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 Software Define Network побудованих з використанням комутаторів Agema”

Виконав здобувач вищої освіти
IV курсу, групи КІ-22-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Гой Д.О.
«____» _____ 2026 р.

Керівник проекту
доктор філософії (PhD)
_____ Усік П.С.
«____» _____ 2026 р.
Рецензент _____
_____

м. Кропивницький

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
« 15 » січня 2026 року

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

Гою Дмитру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи Software Define Network побудованих з використанням комутаторів Agema*

2. Керівник роботи *Усік Павло Сергійович, доктор філософії (PhD)*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 133-02 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 Software Define Network побудованих з використанням комутаторів Agema*

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 « 15 » січня 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
« 15 » січня 2026 р.

Підпис керівника
_____ Усік П.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 15 » січня 2026 р.

Підпис здобувача
_____ Гой Д.О.
(прізвище та ініціали)

АНОТАЦІЯ

Гой Д.О. Програмне забезпечення системи Software Define Network побудованих з використанням комутаторів Agema. 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 Software Define Network побудованих з використанням комутаторів Agema.

Метою розробки є програмне забезпечення системи Software Define Network побудованих з використанням комутаторів Agema.

Результат роботи – програмна реалізація системи Software Define Network побудованих з використанням комутаторів Agema.

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, Software Define Network, Agema

ABSTRACT

Hoi D.O. Software for the Software Define Network system built using Agema switches. 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, which is intended for the Software Define Network system built using Agema switches.

The purpose of the development is the software for the Software Define Network system built using Agema switches.

The result of the work is the software implementation of the Software Define Network system built using Agema switches.

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, Software Defined Network, Agema

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	база даних
УЛММР	–	узагальнена лінійна модель множинної регресії
МНК	–	метод найменших квадратів
СУБД	–	система управління базою даних
УМНК	–	узагальнений метод найменших квадратів
ОПР	–	особа яка приймає рішення

ВСТУП

Актуальність теми. Загальна тенденція забезпечення незалежності вибору апаратного й програмного забезпечення поступово набирає силу й у мережевому сегменті IT-ринку. Це виражається в усі більше широкій пропозиції устаткування без передвстановленої ОС або з ОС на базі відкритого коду (bare-metal switch і white-box switch). Як продемонстрували проведені AT&T рік назад польові випробування, комутатори від різних виробників під керуванням однієї й тої ж мережевої операційної системи можуть цілком успішно справлятися з передачею трафіку. У тестуванні використовувалися в тому числі пристрої Agema AGC7648A компанії Delta Electronics.

Комутатори під маркою Agema виробництва Delta Electronics орієнтовані на підтримку програмно обумовлених мереж (Software Define Network, SDN). Вони можуть поставлятися як із передвстановленою операційною системою, так і без неї. В останньому випадку для інсталяції ОС підтримується програмне завантажувальне середовище (Open Network Install Environment, ONIE). Користувач за своїм вибором може встановити одну із трьох відкритих операційних систем Broadcom ICOS, IP Infusion OcNOS, Cumulus Linux. На деяких моделях також підтримується мережева ОС Pica8 PICOS.

Delta Electronics пропонує широку лінійку комутаторів для трьох цільових ринків: корпоративні мережі, центри обробки даних і сервіси-провайдери. Вони підтримують широкий діапазон швидкостей від 1 до 400 Gigabit Ethernet залежно від цільової області застосування. Так, комутатори для ЦОД підтримують швидкості 1, 10, 25, 40 і 100Gb. На відміну від традиційних комутаторів замість пропрієтарних ASIC у них використовуються загальнодоступні готові набори мікросхем. В устаткуванні Agema застосовуються різні ASIC StrataDNX і StrataXGS виробництва Broadcom (Qumran-MX, Tomahawk, Helix, Trident і ін).

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи Software Define Network побудованих з використанням комутаторів Agema.

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

– Огляд існуючих систем Software Define Network побудованих з використанням комутаторів Agema.

– Дослідження системи Software Define Network побудованих з використанням комутаторів Agema.

– Програмна реалізація системи Software Define Network побудованих з використанням комутаторів Agema.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі Software Define Network побудованих з використанням комутаторів Agema.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи Software Define Network побудованих з використанням комутаторів Agema, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Комутатори Agema пропонують високопродуктивну й у той же час доступну за ціною альтернативу традиційному мережевому устаткування. Так, комутатор AG7648 без передвстановленої ОС для центрів обробки даних призначений для установки в стійку (Top-of-Rack, ToR) або використання як периферійний вузол (leaf). Він оснащений 48 портами 10Gb SFP+ і 6 портами 40Gb QSFP. У нього можуть установлюватися різні процесори, чим забезпечується необхідна гнучкість для завантаження різних застосунків і підтримки необхідної пропускної здатності.

У березні Delta Electronics представила нові високопродуктивні комутатори з високою щільністю портів для центрів обробки даних і сервіс-провайдерів. Так, масштабований програмувальний комутатор AG9064v2 на базі комутуючих мікросем Broadcom StrataXGS ASIC має 64 порту 100 Gigabit Ethernet QSFP28 і забезпечує комутацію до 6,4 Тбіт/с. Він виконаний у форм-факторі 2U і призначений для використання як комутатор ядра у дворівневій топології leaf-spine. У найближчих планах компанії випуск комутатора 400Gb з об'єднавчою панеллю продуктивністю 12,8 Тбіт/с для використання в якості агрегатора інших spine-комутаторів.

Крім комутатор Delta Electronics випускає й інше устаткування, що відповідає принципам відкритих обчислень. Компанія анонсувала OCPv 2-сумісну стійку висотою 41OU для устаткування з живленням від постійного струму 48 У и відповідну систему електроживлення. Полку живлення висотою 2U з резервуванням N+1 розрахована на навантаження потужністю до 33 кВт, а резервна система електроживлення у вигляді полиці висотою 3U з 15 акумуляторними батареями по 2,7 кВт здатна забезпечувати подачу живлення

протягом 2 хв. Моделі комутаторів для ЦОД і сервіс-провайдерів можуть оснащуватися блоками живлення від постійного струму.

