Приміром, замість установки VM у зборі цю функцію прийде передоручити операційній системі, але для того, щоб вона змогла неї зібрати, для VM попередньо прийдеться задати обсяг пам'яті й сховища й тільки потім її можна буде розгорнути у вигляді застосунку, приміром, MySQL.

– **Мультиарендність.** Якщо в публічній хмарі поняття мультиарендності (multitenant) прирівнюється до набору аккаунтів, виділених на доступ до одного екземпляра застосунку, то в частному воно позначає поділ окремих напрямків бізнесу, кожному з яких привласнюються свої рівні доступу, продуктивності й політики безпеки.

– **Аналітика.** Аналітичні інструменти застосовуються для моніторингу гібридної хмари, оптимізації розгортання застосунків (вибору оптимального за вартістю хостинга) і виявлення невикористовуваних або надлишкових ресурсів.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		20

Як додаткову вимогу до HCl-платформи можна розглянути інтеграцію з одним з публічних хмар – це забезпечує мобільність даних, дозволяючи маневрувати додатками або робочими навантаженнями між хмарою й локальним середовищем.

Гібридна хмара й мультихмара

Підприємствам, особливо великим, уже давно доводиться звертатися до різних хмарних сервісів, що викликано або вимогами до інфраструктури, або виробничою необхідністю. Очевидно, що в майбутньому це приведе до інтеграції хмарних служб різних типів (приватних, гібридних і мультихмарних), оскільки внутрішні процеси підприємств поступово переносяться в хмару. Більше того, існує чимало компаній, які обзавелися хмарними стратегіями. Деякі вибирають публічну хмару як еталонна модель для переносу базових служб, деякі використовують її як засіб для розробки.

Основна складність для підприємств полягає в тому, щоб сполучити роботу локальних служб і служб, які працюють у публічній хмарі, тобто створити гібридна хмара. Якість його роботи визначається досягнутим рівнем інтегрованості між локаціями. Також приймаються в увагу вимоги до проектування застосунків, припустима пропускна здатність і переносимість робочих навантажень застосунків.

Гібридні хмари служать своєрідним буфером обчислювальної потужності в моменти більших навантажень. Якщо компанія виходить за межі потужності власної хмари, то підключається ще одне – зовнішнє. Як правило, у публічній хмарі перебувають клієнтські застосунки, які недоцільно видавати у відкритий простір прямо зі своєї хмари; критичні застосунки, дані й навантаження звичайно розміщуються в приватній хмарі.

Альтернативою гібридному є мультихмара – вона дозволяє оперувати робочими навантаженнями як локально, так і в публічних хмарних службах декількох провайдерів. Значимість цього підходу полягає в тому, що він дозволяє вибрати для певних застосунків конкретних постачальників.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		21

3.2 Розробка структурної схеми

Гіперконвергенція – об'єднання ІТ-ресурсів у цілісне рішення за допомогою типових апаратних блоків і програмно обумовлених пристроїв – стала одним із самих модних трендів в ІТ. Спробуємо розібратися в перевагах гіперконвергентних систем, напрямках подальшого розвитку рішень і сценаріях їхнього використання в бізнесі.

Етапи розвитку обчислювальних інфраструктур

Віртуалізація машин, що застосовувалася ще в радянський час, дозволила оптимізувати використання обчислювальних ресурсів комп'ютера, відокремити виконання коду від конкретного апаратного забезпечення й створила передумови до виникнення масштабованих віртуальних обчислювальних потужностей на blade-серверах, що містять, що додаються по необхідності «леза» – материнські плати із процесорами, установлювані в blade-кошики.

На початку 90-х з'явилася можливість нарощувати обсяг дискового простору шляхом додавання дисків у виділене сховище даних – СЗД. Для роботи з ними використовувалася розроблена в 1987 році технологія віртуалізації даних RAID, що поєднує диски в логічні елементи, з якими й працювали операційні системи.

Архітектура, що включає кластер комп'ютерів або блейд-серверів і пов'язаних з ними швидкісним каналом (fibre channel) СЗД, де-факто стала стандартом в 2005 році й донині використовується багатьма компаніями.

Цифрова трансформація ставить перед розроблювачами комп'ютерних систем всі нові завдання. Складність рішень постійно росте, обчислювальні системи потрібно запускати швидко, забезпечуючи необхідну гнучкість при подальших змінах конфігурації. У відповідь на запити ринку з'явилися хмарні технології, що надають обчислювальні потужності як сервіс на вимогу, і конвергентні рішення – заздалегідь інтегровані обчислювальні модулі, що включають обчислювальні потужності, системи зберігання, мережне

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		22

устаткування, засоби віртуалізації й оркестрації. Раніше компанія купувала комп'ютери й СЗД, з'єднувала їх і розвертала платформу віртуалізації. Це вимагало багато часу й участі кваліфікованих фахівців. З появою конвергентних систем (у термінології Gartner і інтегрованих інфраструктур) стало досить купити заздалегідь інтегрований модуль (стійку), готовий для установки прикладних програм.

Наступний крок – розгорнути платформу віртуалізації на інтегрованих блоках або серверах, об'єднавши в єдине ціле їхні внутрішні диски/дискові полки на програмному рівні, у такий спосіб формуючи віртуальне відказостійке сховище. З погляду нарощування обчислювальних потужностей гіперконвергентні інфраструктури, що виходять (HCI – Hyper-Converged Infrastructure) уже не обмежені можливостями одного інтегрованого модуля пристрою або сервера – платформа поєднує їхні ресурси в єдине ціле, з якого можна нарізати потрібні для конкретного завдання обчислювальні потужності.

Весь «інтелект» HCI для забезпечення відказостійкості й продуктивності опирається на програмно обумовлені технології (віртуальні машини, програмно обумовлені сховища).

Такий підхід сильно спрощує керування складною системою, що тепер здійснюється через загальну консоль адміністрування, забезпечує більшу гнучкість і масштабованість, а також виконання заздалегідь заданих вимог до відказостійкості й продуктивності. При цьому можна обійтися без дорогих СЗД, відмовитися від мереж fiber channel і зовнішніх мереж зберігання даних – гіперконвергентні системи, як правило, складаються з коммодити-рішень, побудованих на стандартній для галузі архітектурі x86, що значно підвищує економічну ефективність їхнього використання.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		23

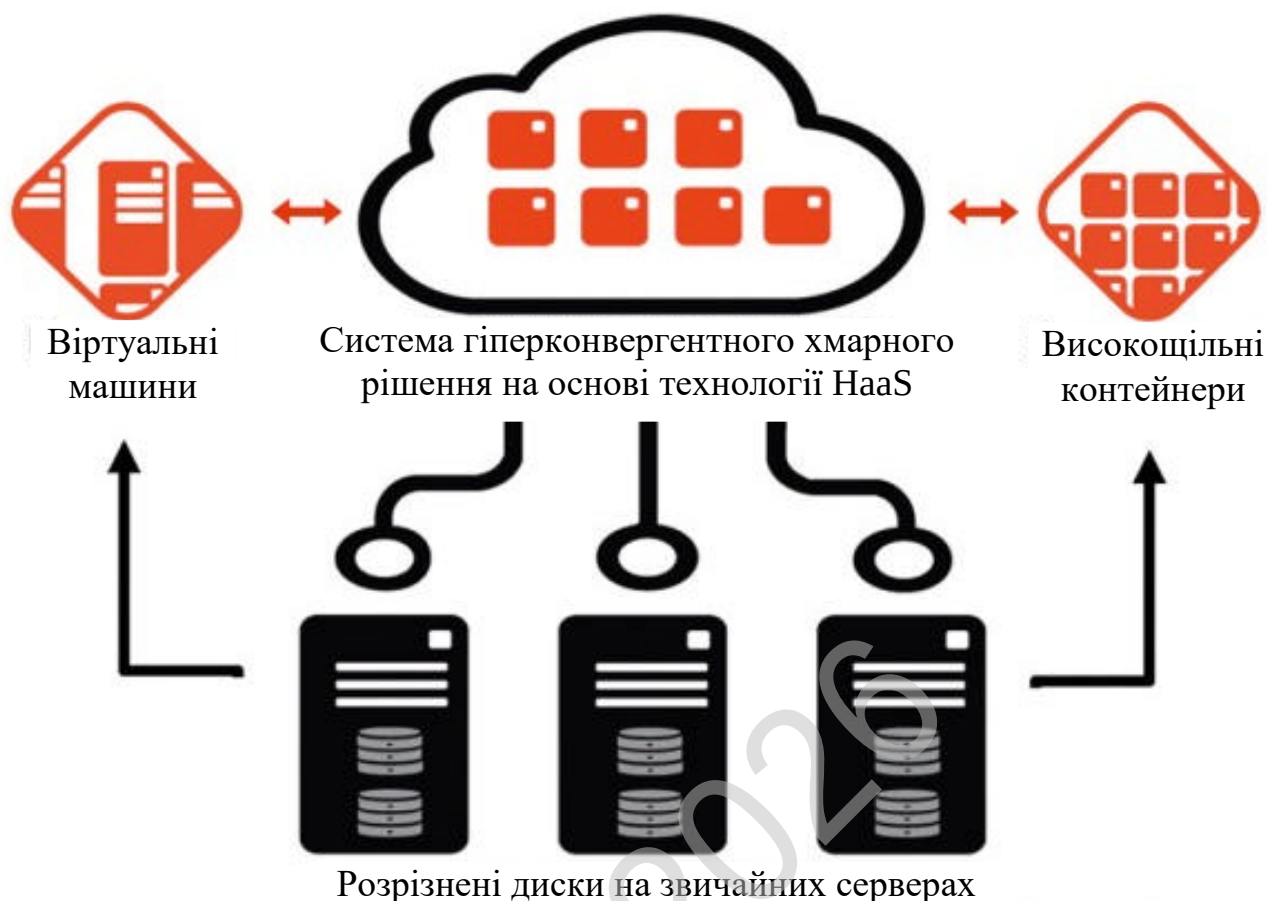


Рисунок 3.1 – Структурна схема системи

У нас просте правило в плані сумісності з «залізом». Якщо на сервер встановлюється Linux, а в серверному сегменті таких моделей переважна більшість, то на цьому сервері заробить і наше ПЗ. Секрет простий – ми опираємося на екосистему Linux-Драйверів для взаємодії з «залізом». При цьому система підтримує будь-які гостьові операційні системи, виконувані усередині системи віртуалізації.

Сценарії застосування

НСІ не є рішенням на всі випадки життя. Масштабування може вийти не збалансованим: наприклад, клієнтові потрібно тільки наростити обчислювальну потужність, а доводиться докуповувати весь блок. Крім того, є проблеми з розгортанням деяких застосунків у віртуальних середовищах.

Два основних сценарії використання гіперконвергентних систем – вилучені офіси й віртуалізовані робітники місця»,. При гомогенному навантаженні, як у випадку VDI, проявляється основна перевага гіперконвергентної системи – легка масштабованість. Допустимо, сьогодні потрібно 1 тис. робочих місць – купуємо десять вузлів HCI. Завтра потрібно 1,5 тис. – купуємо ще п'ять. У такий спосіб сильно спрощується розгортання обчислювальної інфраструктури вилученого офісу або філії. При цьому не треба шукати місцевих фахівців для запуску й підтримки системи, тим більше що їх може й не бути.

Втім, можливі й інші сценарії застосування. Наприклад, об'єднання різнорідних обчислювальних систем при злитті компаній легше здійснити при інтеграції на HCI-системах. Та й не тільки при об'єднанні – міграція с дорогих високонадійних обчислювальних систем, на яких працюють бізнес-критичні застосунки, на комодіті-рішення, використовувані в HCI, може заощадити багато грошей, не знижуючи при цьому показники надійності.

І, звичайно, гнучкі гіперконвергентні системи зручні для розробки й тестування нових бізнес-застосунків.

Перехід на HCI

Найпростіший спосіб перейти на гіперконвергентну систему – купити заздалегідь інтегрований програмно-апаратний комплекс. Такий підхід ефективний при розгортанні нових філій підприємств і банків, створенні нових компаній.

Але вже працюючі на ринку організації мають свою IT-інфраструктуру й не хочуть втрачати зроблені в неї інвестиції. У компаній існують програми закупівель, орієнтовані на конкретних вендорів, є експертиза по роботі з устаткуванням і програмним забезпеченням конкретних виробників.

Постачальники у відповідь на запит ринку почали пропонувати різні варіанти гіперконвергентних систем. Постачальники апаратних рішень забезпечують можливість роботи з різними системами віртуалізації.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		25

Постачальники програмних рішень також провели диверсифікованість апаратних платформ.

Написати програму, що працює з конкретним апаратним комплексом, значно простіше, ніж забезпечити її роботу з усім спектром «заліза», представленим на ринку. Це завдання різного масштабу. Рішення тепер з певними обмеженнями можуть бути запуснені практично на будь-якому стандартному устаткуванні. Але, отут є підводні камені. Робота ПЗ залежить від особливостей апаратної частини й версій її прошивання. У єдине ціле можна об'єднати не тільки обчислювальні потужності й системи зберігання даних, але й віртуалізовані мережі, що вже стає стандартом для HCl. Пропонуються рішення, інтегровані із системами резервування й аварійне відновлення даних.

Таким чином, гіперконвергентні рішення виходять на новий рівень. Якщо раніше пропонувалася тільки інфраструктура, на якій фахівці замовника «піднімали» свої системи, то тепер гіперконвергентні рішення пропонуються як платформа, на зразок хмарного PaaS. Це сильно полегшує працю розроблювачів, дозволяючи автоматизувати створення середовищ виконання програм, розгортання віртуальних машин, контейнерів і мікросервісів.

Ще один напрямок розвитку гіперконвергентних систем – надання їхнім хмарним провайдером в оренду по моделі dedicated. Компанія може не купувати HCl-систему, а користуватися нею із ЦОДу провайдеру по сервісній моделі. Або розгорнути в себе й масштабувати в ЦОД провайдеру, створюючи розтягнутий кластер. Аналогічний кластер можна створити, об'єднавши зі своєї локальної HCl-системою гіперконвергентну, розташовану по моделі colocation у ЦОДі провайдеру.

Сьогодні обчислювальна інфраструктура підприємств містить у собі усе більше різноманітних систем і її розвиток спрямований на перехід від забезпечення працездатності обчислювальних систем до забезпечення доступності й працездатності пропонованих клієнтові сервісів. У цих умовах одне з головних переваг гіперконвергентних систем – спрощення керування обчислювальною інфраструктурою, причому як на площадці замовника, так і в провайдеру.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		26

3.3 Розробка функціональної схеми

Принцип роботи гіперконвергентної інфраструктури:

- Всі важливі процеси ЦОД виконуються на тісно інтегрованому програмному рівні, а не на спеціалізованому устаткуванні.
- Гіперконвергентна платформа складається із трьох програмних компонентів: засобів віртуалізації сховища, засобів віртуалізації обчислювальних ресурсів і системи керування.
- ПЗ для віртуалізації абстрагує й поєднує в пули базові ресурси, а потім динамічно виділяє їхнім застосункам, які виконуються у ВМ або контейнерах.
- Конфігурація визначається політиками, які враховують потреби застосунків, що усуває необхідність у використанні таких складних об'єктів, як дискові й логічні томи.
- Розширені можливості керування скорочують число завдань, виконуваних вручну, і допомагають повністю автоматизувати експлуатацію.

Таким чином, функціональна схема системи складається з наступних блоків:

1. Головне вікно програмного продукту системи гіперконвергентного хмарного рішення на основі технології HaaS.
2. Засоби віртуалізації сховища:
 - Блок абстрагування базових ресурсів.
 - Блок об'єднання в пули базових ресурсів.
 - Блок динамічного виділення базових ресурсів застосункам, які виконуються у віртуальних машинах або контейнерах
3. Засоби віртуалізації обчислювальних ресурсів.
4. Система керування.
5. Система конфігурації, містить у собі блок визначення політик, які враховують потреби застосунків.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		27