Комутатори whitebox традиційно розглядалися як устаткування з обмеженою функціональністю дешеве в придбання, але дороге в обслуговування. Через необхідність залучення кваліфікованих фахівців з Linux їхнє застосування обмежувалося компаніями зі значними технічними й програмисткими ресурсами. Однак, багато хто з функцій брендових комутаторів надлишкові для багатьох застосувань, а «білі коробки» можна придбати тепер і в іменитих виробників, які готові надати технічну підтримку. А такі операційні системи як Cumulus Linux дозволяють використовувати стандартні Linux-Застосунки для оркестрування, керування, налаштування й автоматизації функціонування.

Як стверджують експерти, ринок дозрів для широкого застосування комутаторів whitebox. Звичайно, вони підійдуть далеко не для всіх завдань, але тим, хто планує розгортання SDN, варто розглянути їх у першу чергу.

1.2 Область застосування

Програмно-визначаєма глобальна мережа (SD-WAN або SDWAN) являє собою спеціальний застосунок технології програмно-визначаємої мережі (SDN), застосовуваної до з'єднань WAN, таким як широкополосний Інтернет, 4G, LTE або MPLS. Він з'єднує корпоративні мережі, включаючи філії й центри обробки даних, на більших географічних відстанях. WAN може використовуватися, наприклад, для підключення філій до центральної корпоративної мережі або для підключення центрів обробки даних, розділених відстанню. У минулому для WAN-з'єднань часто використовувалася технологія, що вимагала спеціального пропрієтарного устаткування. SD-WAN, з іншого боку, використовує Інтернет або хмарні приватні мережі. SD-WAN відокремлює мережу від площини керування й від'єднує функції керування трафіком і апаратного забезпечення.

Мережі WAN дозволяють компаніям розширювати свої комп'ютерні мережі на більші відстані, з'єднувати віддалені філії із центрами обробки даних і один з одним, а також надавати застосунки й послуги, необхідні для виконання бізнес-функцій. Коли компанії розширюють мережі на більші відстані, а іноді й по мережах з декількома операторами, вони зіштовхуються з експлуатаційними проблемами, включаючи перевантаження мережі, зміна затримки пакетів, втрату пакетів і навіть перебої в обслуговуванні. Сучасні застосунки, такі як VoIP-виклики, відеоконференції, потокове мультимедіа, віртуалізовані застосунки й робочі столи, вимагають мінімальних затримок. Вимоги до пропускної здатності також ростуть, особливо для застосунків з відео високої чіткості. Розширення можливостей WAN може бути дорогим і скрутним з відповідними труднощами, пов'язаними з керуванням мережею й усуненням неполадок.

Продукти SD-WAN призначені для рішення цих мережевих проблем. Завдяки розширенню або навіть заміні традиційних галузевих маршрутизаторів пристроями віртуалізації, які можуть управляти політиками на рівні застосунків і забезпечувати накладення мережі, менш дорогі інтернет-канали споживчого рівня можуть діяти скоріше як виділений канал. Це спрощує процес налаштування для персоналу філії. Продукти SD-WAN можуть бути фізичними або віртуальними пристроями й розміщатися в невеликих віддалених офісах і філіях, великих офісах, корпоративних центрах обробки даних і всі частіше на хмарних платформах. Централізований контролер використовується для установки політик і визначення пріоритетів трафіку. SD-WAN ураховують ці політики й доступність пропускної здатності мережі для маршрутизації трафіку. Це допомагає забезпечити відповідність продуктивності застосунків угодам про рівень обслуговування (SLA).

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи Software Define Network побудованих з використанням комутаторів Agema, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

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

Послуги IBM для програмно-визначаємих мереж (SDN)

Послуги IBM для SDN допомагають корпоративним клієнтам створити програмувальне середовище комутації, що охоплює ЦОД/хмара (SDN-DC), глобальну мережу (SD-WAN) і локальні мережі (SD-LAN). IBM користується консультаційним підходом і допомагає створювати хмарні динамічні відказостійкі мережі, що відповідають майбутнім потребам організацій.

Переваги SDN

Інфраструктура, готова до майбутнього

Технологічні інновації розвиваються швидко. Гнучка масштабована IT-платформа допоможе вам іти в ногу згодом, а не просто намагатися погнатися. Мережі SDN мають необхідну гнучкість для ефективного використання новітніх хмарних ресурсів і є основою для підтримки застосунків, що працюють із великими обсягами даних у реальному часі.

Скорочення витрат на апаратне забезпечення й експлуатацію

Мережа SDN допомагає знизити витрати на апаратне забезпечення за рахунок виключення ручного налаштування й експлуатаційні витрати завдяки підвищенню ефективності адміністрування, керуванню віртуалізацією і підвищенню завантаження серверів. SDN також допомагає продовжити використання наявних мережевих пристроїв і апаратного забезпечення.

Централізоване керування мережею

Мережа SDN забезпечує централізоване керування всією мережею як єдиним цілим, підвищуючи прозорість і спрощуючи адміністрування.

Централізація керування дозволяє підвищити рівень захисту мережі за рахунок погодженого поширення інформації про безпеку й політиків по всьому підприємству.

SD-WAN

Динамічно управляйте трафіком WAN, переданим через різні з'єднання MPLS і Інтернету, різних операторів і різні регіони.

Integrated Managed Infrastructure (IMI) для SDN

Що набудовуються, модульні й каталожні послуги в області моніторингу віддалених мереж, керування й підготовки звітів для підтримки середовищ SDN.

IBM Client Innovation Centers

Створюйте, інтегруйте й тестуйте рішення SDN перед впровадженням у виробничому середовищі.

SDN для ЦОД

Проектування, розгортання й керування мережами ЦОД на основі технологій SDN, віртуалізації мереж і мережевих функцій.

Огляд SDN в Windows Server

Відноситься до: Windows Server (Semi-Annual Channel), Windows Server 2025.

Програмно-визначаєма мережа (SDN) дозволяє централізовано налаштовувати й контролювати фізичні й віртуальні мережеві пристрої, наприклад маршрутизатори, комутатори й шлюзи в центрі обробки даних. Ви можете використовувати існуючі пристрої, сумісні з SDN, щоб забезпечити більше глибоку інтеграцію між віртуальною мережею й фізичною мережею. Такі елементи віртуальної мережі, як віртуальний комутатор Hyper-V, віртуалізація мережі Hyper-V і шлюз RAS, є невід'ємною частиною інфраструктури SDN.

На вузлах Hyper-V і віртуальних машинах під керуванням серверів інфраструктури SDN, таких як вузли мережевого контролера й програмного балансування навантаження, повинен бути встановлений Windows Server 2025 Datacenter Edition.