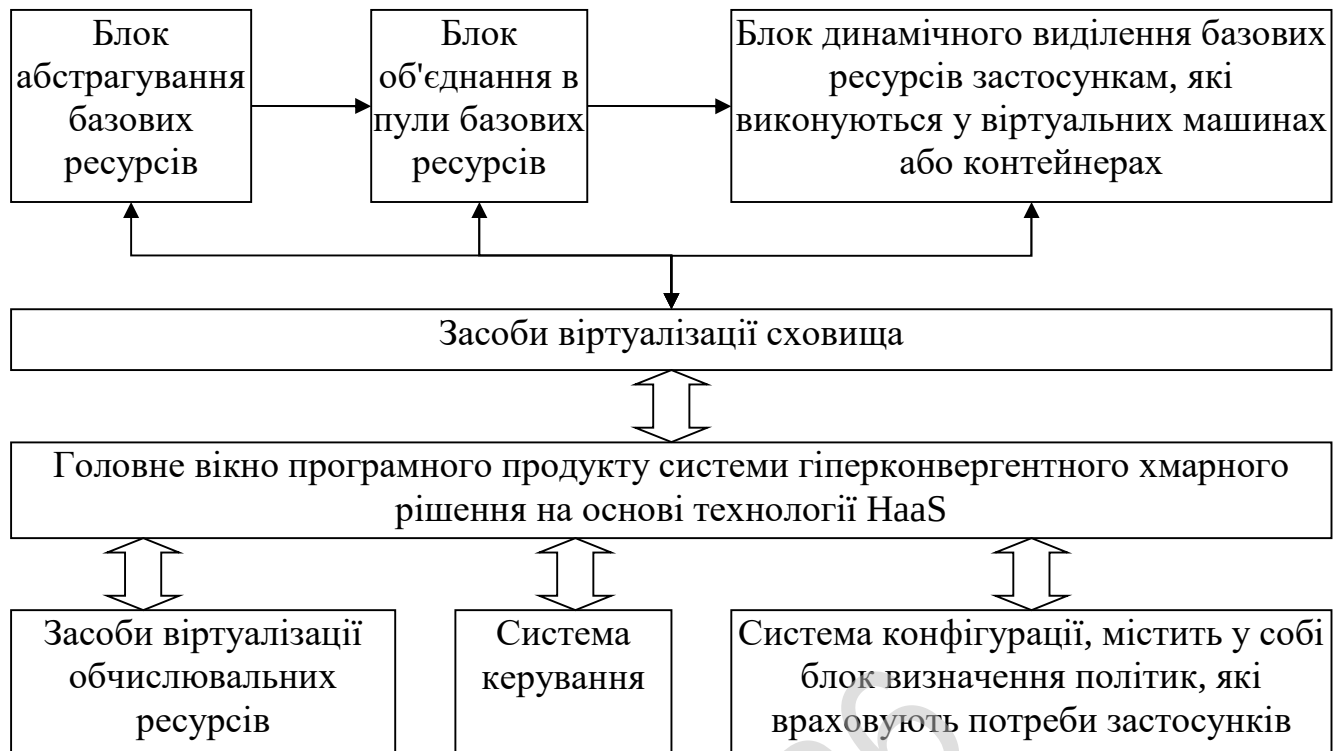


Рисунок 3.2 – Функціональна схема системи

Розходження між гіперконвергентною й конвергентною інфраструктурою

Гіперконвергентні й конвергентні ІТ-інфраструктури поєднують чотири компоненти ЦОД. При цьому гіперконвергентні системи роблять це за допомогою програмного забезпечення (тому вони не мають прив'язки до певного устаткування), а конвергентні рішення залежать від устаткування. Для створення конвергентної інфраструктури ЦОД використовуються в основному ті ж продукти, що й у традиційної ІТ-середовищу, тільки зі спрощеною архітектурою й оптимізованим керуванням.

Переваги гіперконвергентної інфраструктури перед традиційною трирівневою архітектурою

Спрощення експлуатації

Усунете процеси, виконувані вручну, і необхідність у розрізнених групах фахівців для керування різними процесами експлуатації. Один об'єднаний ІТ-

відділ може виконувати моніторинг обчислювальних ресурсів і сховища й управляти ними, завдяки чому забезпечується економія часу співробітників.

Скорочення витрат

Скоротите капітальні витрати завдяки використанню горизонтально й вертикально масштабованої архітектури на базі стандартних серверів x86 і економічних спеціалізованих мереж. Надалі можна додавати ресурси в міру необхідності без переривання процесів.

Підвищена адаптивність

Прискорте реагування на швидко мінливі потреби бізнесу. Устаткування можна запустити за кілька годин, а робітники навантаження ініціалізуються за кілька хвилин. Підвиште продуктивність важливих застосунків, наприклад реляційних баз даних.

Впровадження гіперконвергентної інфраструктури: три фактори, які варто врахувати

Переваги гіперконвергентної інфраструктури

Гіперконвергентна інфраструктура допомагає усунути розрізненість, характерну для традиційної IT-середовища, і реалізувати комплексне керування за допомогою єдиного засобу, завдяки чому можна знизити операційні й капітальні витрати, пов'язані з IT-інфраструктурою, без шкоди для безпеки, гнучкості або масштабованості.

Яким образом рішення інтегрується з існуючим середовищем?

Для реалізації максимальних переваг і наступного переходу до повністю програмно-визначаємого ЦОД гіперконвергентна інфраструктура не повинна бути прив'язана до конкретної апаратної платформи. Гіперконвергентна інфраструктура VMware надає максимальну волю вибору й гнучкість, завдяки чому можна вибрати кращу платформу, забезпечити повну інтеграцію з наявною інфраструктурою, а також зберегти й оптимізувати існуючі технології.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		29

Чи надає рішення можливість ефективно й економічно збільшувати обсяг ресурсів у міру необхідності в зручний час, у будь-якому місці й зручному способі?

Масштабування ЦОД може бути дорогим і складним процесом. Гіперконвергентна інфраструктура забезпечує масштабованість і зручність, допомагаючи прискорити реагування на швидко мінливі потреби бізнесу. На даний момент— це єдиний постачальник, що пропонує повний набір продуктів для створення повністю програмно-визначаємого ЦОД у виробничому середовищі. Завдяки цьому забезпечується адаптивність для розгортання застосунків, стандартизації ІТ-рішень у різних середовищах і підготовки ІТ-середовища до використання загальнодоступних хмар.

Переваги гіперконвергентної інфраструктури

Спрощений перехід до повного стеку рішень для гіперконвергентної інфраструктури

- Збереження інтеграції з гіпервізором і провідними хмарними платформами.
- Використання наявних засобів віртуалізації.
- Захист і оптимізація існуючих інвестицій.

Найвища гнучкість

- Використання сертифікованих рішень постачальника, що воліється.
- Підтримка новітніх технологій при розгортанні.
- Об'єднання ресурсів за допомогою керування на основі політик.

Підтримка багатохмарних середовищ

- Погоджені процеси в ЦОД, хмарі й на периметрі мережі.
- Убудована система безпеки для даних на зберіганні й передані даних.
- Підтримка загальнодоступних хмар сотень постачальників.

3.4 Розробка діаграми процесів

Розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3. Основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей.

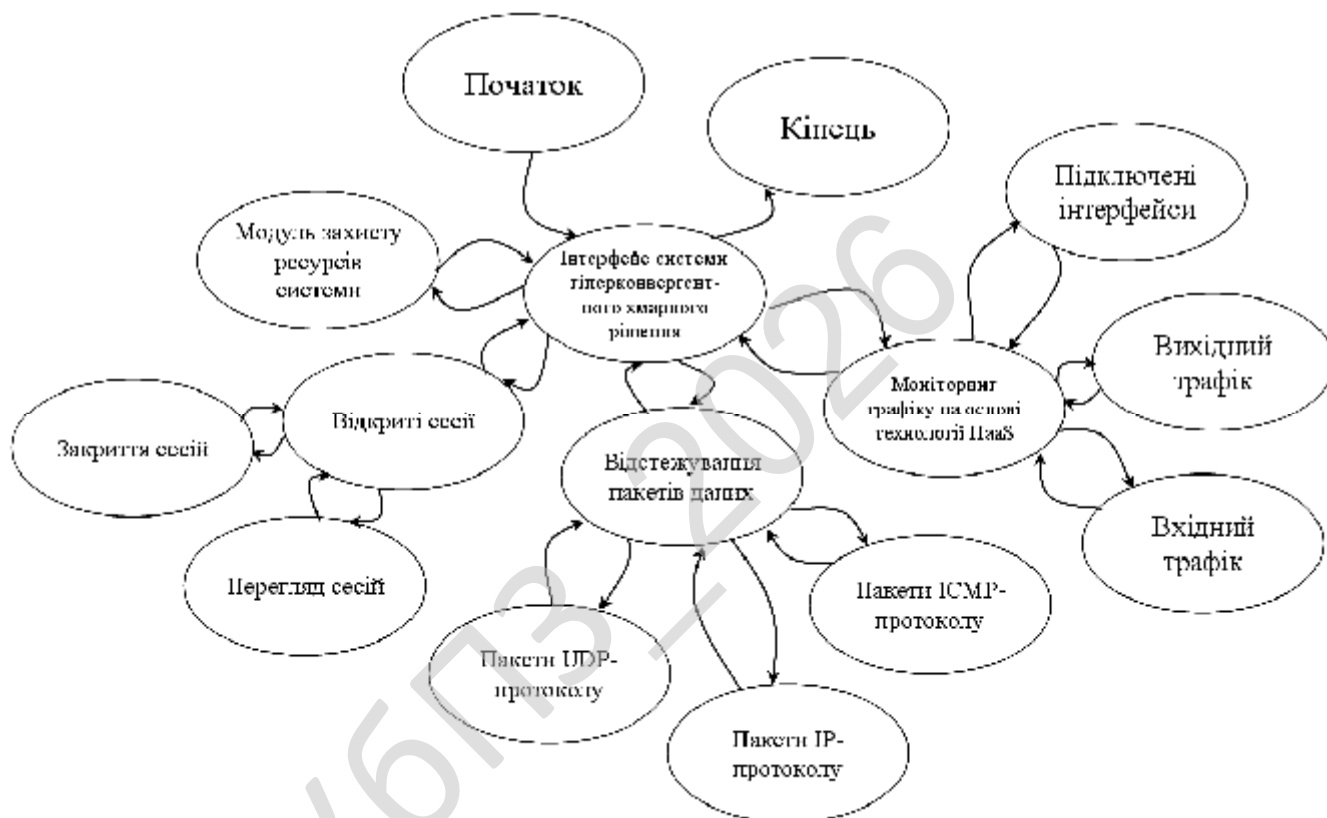


Рисунок 3.3 – Діаграма взаємодії процесів

Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

При детальному її розгляді можна побачити як саме проходить взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграми потоків даних містять чотири типи елементів:

- Зовнішні по відношенню до системи сутності.
- Потоки даних між елементами трьох попередніх типів.
- Процеси які являють собою трансформацію даних в рамках описуваної системи.
- Сховища даних (репозиторії).

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо реалізацію бакалаврської дипломної роботи. Були проведені розрахунки і підібрані набори тестових даних для перевірки правильності реалізації проектних рішень.

Було створено блок-схеми роботи системи. Перед їх розглядом необхідно провести роз'яснення який саме тип блок-схем використовується. Блок-схеми показують весь процес роботи системи з підсистемами та частково доказують правильність вибраних проектних рішень. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об'єктно-орієнтована що вимагає високого рівня декомпозиції задач на класи.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підсистеми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		33

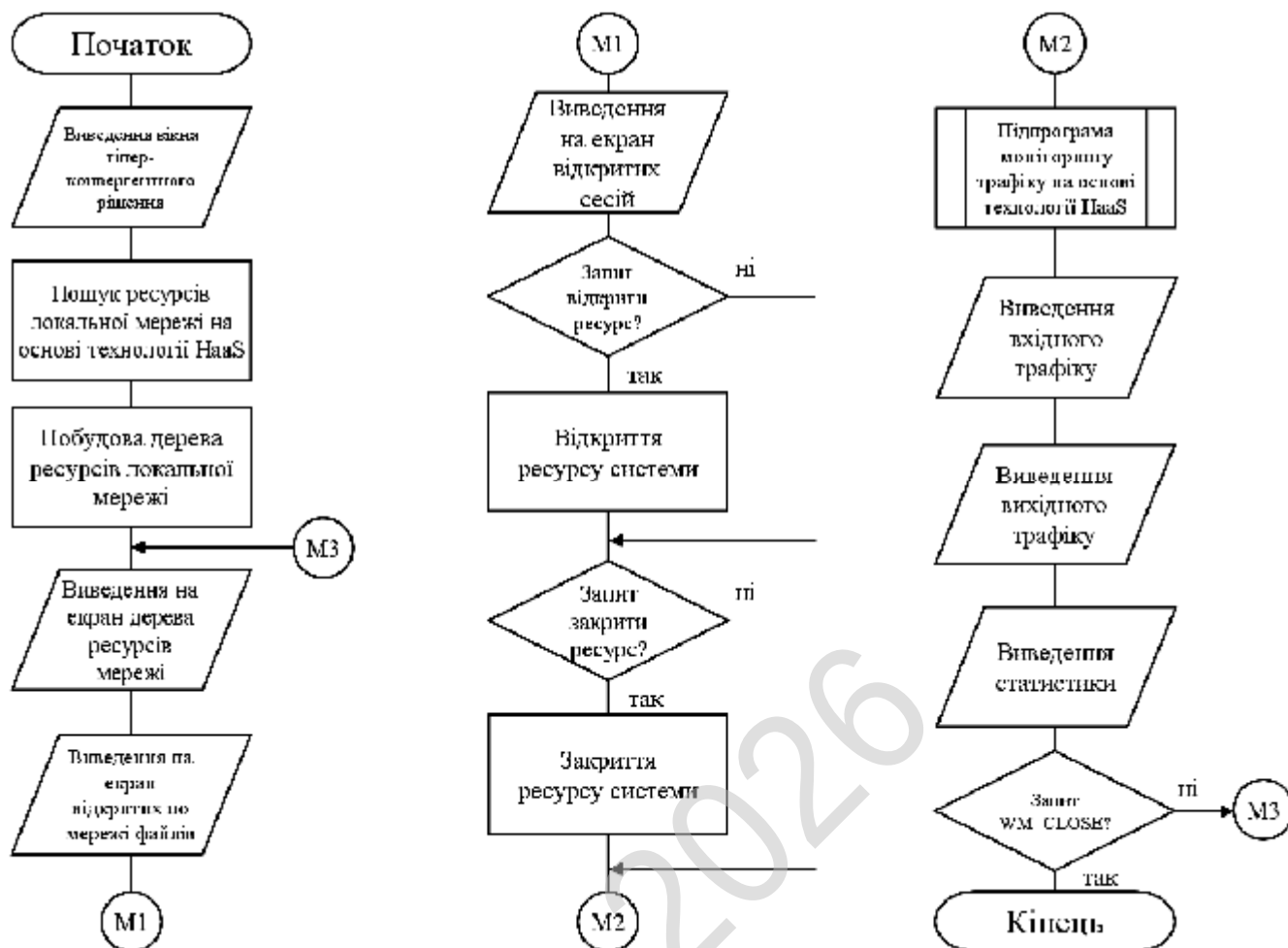


Рисунок 4.1 – Блок-схема основної програми

Було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); Діаграма розгортання.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності.

Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

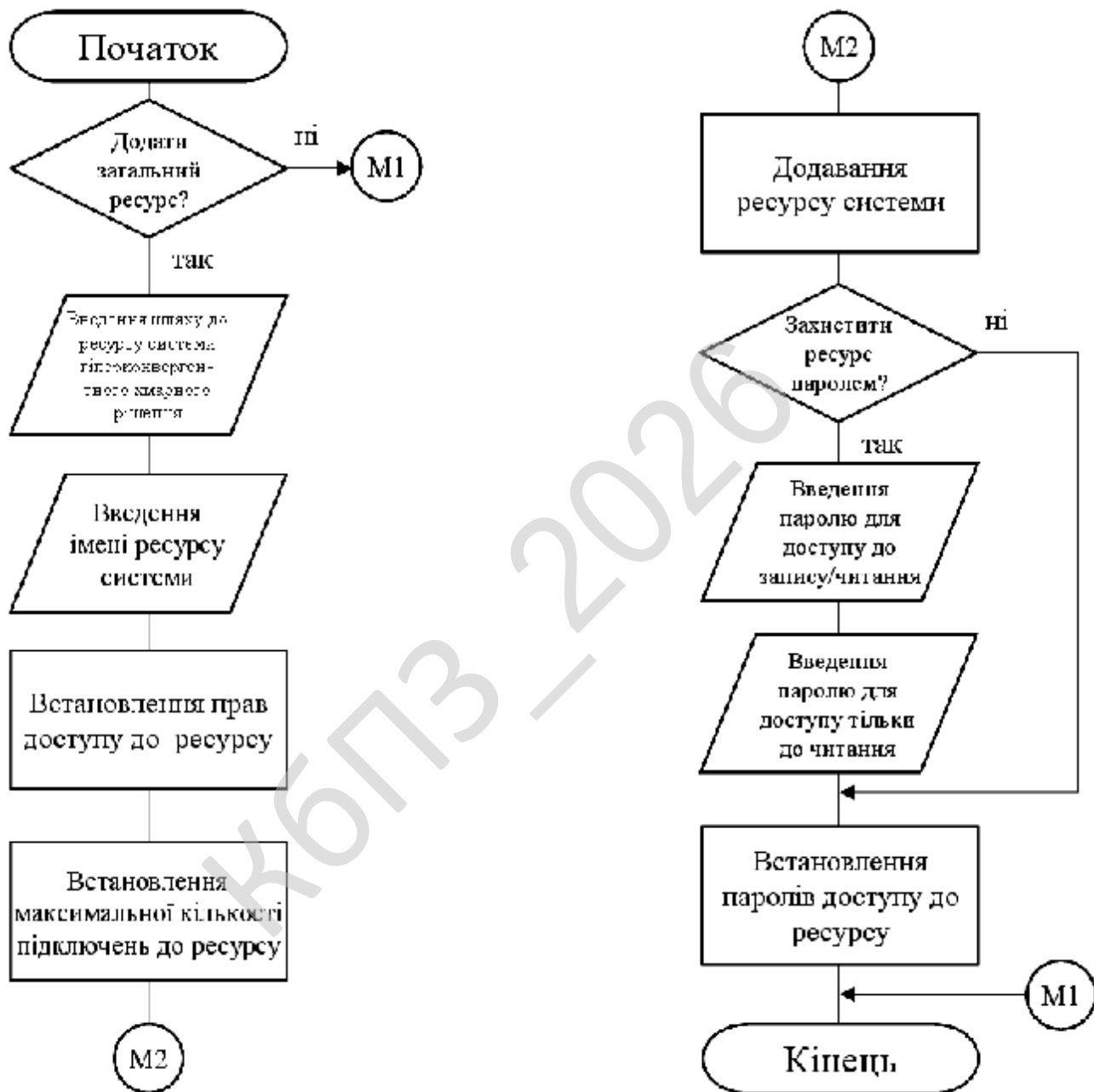


Рисунок 4.2 – Блок-схема роботи підпрограми

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних програм і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
- набір клієнтів, які використовують сервіси, що надаються серверами;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт». Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших комп'ютерів. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		36

відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є положення, що клієнти та сервери – це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

- рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;
- прикладний рівень, який реалізує основну логіку ПЗ і на якому здійснюється необхідна обробка інформації;
- рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

- модель тонкого клієнта, в рамках якої вся логіка ПЗ та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;
- модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Типовим прикладом клієнт-серверної взаємодії є WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		37

найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

Для того, щоб людина, яка працює в Інтернеті, могла переглянути ту чи іншу сторінку, на її комп'ютері повинно бути встановлено відповідне програмне забезпечення. Програми для перегляду веб-сторінок називаються браузером.

Але, крім браузерів, до серверів можуть звертатися і інші клієнти, а саме – автономні програми. Вони можуть передбачати взаємодію з людиною, а можуть працювати в цілком автоматичному режимі. Типовим класом таких програм є роботи, призначені для автоматичного перегляду веб-ресурсів. Зокрема, роботи є важливим елементом пошукових систем і використовуються ними для перегляду сторінок і збору інформації про них.