Вузли Hyper-V, що містять тільки віртуальні машини робочого навантаження клієнта, підключені до мереж, керованим SDN, можуть використовувати Windows Server 2025 Standard Edition.

Це можливо, оскільки мережеві площини більше не прив'язані до мережевих пристроїв. Однак інші сутності, такі як програмне забезпечення для керування центром обробки даних, наприклад System Center 2025, використовують мережеві площини. SDN дозволяє динамічно управляти мережею центра обробки даних, надаючи автоматизований, централізований спосіб задоволення вимог застосунків і робочих навантажень.

SDN можна використовувати для:

– Динамічно створюйте, Забезпечуйте безпеку й підключайте мережу для задоволення зростаючих потреб ваших застосунків.

– Прискорене розгортання робочих навантажень без порушення роботи.

– Захист від поширення уразливостей по мережі.

– Визначення й керування політиками, що контролюють як фізичні, так і віртуальні мережі.

– Однакова реалізація мережевих політик у масштабі.

SDN дозволяє виконувати всі ці операції, одночасно зменшуючи загальні витрати на інфраструктуру.

Планування інфраструктури програмно-конфігуруємої мережі

Попередні вимоги

У цьому розділі описується ряд попередніх вимог до устаткування й програмного забезпечення, включаючи:

– **Налаштовані групи безпеки, розташування файлів журналів і динамічна реєстрація DNS.** Необхідно підготувати центр обробки даних для розгортання мережевого контролера, для чого потрібно один або кілька комп'ютерів або віртуальних машин, а також один комп'ютер або віртуальна машина. Перш ніж можна буде розгорнути мережевий контролер, необхідно налаштувати групи безпеки, розташування файлів журналу (при необхідності) і

динамічну реєстрацію DNS. Якщо ви не підготували центр обробки даних для розгортання мережевого контролера, див. Додаткові відомості про вимоги до установки й підготовки для розгортання мережевого контролера.

– **Фізична мережа**. Вам буде потрібно доступ до фізичних мережевих пристроїв, щоб налаштувати віртуальні ЛМ, маршрутизацію, BGP, мости центрів обробки даних (ETS) при використанні технології RDMA і мостового центра обробки даних (коефіцієнт потужності) при використанні технології RDMA на основі Роце. У цьому розділі показано, як налаштувати комутатор вручну, так і пірінг BGP на комутаторах і маршрутизаторах рівня 3 або на віртуальній машині сервера маршрутизації й віддаленого доступу (RRAS).

– **Фізичні вузли обчислень**. Ці вузли працюють під керуванням Hyper-V і необхідні для розміщення інфраструктури SDN і віртуальних машин клієнта.

Конфігурація фізичної мережі й вузла обчислень

Кожному фізичному вузлу обчислень потрібне мережеве підключення через один або кілька мережевих адаптерів, підключених до портів фізичного комутатора. Віртуальна ЛМ рівня 2 підтримує мережі, розділені на кілька логічних сегментів мережі.

Використовуйте віртуальну ЛМ 0 для логічних мереж у режимі доступу або без тегів.

Логічні мережі

Постачальник керування й HNV

Всі фізичні вузли обчислень повинні одержати доступ до логічної мережі керування й логічною мережею постачальника HNV. Для цілей планування IP-адрес кожний фізичний вузол обчислень повинен мати принаймні одну IP-адресу, призначену в логічній мережі керування. Мережевому контролеру потрібно зарезервована IP-адреса, що буде використовуватися в якості IP-адреси для інших компонентів.

DHCP-сервер може автоматично призначати IP-адреси для мережі керування. також можна призначити статичний IP-адресу вручну. Стік SDN

автоматично призначає IP-адреси для логічних мереж постачальника HNV для окремих вузлів Hyper-V з пула IP-адрес, зазначеного через мережевий контролер і управляє їм.

Мережевий контролер призначає IP-адресу постачальника HNV фізичному вузлу обчислень тільки після того, як агент вузла мережевого контролера одержує політику мережі для конкретної віртуальної машини клієнта.

Шлюзи й програмне забезпечення Load Balancer

Необхідно створити й підготувати додаткові логічні мережі для шлюзу й використання SLB. Обов'язково одержите правильні IP-префікси, ідентифікатори віртуальних ЛМ і IP-адреси шлюзу для цих мереж.

Вимоги до розгортання мережевого контролера

Підготуйте центр обробки даних до розгортання мережевого контролера, для чого потрібно один або кілька комп'ютерів або віртуальних машин, а також один комп'ютер або віртуальна машина. Перш ніж можна буде розгорнути мережевий контролер, необхідно налаштувати групи безпеки, розташування файлів журналу (при необхідності) і динамічну реєстрацію DNS.

Вимоги до мережевого контролера

Для розгортання мережевого контролера потрібно один або кілька комп'ютерів або віртуальних машин, які служать мережевим контролером, а один комп'ютер або віртуальна машина – як клієнт керування для мережевого контролера:

– Всі віртуальні машини й комп'ютери, заплановані як вузли мережевого контролера, повинні працювати під керуванням Windows Server 2025 Datacenter Edition.

– На будь-якому комп'ютері або віртуальній машині, на якій установлюється мережевий контролер, повинен працювати випуск Datacenter операційної системи Windows Server 2025.

– Клієнтський комп'ютер керування або віртуальна машина для мережевого контролера повинні працювати під керуванням Windows 10.

Вимоги до конфігурації

Перед розгортанням мережевого контролера необхідно налаштувати групи безпеки, розташування файлів журналу (при необхідності) і динамічну реєстрацію DNS.

Крок 1. Налаштування груп безпеки

Перше, що потрібно зробити, – створити дві групи безпеки для перевірки дійсності Kerberos.

Ви створюєте групи для користувачів, що мають дозвіл на:

– Налаштування мережевого контролера. Ви можете привласнити цьому групі адміністраторів мережевого контролера, наприклад.

– Налаштування мережі й керування нею за допомогою мережевого контролера. Можна привласнити ім'я цій групі користувачів мережевого контролера, наприклад. Для налаштування мережевого контролера й керування їм використовуйте пересилання даних про стан (інше).

Всі користувачі, які додаються, повинні бути членами групи "Користувачі домена" в Active Directory "користувачі й комп'ютери".

Крок 2. Налаштування розташування файлів журналів при необхідності

Далі необхідно налаштувати розташування файлів для зберігання журналів налагодження мережевого контролера на комп'ютері мережевого контролера або віртуальній машині або на віддаленому файловому ресурсі.

Якщо журнали зберігаються у віддаленому файловому ресурсі, переконайтеся, що загальна папка доступна з мережевого контролера.

Крок 3. Налаштування динамічної реєстрації DNS для мережевого контролера

Нарешті, далі потрібно розгорнути вузли кластера мережевого контролера в тої ж підмережі або в різних підмережах.

– Членство в групах **"Адміністратори домена" або "**еквівалентне" є мінімальною вимогою для виконання цих процедур.

1. Дозволити динамічне відновлення DNS для зони.

– Відкрийте диспетчер DNS і в дереві консолі клацніть правою кнопкою миші відповідну зону й виберіть пункт **Властивості**.

– На вкладці **Загальні** переконайтеся, що зона належить до типу **первична** або **Active Directory-Integrated**.

– У меню **динамічні відновлення** переконайтеся, що обрано параметр **тільки захист**, а потім натисніть кнопку **ОК**.

2. Налаштування дозволів безпеки зони DNS для вузлів мережевого контролера

– Відкрийте вкладку **Безпека** й виберіть **Додатково**.

– У вікні **Додаткові параметри безпеки** натисніть кнопку **Додати**.

– Клацніть **Вибрати суб'єкт**.

– У діалоговому вікні **Вибір користувача, комп'ютера, облікового запису служби або групи** натисніть кнопку **типи об'єктів**.

– У списку **типи об'єктів** виберіть **Комп'ютери**, а потім натисніть кнопку **ОК**.

– У діалоговому вікні **Вибір користувача, комп'ютера, облікового запису служби або групи** введіть NetBIOS-Ім'я одного з вузлів мережевого контролера в розгортанні, а потім натисніть кнопку **ОК**.

– У **поле запис дозволу** перевіте наступні значення.

- **Тип** = дозволити
- **Застосовно до** = цей об'єкт і всі дочірні об'єкти

– У списку **дозволи** виберіть **записати всі властивості** й **Видалити**, а потім натисніть кнопку **ОК**.

3. Повторіть ці дії для всіх комп'ютерів і віртуальних машин у кластері мережевого контролера.

Крок 4. Налаштування імені суб'єкта-служби при використанні перевірки дійсності на основі Kerberos

Якщо мережевий контролер використовує перевірку дійсності на основі Kerberos для взаємодії із клієнтами керування, необхідно налаштувати ім'я учасника-служби (SPN) для мережевого контролера в Active Directory. Мережевий контролер автоматично набудовує ім'я суб'єкта-служби. Усе, що потрібно зробити, – надати дозвіл на комп'ютери мережевого контролера для реєстрації й зміни ім'я учасника-служби.

Варіанти розгортання

Розгортання мережевого контролера

Налаштування забезпечує високу доступність із трьома вузлами мережевого контролера, налаштованими на віртуальних машинах. Також показані два клієнти з віртуальною мережею клієнта 2, які розбиваються на дві віртуальні підмережі для імітації веб-рівня й рівня бази даних.

Розгортання мережевого контролера й програмної підсистеми балансування навантаження

Для високого доступність існує два або більше вузли SLB/Мультиплексора.

Мережевий контролер, програмне Load Balancer і розгортання шлюзу RAS

Існує три віртуальних машини шлюзу. два є активними, а одна – надлишкової.

Для автоматизації розгортання на основі TP5 Active Directory повинні бути доступні для доступу із цих підмереж.

Розгортання програмно-визначаємой мережевої інфраструктури

Ці розгортання включають всі необхідні технології для повністю функціональної інфраструктури, включаючи віртуалізацію мережі Hyper-V (HNV), мережеві контролери, підсистеми балансування навантаження програмного забезпечення (SLB і МУЛЬТИПЛЕКСОР) і шлюзи.

Розгортання й контроль інфраструктури програмно-визначаємоїмережі (SDN) у структурі VMM

System Center Virtual Machine Manager (VMM) можна використовувати для розгортання й адміністрування інфраструктури програмно-конфігуруємої мережі (SDN).

Загальні відомості про програмно обумовлену мережу

Програмно-визначаєма мережа (SDN) віртуалізує мережу для абстрагування фізичного устаткування мережі, такого як комутатори й маршрутизатори. За допомогою SDN можна динамічно управляти мережею центра обробки даних відповідно до вимог робочих навантажень і застосунку. Це забезпечує масштабованість і погодженість застосування політик мережі навіть при розгортанні нових робочих навантажень і переміщенні робочих навантажень у віртуальних або фізичних мережах.

Розгортання SDN у структурі VMM дозволить вам вирішувати наступні завдання.

– Підготовка й керування віртуальними мережами при зміні масштабів середовища.

– Розгортання й керування інфраструктурою SDN, включаючи мережеві контролери, програмні підсистеми балансування навантаження й шлюзи.

– Визначення й централізоване керування політиками віртуальної мережі й зв'язування їх з додатками й робочими навантаженнями. При розгортанні або переміщенні робочого навантаження конфігурація мережі буде коректуватися автоматично. Це важливо, тому що рятує від необхідності ручного переналаштування мережевого устаткування, що знижує операційну складність при збереженні коштовних ресурсів для більше ефективної роботи.

– Керування потоком трафіку між віртуальними мережами, включаючи можливість визначення гарантованої пропускної здатності для критично важливих застосунків і робочих навантажень.

Мережа SDN сполучить кілька технологій, включаючи перераховані нижче.

– **Мережевий контролер**. Мережевий контролер дозволяє автоматизувати налаштування мережевої інфраструктури замість ручного налаштування мережевих пристроїв і служб.

– **Шлюз RAS для SDN**. Шлюз RAS – це програмний мультитенантний маршрутизатор, що підтримує функції BGP, у середовищі Windows Server 2025, призначений для постачальників хмарних рішень і підприємств, у середовищах яких розміщаються мультитенантні віртуальні мережі на основі технології HNV.

– **Програмна підсистема балансування навантаження (SLB) для SDN**. Мережа SDN в Windows Server 2025 може використовувати програмну підсистему балансування навантаження для рівномірного розподілу мережевого трафіку клієнта й замовників клієнта між ресурсами віртуальної мережі.Підсистема SLB в Windows Server дозволяє використовувати кілька серверів для розміщення одного робочого навантаження, забезпечуючи високу доступність і масштабованість.