Для запиту до веб-сервера клієнтська програма повинна задати місцезнаходження комп'ютера, на якому розміщується серверна програма, назву потрібного документа і, можливо, інші дані, які специфікують запит. Мережа забезпечує знаходження сервера і передачу йому клієнтського запиту. Серверні програми обробляють цей запит, відповідь пересилається по мережі клієнтові.

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка ПЗ. Програми проміжного рівня можуть функціонувати під управлінням спеціальних серверів ПЗ, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення – звичайний браузер. Це позбавляє його необхідності завантажувати

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		38

та інсталиувати спеціальні програми (хоча інколи така необхідність все-таки виникає).

Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на веб-сторінках та завантажуються разом з ними.

Веб-оглядач формує запит та пересилає його до сервера, який здійснює обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних. Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше веб-сервер і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі.

На сучасному етапі для програмування модулів проміжного рівня використовується мова серверних сценаріїв PHP, а для управління даними – СУБД MySQL. Таким чином, зв'язку PHP-MySQL слід розглядати як стандартний інструмент для створення порівняно простих інтерактивних веб-сайтів та систем електронної комерції; близько 90% комерційних систем сьогодні створюється саме на цій основі. Водночас як засоби управління даними, так і middleware-засоби можуть бути найрізноманітнішими. Так, для створення серверних програм, крім PHP, широко застосовуються Java, Perl, Python, Delphi.

Взагалі, технології створення розподілених, зокрема веб-програм, стрімко розвиваються. Слід згадати про технології EJB (Enterprise Java Beans), CORBA, а також про .NET – порівняно нову ініціативу компанії Microsoft. Для зберігання

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		39

даних та їх передачі часто використовується так звана розширювана мова розмітки XML (Extensible Markup Language).

Розглянемо протокол SNMP (Simple Network Management Protocol, простий протокол керування мережею) – це протокол керування мережами зв'язку на основі архітектури TCP/IP.

На основі концепції TMN в 1980–1990 р. різними органами стандартизації був вироблений ряд протоколів керування мережами передачі даних з різним спектром реалізації функцій TMN. До одного з типів таких протоколів керування належить Simple Network Management Protocol (SNMP).

SNMP – це технологія, покликана забезпечити керування й контроль за пристроями й програмами в мережі зв'язку шляхом обміну керуючою інформацією між агентами, що розташовуються на мережних пристроях, і менеджерами, розташованими на станціях керування. SNMP визначає мережу як сукупність мережних керуючих станцій й елементів мережі (головні машини, шлюзи й маршрутизатори, термінальні сервери), які спільно забезпечують адміністративні зв'язки між мережними керуючими станціями й мережними агентами. SNMP різних версій присвячений цілий ряд рекомендацій IETF (RFC).

Зазвичай при використанні SNMP присутні керовані та керуючі системи. До складу керованої системи входить компонент, який називається агентом, який відправляє звіти керуючій системі. По суті SNMP агенти передають управлінську інформацію на керуючі системи як змінні (такі як «вільна пам'ять», «ім'я системи», «кількість працюючих процесів» тощо).

Керуюча система може отримати достовірну інформацію через операції протоколу GET, GETNEXT і GETBULK. Агент може самостійно без запиту надсилати дані, використовуючи операцію протоколу TRAP або INFORM. Управляючі системи можуть також відправляти конфігураційні оновлення або контролюючі запити, використовуючи операцію SET для безпосереднього управління системою. Операції конфігурування та управління використовуються

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		40

тільки тоді, коли потрібні зміни у мережній інфраструктурі. Операції моніторингу зазвичай виконуються на регулярній основі.

Змінні, доступні через SNMP, організовані в ієрархії. Ці ієрархії та інші метадані (такі як тип і опис змінної) описуються Базами Керуючої Інформації' (Management Information Bases (MIBs)).

SNMP не визначає, яку інформацію (які змінні) керована система повинна надавати. Навпаки, SNMP використовує розширювану модель, в якій доступна інформація визначається Базами Керуючої Інформації (MIB – Management Information Base).

Бази Керуючої Інформації описують структуру керуючої інформації пристроїв. Вони використовують ієрархічний адресний простір імен, що містить унікальний ідентифікатор об'єкта (object identifier (OID)).

Грубо кажучи, кожен унікальний ідентифікатор об'єкта ідентифікує змінну, яка може бути прочитана чи встановлена через SNMP. MIB'и використовують нотацію, визначену в ASN.1.

Ієрархія MIB може бути зображена як дерево з безіменним коренем, рівні якого приписані різними організаціями. На найвищому рівні MIB OID'и належать різним організаціям, що займаються стандартизацією, в той час як на нижчих рівнях OID'и виділяються асоційованими організаціями. Ця модель забезпечує управління на всіх шарах мережної моделі OSI, адже MIB'и можуть бути визначені для будь-яких типів даних і операцій.

Керований об'єкт – це одна з будь-якого числа характеристик, специфічних для керованого пристрою. Керований об'єкт включає в себе один або більше екземплярів об'єкта (що ідентифікуються за OID), які насправді є змінними.

Існує два типи керованих об'єктів: Скалярні об'єкти визначають єдиний екземпляр об'єкта; Табличні об'єкти визначають множинні, пов'язані екземпляри об'єктів які групуються в таблицях MIB.

Прикладом керованого об'єкта може бути atInput, який є скалярним об'єктом що містить єдиний екземпляр об'єкта, ціле число, яке показує загальну кількість вхідних пакетів AppleTalk на мережний інтерфейс маршрутизатора.

Ідентифікатор об'єкта (OID) унікально ідентифікує керований об'єкт в ієрархії MIB.

В телекомунікаціях і комп'ютерних мережах, ASN.1 є стандартною гнучкою нотацією для опису структур даних, що служать для кодування, передачі і декодування даних. ASN.1 являє собою набір правил для опису структури об'єктів, незалежних від специфічних для обладнання методик кодування, і формальну нотацію, яка дозволяє уникнути неоднозначностей.

ASN.1 це єдиний ISO та ITU-T стандарт, спочатку визначений у 1984 році як частина стандарту CCITT X.409: 1984. Пізніше, в 1988 році, завдяки його широкому застосуванню, він був виділений в окремий стандарт X.208. Значно переглянута версія 1995 року описана в X.680.

Адаптована підмножина ASN.1 Структура Керуючої Інформації (SMI) описана в протоколі SNMP для визначення наборів пов'язаних MIB об'єктів, також званих MIB модулями.

SNMP працює на прикладному рівні TCP/IP (сьомий рівень моделі OSI). Агент SNMP отримує запити по UDP-порту 161. Менеджер може посилати запити з будь-якого доступного порту джерела на порт агента. Відповідь агента буде відправлений назад на порт джерела на менеджері. Менеджер отримує повідомлення (Traps і InformRequests) по порту 162. Агент може генерувати повідомлення з будь-якого доступного порту. При використанні TLS або DTLS запити виходять по порту 10161, а пастки відправляються на порт 10162.

У SNMPv1 зазначено п'ять основних протокольних одиниць обміну (protocol data units – PDU). Ще дві PDU, GetBulkRequest і InformRequest, були введені в SNMPv2 і перенесені в SNMPv3.

Нижче перераховані сім протокольних одиниць обміну SNMP:

1. GetRequest. Запит від менеджера до об'єкту для отримання значення змінної або списку змінних. Необхідні змінні вказуються в полі variable bindings (розділ поля values при цьому не використовується). Отримання значень зазначеної змінної повинно бути виконано агентом як Атомарна операція. Менеджеру буде повернений Response (відповідь) з поточними значеннями.

2. SetRequest. Запит від менеджера до об'єкту для зміни змінної або списку змінних. Зв'язані змінні вказуються в тілі запиту. Зміни всіх зазначених змінних повинні бути виконані агентом як атомарна операція. Менеджеру буде повернений Response з (поточними) новими значеннями змінних.

3. GetNextRequest. Запит від менеджера до об'єкту для виявлення доступних змінних і їх значень. Менеджеру буде повернений Response зі зв'язаними змінними для змінної, яка є наступною в базі MIB в лексикографічному порядку. Обхід всієї бази MIB агента може бути проведений ітераційним використанням GetNextRequest, починаючи з OID 0. Рядки таблиці можуть бути прочитані, якщо вказати в запиті OID-и колонок в пов'язаних змінних.

4. GetBulkRequest. Покращена версія GetNextRequest. Запит від менеджера до об'єкту для численних ітерацій GetNextRequest. Менеджеру буде повернений Response з декількома пов'язаними змінними, обійденими починаючи зі пов'язаної змінної (змінних) в запиті. Специфічні для PDU поля non-repeaters і max-repetitions використовуються для контролю за поведінкою відповіді. GetBulkRequest був введений в SNMPv2.

5. Response. Повертає зв'язані змінні і значення від агента менеджеру для GetRequest, SetRequest, GetNextRequest, GetBulkRequest і InformRequest. Повідомлення про помилки забезпечуються полями статусу помилки і індексу помилки.