Додаткові відомості про технології в стеці SDN.

Подальші дії

– Розгортання компонентів SDN за допомогою PowerShell.

– Розгортання компонентів SDN вручну в консолі VMM.

– Налаштування мережевого контролера.

– Налаштування програмної підсистеми балансування навантаження.

– Налаштування шлюзу RAS.

Розгортання інфраструктури програмно-конфігуруємої мережі за допомогою скриптів

У цьому розділі описується розгортання інфраструктури програмно-визначаємої мережі (SDN) за допомогою сценаріїв. Інфраструктура включає мережевий контролер високої доступності (HA), високодоступне програмне забезпечення Load Balancer (SLB)/МУКС, віртуальні мережі й зв'язані списки

керування доступом (ACL). Крім того, інший сценарій розгортає робоче навантаження клієнта для перевірки інфраструктури SDN.

Якщо ви хочете, щоб робочі навантаження клієнта взаємодіяли за межами своїх віртуальних мереж, можна налаштувати правила NAT SLB, тунелі шлюзу "мережа – мережа" або перенапрямок рівня 3 для маршрутизації між віртуальними й фізичними робочими навантаженнями.

Можна також розгорнути інфраструктуру SDN за допомогою Virtual Machine Manager (VMM). Додаткові відомості див. у розділі Налаштування інфраструктури програмно-визначаємої мережі (SDN) у структурі VMM.

Перед розгортанням

Перед початком розгортання необхідно спланувати й налаштувати вузли й інфраструктуру фізичної мережі.

На всіх вузлах Hyper-V повинен бути встановлений Windows Server 2025.

Кроки розгортання

Почніть із налаштування віртуального комутатора (фізичних серверів) вузла Hyper-v і призначення IP-адрес. Можна використовувати будь-який тип сховища, сумісний з Hyper-V, Shared або local.

Установка мережі вузла

1. Установіть останні версії мережевих драйверів, доступних для мережевого устаткування.

2. Установіть роль Hyper-V на всіх вузлах.

PowerShell Копіювати

```
Install-WindowsFeature -Name Hyper-V -ComputerName <computer_name> -IncludeManagementTools -Restart
```

3. Створіть віртуальний комутатор Hyper-V.

– Використовуйте те саме ім'я комутатора для всіх вузлів, наприклад **сднсвитч**. Настройте принаймні один мережевий адаптер або, якщо використовується набір, настройте принаймні два мережевих адаптери. Максимальна кількість вхідних підключень відбувається при використанні двох мережевих інтерфейсів.

PowerShell Копіювати

```
New-VMSwitch "<switch name>" -NetAdapterName "<NetAdapter1>" [, "<NetAdapter2>" -EnableEmbeddedTeaming $True] -AllowManagementOS $True
```

Ви можете пропустити кроки 4 і 5, якщо у вас є окремі мережеві адаптери керування.

4. Звернетеся до розділу планування (планування програмно-визначаємої мережевої інфраструктури) і звернетеся до адміністратора мережі для одержання ідентифікатора віртуальної ЛМ віртуальної ЛМ керування. Підключіть vNIC керування створеного віртуального комутатора до віртуального ЛМ керування. Цей крок можна опустити, якщо в середовищі не використовуються теги VLAN.

PowerShell Копіювати

```
Set-VMNetworkAdapterIsolation -ManagementOS -IsolationMode Vlan -DefaultIsolationID <Management VLAN> -AllowUntaggedTraffic $True
```

5. Ознайомтеся з розділом планування (планування програмно-обумовленої мережевої інфраструктури) і звернетеся до адміністратора мережі для призначення IP-адреси vNICу керування тільки що створеної vSwitch. У наступному прикладі показано, як створити статичний IP-адреса й призначити його vNIC керування vSwitch:

PowerShell Копіювати

```
New-NetIPAddress -InterfaceAlias "vEthernet (<switch name>)" -IPAddress <IP> -DefaultGateway <Gateway IP> -AddressFamily IPv4 -PrefixLength <Length of Subnet Mask – for example: 24>
```

6. Використовуваних Розгорніть віртуальну машину для розміщення служб домен Active Directory (установіть служби домен Active Directory Services (рівень 100) і DNS-сервер.

1. Підключіть віртуальну машину Active Directory або DNS-сервера до віртуального ЛМ керування:

Копіювати

```
'''PowerShell
Set-VMNetworkAdapterIsolation -VMName "<VM Name>" -Access -VlanId <Management VLAN> -AllowUntaggedTraffic $True
'''
```

2. Установіть служби домен Active Directory і DNS.

Мережевий контролер підтримує сертифікати Kerberos і X. 509 для перевірки дійсності. У цьому керівництвом використовуються обидва механізми перевірки дійсності для різних цілей (хоча потрібно тільки один).

7. Приєднаєте всі вузли Hyper-V до домену. Переконайтеся, що запис DNS-сервера для мережевого адаптера з IP-адресою, призначеною мережі керування, указує на DNS-сервер, що може дозволити доменне ім'я.

PowerShell Копіювати

```
Set-DnsClientServerAddress -InterfaceAlias "vEthernet (<switch name>)" -ServerAddresses <DNS Server IP>
```

– Клацніть правою кнопкою миші **Пуск**, виберіть пункт **система**, а потім клацніть **змінити параметри**.

– Натисніть кнопку **Змінити**.

– Клацніть **домен** і вкажіть ім'я домена.

– Натисніть кнопку **ОК**.

– З появою запиту введіть ім'я користувача й пароль.

– Перезавантажите сервер.

Перевірка

Виконаєте наступні дії, щоб перевірити правильність налаштування мережі вузла.

1. Переконайтеся, що комутатор віртуальної машини успішно створений:

PowerShell Копіювати

```
Get-VMSwitch "<switch name>"
```

2. Переконайтеся, що vNIC керування на комутаторі віртуальної машини підключений до віртуального ЛМ керування:

Доречно, тільки якщо трафік керування й клієнта має загальну мережеву карту.

PowerShell Копіювати

```
Get-VMNetworkAdapterIsolation -ManagementOS
```

3. Перевірте всі вузли Hyper-V і зовнішні ресурси керування, наприклад DNS-сервери.

Переконайтеся, що вони доступні через перевірку зв'язку, використовуючи свій IP-адресу керування й (або) повне доменне ім'я (FQDN).

```
ping < Hyper-V Host IP>
ping < Hyper-V Host FQDN>
```

4. Виконаєте наступну команду на вузлі розгортання й укажіть повне доменне ім'я кожного вузла Hyper-V, щоб облікові дані Kerberos використовувалися для доступу до всіх серверів.

```
winrm id -r:< Hyper-V Host FQDN>
```

Вимоги й примітки для установки Nano

При використанні Nano як вузли Hyper-V (фізичних серверів) для розгортання нижче наведені додаткові вимоги.

1. На всіх вузлах Nano повинен бути встановлений пакет DSC з язиковим пакетом:

– Наносервер-дск-паккаже. cab

– NanoServer-DSC-Package_ en-us. cab

```
dism /online / add-package /packagepath:<Path> /loglevel:4
```

2. Сценарії SDN Express необхідно запускати з вузла, що не є вузлом Nano (Windows Server Core або Windows Server із графічним інтерфейсом користувача). Робочі процеси PowerShell не підтримуються в Nano.

3. Виклик API мережевого контролера обміну за допомогою оболонок викликів PowerShell або NC (які ґрунтуються на Invoke-WebRequest і Invoke-RestMethod) необхідно виконувати на вузлі, що не є вузлом Nano.

Виконання скриптів SDN Express

1. Перейдіть у репозиторій Microsoft Sdn GitHub для файлів установки.

– Скачайте файли установки з репозиторія на зазначений комп'ютер розгортання. Клацніть **клонувати або скачати**, а потім клацніть **скачати ZIP-файл**.

Зазначений комп'ютер розгортання повинен працювати під Windows Server 2025 або більше пізньої версії.

– Розгорніть ZIP-файл і скопіюйте папку **SDNExpress** у папку:\ комп'ютера розгортання.

– Надайте загальний доступ до папки `C:\SDNExpress` у вигляді "**SDNExpress**" з дозволом на **читання й запису** для **всіх користувачів**.

2. Перейдіть у папку `C:\SDNExpress`.

– Переконайтеся, що VHDX-файл Windows Server 2025 перебуває в папці **Images**.

– Настройте файл SDNExpress\scripts\FabricConfig.psd1, змінивши**< < замініте > теги >** на конкретні значення відповідно до інфраструктури лабораторії, включаючи імена вузлів, імена доменів, ім'я користувача й пароль, а також відомості про мережу для мережі, перераховані в розділі Планування мережі.

6. Створіть запис вузла A в DNS для Networkcontrollerrestname (FQDN) і Networkcontrollerretype.

7. Запустите сценарій від імені користувача з обліковими даними адміністратора домена:

```
SDNExpress\scripts\SDNExpress.ps1 -ConfigurationDataFile FabricConfig.psd1 -Verbose
```

8. Щоб скасувати всі операції, виконаєте наступну команду:

```
SDNExpress\scripts\SDNExpressUndo.ps1 -ConfigurationDataFile FabricConfig.psd1 -Verbose
```

Перевірка

Припускаючи, що сценарій SDN Express завершив роботу без повідомлення про помилки, можна виконати наступні дії, щоб переконатися, що ресурси структури розгорнуті правильно й доступні для розгортання клієнта.

Використовуйте засоби діагностики, щоб переконатися у відсутності помилок у ресурсах структури в мережевому контролері.

```
Debug-NetworkControllerConfigurationState -NetworkController <FQDN of Network Controller Rest Name>
```

Розгортання приклада робочого навантаження клієнта за допомогою програмної підсистеми балансування навантаження

Тепер, коли ресурси структури розгорнуті, можна перевірити комплексне розгортання SDN, розгорнувши приклад робочого навантаження клієнта. Це Робоче навантаження клієнта складається із двох віртуальних підмереж (веб-рівень і рівень бази даних), захищених за допомогою правил списку керування доступом (ACL) з використанням розподіленого брандмауера SDN. Віртуальну підмережу веб-рівня можна одержати за допомогою SLB або Мультиплексора, використовуючи віртуальний IP-адресу (VIP).Сценарій автоматично розгортає дві віртуальні машини веб-рівня й одну віртуальну машину рівня бази даних і підключає їх до віртуальних підмереж.

– Настройте файл SDNExpress\scripts\TenantConfig.psd1, змінивши< < **заміните > теги >** заданими значеннями (наприклад: Ім'я образа VHD, ім'я мережі мережевого контролера, ім'я vSwitch і т.д., як раніше було визначено у файлі Фабрикконфиг. PSD1

1. Запустите скрипт. Приклад:

```
SDNExpress\scripts\SDNExpressTenant.ps1 -ConfigurationDataFile TenantConfig.psd1 -Verbose
```

– Щоб скасувати налаштування, виконаєте той же сценарій з параметром **Undo**. Приклад:

```
SDNExpress\scripts\SDNExpressTenant.ps1 -Undo -ConfigurationDataFile TenantConfig.psd1 -Verbose
```

Перевірка

Щоб перевірити успішність розгортання клієнта, виконаєте наступні дії.

1. Увійдіть на віртуальну машину рівня бази даних і спробуйте перевірити зв'язок з IP-адресою однієї з віртуальних машин веб-рівня (переконайтеся, що брандмауер Windows відключений на віртуальних машинах веб-рівня).

2. Перевірте ресурси клієнта мережевого контролера на наявність помилок. Виконаєте наступну команду з будь-якого вузла Hyper-V з підключенням рівня 3 до мережевого контролера:

```
Debug-NetworkControllerConfigurationState -NetworkController <FQDN of Network Controller REST Name>
```

3. Щоб переконатися, що балансувальник навантаження працює правильно, виконаєте наступну команду з будь-якого вузла Hyper-V:

```
wget <VIP IP address>/unique.htm -disablekeepalive -usebasicparsing
```

де `<VIP IP address>` – IP-адреса віртуальної IP-адреси рівня, налаштована у файлі Тенантконфиг. PSD1.

Знайдіть змінну `VIPIP` у Тенантконфиг. PSD1.

Запустите цей паралельно раз, щоб побачити комутатор підсистеми балансування навантаження між доступними DIP. Це поводження можна також переглянути за допомогою веб-браузера. Перейдіть за адресою `<VIP IP address>/unique.htm`. Закрийте Браузер (і відкрийте новий екземпляр і повторіть огляд. Ви побачите синю сторінку й альтернативу зеленою сторінкою, за винятком випадків, коли браузер кешує сторінку до витікання часу очікування кеша.