Ця одиниця використовується як відповідь і на Get-, і на Set-запити, в SNMPv1 називається GetResponse.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		43

6. Trap. Асинхронне повідомлення від агента – менеджера. Включає в себе поточне значення sysUpTime, OID, що визначає тип trap (пастки), і необов'язкові зв'язані змінні. Адресація одержувача для пасток визначається за допомогою змінних trap-конфігурації в базі MIB. Формат trap-повідомлення був змінений в SNMPv2 і PDU перейменували в SNMPv2-Trap.

7. InformRequest. Асинхронне повідомлення від менеджера менеджера або від агента менеджера. Повідомлення від менеджера менеджера були можливі вже в SNMPv1 (за допомогою Trap), але SNMP зазвичай працює на протоколі UDP, в якому доставка повідомлень не гарантована, і не повідомляється про втрачені пакетах. InformRequest виправляє це зворотним відправленням підтвердження про отримання. Одержувач відповідає Response-му, що повторює всю інформацію з InformRequest. Цей PDU був введений в SNMPv2.

Розглянемо визначення API. Це прикладний програмний інтерфейс (Application Programming Interface, API) – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

Спрощено – це набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

Одним з найпоширеніших призначень API є надання набору широко використовуваних функцій, наприклад для малювання вікна чи іконок на екрані. API є абстрактним поняттям — програмне забезпечення, що пропонує деякий API, часто називають реалізацією (implementation) даного API. У багатьох випадках API є частиною набору розробки програмного забезпечення, водночас, набір розробки може включати як API, так і інші інструменти/апаратне забезпечення, отже ці два терміни не є взаємозамінювані.

Високорівневі API часто програють у гнучкості. Виконання деяких функцій нижчого рівня стає набагато складнішим, або навіть неможливим. В об'єктно-орієнтованих мовах, прикладний програмний інтерфейс зазвичай

включає в себе опис набору визначень класу, з набором форм поведінки, пов'язаних з цими класами. Це абстрактне поняття пов'язане з реальними функціями, які надані або надаватимуться, класами, які реалізуються в методах класу. Прикладний програмний інтерфейс в даному випадку можна розглядати як сукупність всіх методів, які публічно доступні в класах (зазвичай званий інтерфейс класу). Це означає, що прикладний програмний інтерфейс вказує методи, за допомогою яких взаємодіє з об'єктами, отриманими з визначень класів і обробляє їх.

У більш загальному плані можна визначити Прикладний Програмний Інтерфейс як сукупність усіх видів об'єктів, які можна вивести з визначення класу, і пов'язаних з ними можливих варіантів поведінки.

Наприклад: клас, що представляє Stack, може просто виставити публічно два методи Push() (для додавання нового елемента в стек) і Pop() (для вилучення останнього пункту, ідеально розташований на вершині стека).

У цьому випадку Прикладний Програмний Інтерфейс може бути інтерпретованим як два методи pop() і push(), або, більш широко, використовується варіант, коли можна використовувати елемент типу Stack, який реалізує поведінку стека, надаючи йому можливість для додавання / видалення елементів з вершини. Друга інтерпретація видається більш доречною в дусі об'єктно-орієнтованого підходу.

Якість документації, пов'язаної з Прикладним Програмним Інтерфейсом, є часто ключовим фактором, що визначає його успішність з точки зору простоти використання.

ППІ, як правило, пов'язаний із бібліотеками програмного забезпечення: ППІ описує і вказує очікувану поведінку в той час, як бібліотека є фактичною реалізацією даного набору правил. Один ППІ може мати декілька реалізацій (або жодної, будучи абстрактним) у вигляді різних бібліотек, які мають такий же інтерфейс.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		45

Прикладний програмний інтерфейс також може бути пов'язаним з платформами програмування: платформа може бути заснована на кількох бібліотеках реалізує декілька інтерфейсів ППІ, але на відміну від звичайного використання ППІ, доступ до поведінки вбудований в платформу опосередкований шляхом розширення його змісту новими класами і вставлений в саму платформу. Крім того, загальний потік управління програми може бути під контролем абонента.

Прикладний програмний інтерфейс може бути також реалізацією протоколу.

Коли ППІ реалізує протокол, він може бути заснованим на проксі-методах віддалених викликів, що засновані на протоколі зв'язку. Роль ППІ може заключатися саме в тому, щоб приховати деталі транспортного протоколу. Наприклад: RMI є ППІ, який реалізує протокол або JRMP IIOP як RMI-IIOP.

Протоколи, як правило, розподіляються між різними технологіями і зазвичай дозволяють різним технологіям обмінюватися інформацією, діючи як абстракція між двома світами. ППІ, як правило, є специфічним для конкретної технології: звідси, інтерфейси даної мови не можуть бути використані на інших мовах, якщо виклики функції не будуть перетворені з конкретної адаптації бібліотеки.

Деякі мови, серед яких такі, що працюють на віртуальних машинах (наприклад: мови, сумісні з NET CLI середовища CLR і JVM сумісних мов у віртуальній машині Java) можуть ділитися програмними інтерфейсами.

У цьому випадку віртуальна машина дозволяє мові взаємодії завдяки спільному знаменнику віртуальної машини, що абстрагується від конкретної мови, використовувати проміжний байт-код і його мову.

При використанні прикладного програмного інтерфейсу в контексті веб-розробки, як правило, ППІ визначається набором повідомлень запиту HTTP, також визначається структура повідомлень-відповідей, зазвичай у розширенні мови розмітки XML або в форматі об'єктного запису JavaScript (JSON). У той час

як прикладний програмний інтерфейс у Web історично був практично синонімом для веб-служби, останнім часом тенденція змінилась (так званий Web 2.0) на відхід від Simple Object Access Protocol (SOAP) на основі веб-сервісів і сервіс-орієнтованої архітектури (SOA) на більш прямі передачі репрезентативного стану (REST) стилів веб-ресурсів та ресурсів-орієнтованої архітектури (ROA).

Частина цієї тенденції пов'язана з рухом Семантичного веб-ресурсу до Опису Платформ (RDF), Концепції розвитку веб-технологій інженерних онтологій. Прикладні програмні інтерфейси у Web, що дозволяють комбінувати декількома прикладними програмними інтерфейсами в нові додатки називають гібридними.

При розробці ПЗ було використано V-Model (або VEE модель) є моделлю розробки інформаційних систем (IC), спрямованої на спрощення розуміння складнощів, пов'язаних з розробкою систем. Вона використовується для визначення єдиної процедури розробки програмного забезпечення, апаратного забезпечення та людино-машинного інтерфейсу.

Концепція V-подібної моделі була розроблена Німеччиною та США в кінці 1980-х років незалежно один від одного:

– Німецька V-модель була розроблена аерокосмічної компанією IABG в Оттобрунні поряд з Мюнхеном у сприянні з Федеральним департаментом з закупівлі озброєнь в Кобленці, для Міністерства оборони Німеччини. Модель була прийнята німецькою федеральною адміністрацією для цивільних потреб влітку 1992.

– Американська V-Model (VEE) була розроблена національною радою з системної інженерії (міжнародна – з 1995 року) для супутникових систем, включаючи обладнання, програмне забезпечення та взаємодію з користувачами.

Сучасною версією V-Model є V-Model XT, яка була затверджена в лютому 2005 року. V-модель використовується для управління процесом розробки програмного забезпечення для німецької федеральної адміністрації.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		47

Зараз вона є стандартом для німецьких урядових і оборонних проектів, а також для виробників ПЗ в Німеччині. V-Model являє собою скоріше набір стандартів у галузі проектів, що стосуються розробки нових продуктів. Ця модель багато в чому схожа з Prince2 і описує методи як для проектного управління, так і для системного розвитку.

Основні принципи

Основний принцип V-подібної моделі полягає в тому, що деталізація проекту зростає при русі зліва направо, одночасно з плином часу, і ні те, ні інше не може повернути назад. Ітерації в проекті виробляються по горизонталі, між лівою і правою сторонами літери.

Стосовно до розробки інформаційних систем V-Model – варіація каскадної моделі, в якій завдання розробки йдуть зверху вниз по лівій стороні букви V, а завдання тестування – вгору по правій стороні букви V. Усередині V проводяться горизонтальні лінії, що показують, як результати кожної з фаз розробки впливають на розвиток системи тестування на кожній із фаз тестування.

Модель базується на тому, що прийнятно-здавальні випробування ґрунтуються, насамперед, на вимогах, системне тестування – на вимогах та архітектури, комплексне тестування – на вимогах, архітектурі та інтерфейсах, а компонентне тестування – на вимогах, архітектурі, інтерфейсах та алгоритмах

Цілі

V-модель забезпечує підтримку у плануванні та реалізації проекту. В ході проекту ставляться такі завдання:

1. Мінімізація ризиків: V-подібна модель робить проект більш прозорим і підвищує якість контролю проекту шляхом стандартизації проміжних цілей і опису відповідних їм результатів та відповідальних осіб. Це дозволяє виявляти відхилення в проекті і ризики на ранніх стадіях і покращує якість управління проектом.

2. Підвищення та гарантії якості: V-Model – стандартизована модель розробки, що дозволяє домогтися від проекту результатів бажаної якості.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		48

Проміжні результати можуть бути перевірені на ранніх стадіях. Універсальне документування полегшує читаність, зрозумілість та контрольованість.