Розгортання мережевого контролера за допомогою Windows PowerShell

У цьому розділі наведені інструкції з використання Windows PowerShell для розгортання мережевого контролера на одній або декількох віртуальних машинах, що працюють під Windows Server 2025.

– Не розгортайте роль сервера мережевого контролера на фізичних вузлах. Щоб розгорнути мережевий контролер, необхідно встановити роль сервера мережевого контролера на віртуальній машині Hyper-V (VM @ no__ t-1, установленої на вузлі Hyper-V. Після установки мережевого контролера на віртуальних машинах на трьох різних вузлах Hyper @ no__ t-0V необхідно включити вузли Hyper @ no__ t-1V для програмно певних мереж (SDN @ no__ t-3, додавши вузли до мережевого контролера за допомогою Windows PowerShell. Команда New-**Нетворкконтроллерсервер**. Таким чином, ви включаєте програмне забезпечення SDN Load Balancer для роботи.

Установка ролі сервера мережевого контролера

Цю процедуру можна використовувати для установки ролі сервера мережевого контролера на віртуальній машині (VM @ no__ t-1.

Не розгортайте роль сервера мережевого контролера на фізичних вузлах. Щоб розгорнути мережевий контролер, необхідно встановити роль сервера мережевого контролера на віртуальній машині Hyper-V (VM @ no__ t-1, установленої на вузлі Hyper-V. Після установки мережевого контролера на віртуальних машинах на трьох різних вузлах Hyper @ no__ t-0V необхідно включити вузли Hyper @ no__ t-1V для програмно певних мереж (SDN @ no__ t-3, додавши вузли до мережевого контролера. Таким чином, ви включаєте програмне забезпечення SDN Load Balancer для роботи.

– Для виконання цієї процедури необхідно бути членом групи **Адміністратори** або користувачем з аналогічними правами.

Якщо для установки мережевого контролера ви хочете використовувати диспетчер сервера замість Windows PowerShell, див. статтю **Установка ролі сервера мережевого контролера за допомогою Диспетчер сервера**

Щоб установити мережевий контролер за допомогою Windows PowerShell, уведіть у командному рядку Windows PowerShell наступні команди й натисніть клавішу УВЕДЕННЯ.

```
Install-WindowsFeature -Name NetworkController -IncludeManagementTools
```

Для установки мережевого контролера потрібно перезавантажити комп'ютер. Для цього введіть наступну команду й натисніть клавішу УВЕДЕННЯ.

```
Restart-Computer
```

Налаштування кластера мережевого контролера

Кластер мережевого контролера забезпечує високий рівень доступності й масштабованість для застосунку мережевого контролера, яке можна налаштувати після створення кластера, а також розмістити його на вершині кластера.

– Ви можете виконувати процедури, описані в наступних розділах, або безпосередньо на віртуальній машині, де встановлений мережевий контролер, або

за допомогою засобу віддаленого адміністрування сервера для Windows Server 2025, щоб виконувати процедури з віддаленого комп'ютера, на якому працює Windows Server 2025 або Windows 10.Крім того, для виконання цієї процедури потрібне членство в групах «**Адміністратори**» або «еквівалентні».Якщо комп'ютер або віртуальна машина, на якій установлений мережевий контролер, приєднана до домену, обліковий запис користувача повинна бути членом **домена користувачів**.

Кластер мережевого контролера можна створити, створивши об'єкт node, а потім настроївши кластер.

Створення об'єкта node

Необхідно створити об'єкт node для кожної віртуальної машини, що є членом кластера мережевого контролера.

Щоб створити об'єкт node, уведіть у командному рядку Windows PowerShell наступну команду й натисніть клавішу УВЕДЕННЯ. Переконайтеся, що ви додали значення для кожного параметра, що підходить для вашого розгортання.

Копіювати

```
New-NetworkControllerNodeObject -Name <string> -Server <String> -FaultDomain <string>-RestInterface <string> [-NodeCertificate <X509Certificate2>]
```

Налаштування кластера

Щоб налаштувати кластер, уведіть у командному рядку Windows PowerShell наступну команду й натисніть клавішу УВЕДЕННЯ. Переконайтеся, що ви додали значення для кожного параметра, що підходить для вашого розгортання.

Копіювати

```
Install-NetworkControllerCluster -Node <NetworkControllerNode[]> -ClusterAuthentication <ClusterAuthentication> [-ManagementSecurityGroup <string>][-DiagnosticLogLocation <string>][-LogLocationCredential <PSCredential>] [-CredentialEncryptionCertificate <X509Certificate2>][-Credential <PSCredential>][-CertificateThumbprint <String>] [-UseSSL][-ComputerName <string>][-LogSizeLimitInMBs<UInt32>] [-LogTimeLimitInDays<UInt32>]
```

Налаштування застосунку мережевого контролера

Щоб налаштувати застосунок мережевого контролера, уведіть у командному рядку Windows PowerShell наступну команду й натисніть клавішу УВЕДЕННЯ. Переконайтеся, що ви додали значення для кожного параметра, що підходить для вашого розгортання.

Копіювати

```
Install-NetworkController -Node <NetworkControllerNode[]> -ClientAuthentication <ClientAuthentication> [-ClientCertificateThumbprint <string[]>] [-ClientSecurityGroup <string>] -ServerCertificate <X509Certificate2> [-RESTIPAddress <String>] [-RESTName <String>] [-Credential <PSCredential>][-CertificateThumbprint <String> ] [-UseSSL]
```

Після завершення налаштування застосунку мережевого контролера розгортання мережевого контролера буде завершено.

Перевірка розгортання мережевого контролера

Щоб перевірити розгортання мережевого контролера, можна додати облікові дані до мережевого контролера, а потім одержати облікові дані.

– Якщо ви використовуєте Kerberos як механізм EKU, для виконання цієї процедури потрібне членство в створеному **клиентсекуритигрaуп**.

– **PROCEDURE**

– При використанні Kerberos як механізм EKU на клієнтському комп'ютері ввійдіть у систему, використовуючи обліковий запис користувача, що є членом вашій **клиентсекуритигрaуп**.

1. Відкрийте Windows PowerShell, уведіть наступні команди, щоб додати облікові дані до мережевого контролера, і натисніть клавішу УВЕДЕННЯ. Переконайтеся, що ви додали значення для кожного параметра, що підходить для вашого розгортання.