3. Зменшення загальної вартості проекту: Ресурси на розробку, виробництво, управління і підтримку можуть бути заздалегідь прораховані та проконтрольовані. Отримувані результати також універсальні і легко прогножуються. Це зменшує витрати на подальші стадії та проекти.

4. Підвищення якості комунікації між учасниками проекту: Універсальний опис усіх елементів та умов полегшує взаєморозуміння всіх учасників проекту. Таким чином, зменшуються неточності у розумінні між користувачем, покупцем, постачальником і розробником.

Переваги:

– Користувачі V-Model беруть участь у розробці та підтримці V-моделі. Комітет з контролю за змінами підтримує проект і збирається раз на рік для обробки всіх отриманих запитів на внесення змін до V-Model.

– На старті будь-якого проекту V-подібна модель може бути адаптована під цей проект, так як ця модель не залежить від типів організацій та проектів.

– V-model дозволяє розбити діяльність на окремі кроки, кожен з яких буде включати в себе необхідні для нього дії, інструкції до них, рекомендації та докладне пояснення діяльності.

Обмеження. Наступні моменти не враховуються в V-моделі, але можуть бути розглянуті окремо, або можливо адаптувати модель під них:

– Не регулюється розміщення контрактів на обслуговування.

– Організація і виконання управління, обслуговування, ремонту та утилізації системи не враховуються в V-моделі. Однак, планування і підготовка до цих операцій моделлю розглядаються.

– V-подібна модель більше стосується розробки програмного забезпечення в проекті, ніж всієї організації процесу.

Переваги:

– У моделі особливе значення надається плануванню, спрямованому на верифікацію та атестацію розроблювального продукту на ранніх стадіях його розробки. Фаза модульного тестування підтверджує правильність деталізованого проектування. Фази інтеграції та тестування реалізують архітектурне проектування або проектування на вищому рівні. Фаза тестування системи підтверджує правильність виконання етапу вимог до продукту і його специфікації.

– У моделі передбачені атестація та верифікація всіх зовнішніх і внутрішніх отриманих даних, а не тільки самого програмного продукту.

– У V-подібної моделі визначення вимог виконується перед розробкою проекту системи, а проектування ПЗ – перед розробкою компонентів.

– Модель визначає продукти, які повинні бути отримані в результаті процесу розробки, причому кожні отримані дані повинні піддаватися тестуванню.

– Завдяки моделі менеджери проекту можуть відслідковувати хід процесу розробки, так як в даному випадку цілком можливо скористатися тимчасовою шкалою, а завершення кожної фази є контрольною точкою.

Недоліки:

– Модель не передбачає роботу з паралельними подіями.

– У моделі не передбачено внесення вимоги динамічних змін на різних етапах життєвого циклу.

– Тестування вимог в життєвому циклі відбувається занадто пізно, внаслідок чого неможливо внести змін, не вплинувши при цьому на графік виконання проекту.

– У модель не входять дії, спрямовані на аналіз ризиків.

– Деякий результат можна отримати тільки при досягненні низу букви V.

Jira – була використана комерційна система відслідковування помилок, призначена для організації взаємодії з користувачами, хоча в деяких випадках використовується і для управління проектами. Розроблено компанією Atlassian, є

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		50

одним з двох її основних продуктів (поряд з вікі-системою Confluence). Має веб-інтерфейс.

Назва системи отримано шляхом усічення слова «Gojira» – Японського імені монстра Годзилла, що, в свою чергу, є відсиланням до назви конкуруючого продукту – Bugzilla; створювалася в якості заміни Bugzilla і багато в чому повторює її архітектуру. Система дозволяє працювати з декількома проектами. Для кожного з проектів створює і веде схеми безпеки і схеми оповіщення.

До версії 3.13.5 (включно) розрізнялися редакції Enterprise, Professional і Standard, після – Залишилася тільки редакція Enterprise (для великих організацій).

Система заснована на Java EE і працює на кількох популярних системах управління базами даних і операційних системах.

Основний елемент обліку в системі – завдання (ticket або issue). Завдання містить назву проекту, тему, тип, пріоритет, компоненти і зміст. Завдання може бути розширена додатковими полями (також і нові призначені для користувача поля можуть бути визначені), додатками (наприклад – Фотографіями, скріншотами) або коментарями. Завдання може редагуватися або просто змінювати статус, наприклад, з «відкритий» в «закритий». Які переходи між станами можливі, визначається через настраюється потік операцій. Будь-які зміни в задачі записуються в журнал.

Jira має велику кількість можливостей конфігурації: для кожної програми може бути визначений окремий тип завдання з власним workflow, набором статусів, одним або декількома видами уявлення (screens). Крім того, за допомогою так званих «схем» можна визначити для кожного індивідуального Jira-проекту власні права доступу, поведінку і видимість полів і багато іншого.

Завдяки універсальному підходу можна пристосувати Jira для багатьох непрофільних завдань, наприклад, керування вимогами, керування ризиками, аж до реалізації невеликої системи бронювання, автоматизації процесу рекрутингу.

Для інтеграції з зовнішніми системами підтримує інтерфейси SOAP, XML-RPC і REST. Поставляється із засобами інтеграції з такими системами управління

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		51

версіями, як Subversion, CVS, Git, Clearcase, Team Foundation Server, Mercurial і Perforce.

Існують доповнення, що дозволяють вбудувати Jira в інтегровані середовища розробки, в тому числі Eclipse і IntelliJ IDEA. Перекладена багатьма мовами, включаючи російську, англійську, японську, німецьку, французьку, іспанську.

Для сторонніх розробників надаються кошти розробки розширень системи – плагінів. Розробники розширень можуть викладати плагіни для продажу на спеціальний розділ сайту Atlassian.

Є комерційним продуктом, який може бути ліцензований для роботи на локальному сервері або доступний в якості віддаленого додатки. Ціноутворення залежить від максимального числа користувачів, при цьому близько \$50 за користувача для локального і \$7 на місяць за користувача для віддаленого доступу є типовими цінами.

Для академічних і комерційних клієнтів доступний повний вихідний код під ліцензією розробника.

Для проектів з відкритим вихідним кодом Atlassian надає спеціальну безкоштовну ліцензію при дотриманні наступних правил:

- проект використовує ліцензії, схвалені Open Source Initiative;
- Вихідний код проекту доступний для скачування;
- у проекту є публічно доступна веб-сайт;
- програмне забезпечення від Atlassian є на веб-сайті проекту.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм Lucifer. Алгоритм Lucifer являє собою мережу перестановок і підстановок, його основні блоки нагадують блоки алгоритму DES.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		52

В DES результат функції f складається операцією XOR із входом попереднього раунду, утворюючи вхід наступного раунду. В S-блоках алгоритму Lucifer 4-бітові входи й виходи, вхід S-блоків являє собою перетасований вихід S-блоків попереднього раунду, входом S-блоків першого раунду служить відкритий текст. Для вибору використовуваного S-блоку із двох можливих використовується біт ключа. (Lucifer реалізує все це в єдиному T-блоці з 9 бітами на вході й 8 бітами на виході). На відміну від алгоритму DES, половини блоку між раундами не переставляються, та й саме поняття половини блоку в алгоритмі Lucifer не використовується. У цього алгоритму 16 раундів, 128-бітові блоки й більше проста, чим в DES, схема розгорнення ключа.

Блок тексту розглядається як ненегативне ціле число, або як кілька незалежних ненегативних цілих чисел. Довжина блоку завжди вибирається рівною ступеню двійки. У алгоритмі Lucifer використовуються наступні типи операцій:

- Таблична підстановка, при якій група біт відображається в іншу групу біт. Це так звані S-box.
- Переміщення, за допомогою якого біти повідомлення переупорядковуються.
- Операція додавання по модулю 2, позначувана XOR або $\oplus$.
- Операція додавання по модулю 2^{32} або по модулю 2^{16} .
- Циклічне зрушення на деяке число біт.

Ці операції циклічно повторюються в алгоритмі, створюючи так звані раунди. Входом кожного раунду є вихід попереднього раунду й ключ, що отриманий по певному алгоритму із ключа шифрування K .

Ключ раунду називається підключем. Алгоритм шифрування може бути представлений у такий спосіб:

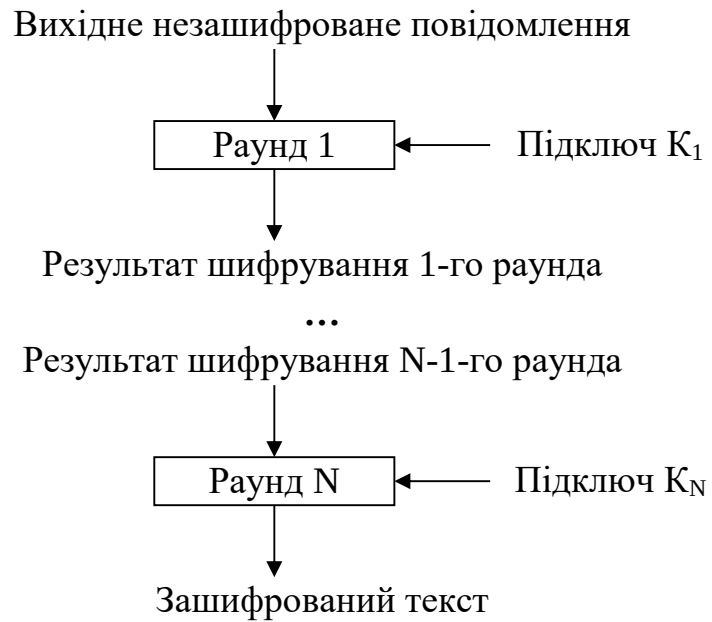


Рисунок 4.3 – Структура алгоритму алгоритмі Lucifer

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у бакалаврської дипломної роботі система гіперконвергентного хмарного рішення на основі технології NaaS. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Розділу обрання групи; Функціональних кнопок ПЗ; Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші; Верхнього меню.

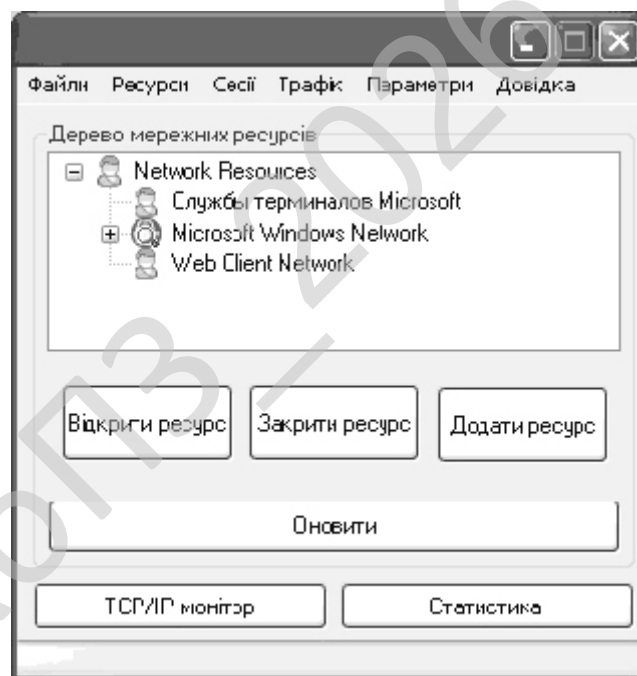


Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких

помилки, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

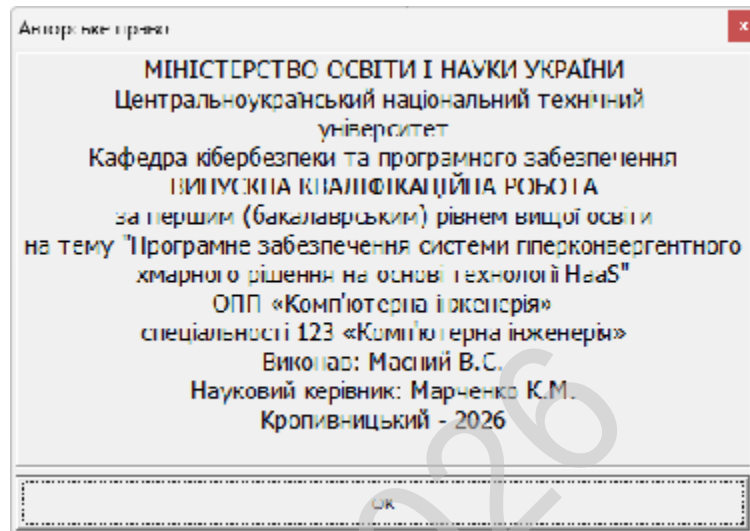


Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		56

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

- Випуск.
- Встановлення та активація.
- Деактивація.
- Адаптація.
- Оновлення.
- Вмонтування.
- Відстежування версій.
- Видалення.
- Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати ІТ рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене ІТ рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в ІТ рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		57

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

- відповідності поставленим вимогам;
- правильна відповідь для усіх можливих вхідних даних;
- виконання функцій за прийнятний час;
- практичність;
- сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

- Гарантується перевірка всіх незалежних маршрутів програми.
- Знаходяться гілки True, False для всіх логічних рішень.
- Виконуються всі цикли (у межах їхніх кордонів та діапазонів).
- Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

- Кількість незалежних маршрутів може бути дуже велика.
- Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.
- У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно.

Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ).

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		59

Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (zareestruvatisya), заплативши авторіві певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

К6ПЗ_2026

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		60

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи гіперконвергентного хмарного рішення на основі технології NaaS.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

- Був проведений огляд існуючих систем гіперконвергентного хмарного рішення на основі технології NaaS.
- Досліджена система гіперконвергентного хмарного рішення на основі технології NaaS.
- На основі отриманих результатів досліджень створена програмна реалізація системи гіперконвергентного хмарного рішення на основі технології NaaS.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання гіперконвергентного хмарного рішення на основі технології NaaS.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створенні програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи гіперконвергентного хмарного рішення на основі технології NaaS. Це

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		61

дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Lucifer.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		62

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Е. Таненбаум, Д. Уезеролл «Комп'ютерні мережі». – [5-е вид.]. – 2016. – 960 с.
2. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 1». Cisco Press. 2020. – 848 p.
3. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 2 Premium Edition eBook and Practice Test». Cisco Press. 2020. – 624 p.
4. Scott Jernigan «CompTIA Network+ Certification All-in-One Exam Guide, Eighth Edition». 2022. – 976 p.
5. Doug Lowe «Networking For Dummies 12th Edition». 2020. – 480 p.
6. Ramon Nastase «Computer Networking: The Beginner's guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.
7. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.
8. Вінтенко Б., Смірнов О., Миронець І., Смірнова Т., Смірнов С. «Імітаційна модель шляхів вхідних даних комп'ютерної інтелектуальної системи підтримки оператора енергоблоку АЕС». *Комбінаторні конфігурації та їхні застосування: Матеріали XXVII Міжнародного науково-практичного семінару, присвяченого 125-річчю Національного університету «Запорізька політехніка» (Запоріжжя-Кропивницький-Київ, 4-6 червня 2025 р.)*. Запоріжжя: НУ «Запорізька політехніка», 2025. С.82-91.
9. Al-Azzeh, J., Ayyoub, B., Mesleh, A., Smirnova, T., Gnatyuk, S., Drieiev, O., Smirnov, O., Dorenskyi, O. «Cloud-Based Information System for Evaluating Caverns in the Process of Blasting Metal Surfaces of Details». *International Review on Modelling and Simulations* 18 (1), 2025. pp. 32-42.
10. Смірнова Т.В., Коноплицька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		63

технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка*. 2024. №4(24), С. 6-27.

11. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології*, 2024, № 13, с. 28-35.

12. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

13. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

14. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchев, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings*, Volume 3530, 2023, pp. 256-265.

15. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

16. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи*, 2023, том 7, № 2, С. 49-56.

17. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». *CEUR Workshop Proceedings* Volume 3156, 2022, Pages 390-399.

18. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		64

телекомунікаційних систем». *Проблеми інформатизації та управління*, № 2(70). 2022. С. 28-37.

19. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 3(69). С. 93-98.

20. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»*, № 2 (307). С. 46-52. 2022.

21. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

22. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

23. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

24. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

					ВКРБ-123.26.0049.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		65

25. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 172-177.
26. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) *Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies*, vol 48. Springer, Cham. 2021. pp 557-587.
27. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings Volume 2616*, 2020, Pages 125-136.
28. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings Volume 2616*, 2020, Pages 366-379.
29. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings Volume 2608*, 2020, Pages 633-645.
30. Smirnov O. Kuznetsov A., Zaichenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.
31. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». *International Journal of Computing*; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

32. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*, Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

33. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*, Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

34. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

35. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv*, Ukraine, 2-6 July, 2019, P. 395-399.

36. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

37. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

38. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings Volume 2353, CEUR Workshop Proceedings 2019*, Pages 618-629.

39. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

40. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

41. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки*. №4. С. 103-110. 2020.

42. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

43. Смірнов О.А., Коноплицька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

44. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія*. – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

45. Смірнов О.А., Дреєва Г.М., Дреєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки*. № 2(33). с. 161-172, 2019.

46. Смірнов О.А., Коноплицька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних

систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

47. Smirnov, O., Kuznetsov, A., Kuznetsova, K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

48. Смірнов О.А., Дреєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

49. Дреєва Г.М., Смірнов О.А., Дреєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральнотраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

50. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дреєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральнотраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

51. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Коноплицька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

52. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

53. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

54. Смірнов О.А., Смірнов С.А., Дідик А.К., Дреєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". - Випуск 5 (142). - Х.: ХУПС - 2016. - С. 148-152.

55. Смірнов О.А., Смірнов С.А. Дідик А.К., Дреєв О.М. Моделі системи нейромережових експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

56. Смірнов О.А., Смірнов С.А., Дідик А.К., Дреєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. - 2016. - С. 121-127.

57. Смірнов О.А., Смірнов С.А., Дідик А.К. Метод безпечної маршрутизації метаданих у хмарні антивірусні системи. Системи озброєння та військова техніка. - Випуск 2 (46) - Х.: ХУПС - 2016. - С. 146-149.

58. Смірнов О.А., Кавун С.В., Доренський О.П., Вялкова В.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 151 с.