Копіювати

```
$cred= New-Object Microsoft.Windows.Networkcontroller.credentialproperties
$cred.type="usernamepassword"
$cred.username="admin"
$cred.value="abcd"
```

```
New-NetworkControllerCredential -ConnectionUri https://networkcontroller -Properties $cred -ResourceId cred1
```

2. Щоб одержати облікові дані, додані в мережевий контролер, уведіть наступну команду й натисніть клавішу УВЕДЕННЯ. Переконайтеся, що ви додали значення для кожного параметра, що підходить для вашого розгортання.

Копіювати

```
Get-NetworkControllerCredential -ConnectionUri https://networkcontroller -ResourceId cred1
```

3. Перевірте вихідні дані команди, які повинні бути схожі на вихідні дані в наступному прикладі.

Копіювати

```
Tags                 :
ResourceRef    : /credentials/cred1
CreatedTime  : 1/1/0001 12:00:00 AM
InstanceId      : e16ffe 62-a701-4d 31-915e-7234d4bc5a18
Etag                : W/"1ec 59631-607f-4d3 e-ac78-94b0822f3a9d"
ResourceMetadata:
ResourceId     : cred1
Properties     : Microsoft.Windows.NetworkController.CredentialProperties
```

– При виконанні команди Get-**Нетворкконтроллеркредентиал** можна призначити вихідні дані команди змінної за допомогою оператора Dot, щоб одержати список властивостей облікових даних. Наприклад, $cred. Властивості.

Додаткові команди Windows PowerShell для мережевого контролера

Після розгортання мережевого контролера можна використовувати команди Windows PowerShell для керування розгортанням і його зміни. Нижче наведені деякі зміни, які можна внести в розгортання.

– Зміна параметрів вузла, кластера й застосунку мережевого контролера

– Видалення кластера й застосунку мережевого контролера

– Керування вузлами кластера мережевого контролера, включаючи додавання, видалення, включення й відключення вузлів.

Команди Windows PowerShell для мережевого контролера перебувають у бібліотеці TechNet у **командлетах мережевого контролера**.

Приклад сценарію конфігурації мережевого контролера

У наступному прикладі сценарію конфігурації показано, як створити кластер мережевого контролера з декількома вузлами й установити застосунок мережевого контролера. Крім того, змінна $cert вибирає сертифікат зі сховища сертифікатів локального комп'ютера, що відповідає рядку ім'я суб'єкта "networkController.contoso.com".

Дії після розгортання для розгортань без обліку Kerberos

Якщо ви не використовуєте Kerberos з розгортанням мережевого контролера, необхідно розгорнути сертифікати.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це об’єктно-орієнтована мова програмування високого рівня загального призначення з відкритим кодом. Це визначення може бути важким для новачків, тому розглянемо кожну характеристику окремо, щоб зрозуміти, що вона означає:

– Відкритий вихідний код: це безкоштовно та доступно для подальших покращень, таких як додавання корисних функцій або виправлення помилок.

– Об’єктно-орієнтована: заснована не на функціях, але в об’єктах з певними атрибутами й методами.

– Високий рівень: зручний для людини, а не для комп’ютера.

– Загальне призначення: можна використовувати для створення будь-яких програм.

Ця мова використовується в будь-якому програмному забезпеченні, про яке ви тільки можете подумати. Ви можете використовувати його для створення веб-сайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу та багато іншого. Також застосовується в науці про дані, аналізі даних,

машинному навчанні, інженерії даних, веб-розробці, розробці програмного забезпечення та інших галузях.

Переваги та недоліки Python

Переваги:

– Її легко читати, вчити та писати. Це мова програмування високого рівня з англійським синтаксисом. Це полегшує читання та розуміння коду. Її дійсно легко зрозуміти і вивчити, тому багато людей рекомендують Python новачкам. Вам потрібно менше рядків коду для виконання того ж завдання в порівнянні з іншими основними мовами, такими як C/C++ та Java.

– Підвищує продуктивність. Це дуже продуктивна мова. Завдяки її простоті розробники можуть зосередитися на розв'язанні проблеми. Їм не потрібно витрачати багато часу на розуміння синтаксису або поведінку мови програмування. Ви пишете менше коду та виконуєте більше завдань.

– Інтерпретована мова. Python мова, що інтерпретується, а це означає, що вона безпосередньо виконує код по рядку. Якщо сталася помилка, вона зупиняє подальше виконання та повідомляє про її виникнення. Вона показує лише одну помилку, навіть якщо у програмі їх кілька. Це спрощує налагодження.

– Динамічно типізована. Python не визначає тип змінної, доки ми не запустимо код. Вона автоматично надає тип даних, коли відбувається процес виконання. Фахівець може не турбуватися про оголошення змінних та типи даних.

– Безкоштовна та з відкритим вихідним кодом. Ця мова постачається під схваленою OSI ліцензією з відкритим вихідним кодом. Це робить його безкоштовним для використання та розповсюдження. Ви можете завантажити вихідний код, змінити його та навіть розповсюджувати свою версію. Це корисно для організацій, які хочуть використати свою версію для розробки.

– Підтримка великих бібліотек. Стандартна бібліотека Python є величезною, ви можете знайти майже всі функції, необхідні для вашого завдання. Таким чином ви не залежите від зовнішніх бібліотек.

– Портативність .У багатьох мовах, таких як C/C++, потрібно змінити свій код, щоб запустити програму на різних платформах. З Python все інакше. Ви тільки пишете один раз і запускаєте її будь-де.

Недоліки:

– Низька швидкість. Вище ми обговорювали, що це інтерпретована мова з динамічною типізацією. Порядкове виконання коду часто призводить до повільного виконання. Динамічна природа Python також є причиною її низької швидкості, оскільки їй доводиться виконувати додаткову роботу при виконанні коду. Тому вона не підходить для цілей, де швидкість важливий аспект проєкту.

– Неефективна для пам’яті. Ця мова програмування використовує великий обсяг пам’яті, це може бути недоліком при створенні програм, коли віддають перевагу оптимізації пам’яті.

– Слабка у мобільних обчисленнях. Python зазвичай використовується у серверному програмуванні. Ми не бачимо – її на стороні клієнта або в мобільних програмах з таких причин: вона не заощаджує пам’ять і має повільну обчислювальну потужність у порівнянні з іншими мовами.

– Доступ до бази даних. Програмувати на цій мові легко, але коли ми взаємодіємо з базою даних, її не вистачає. Рівень доступу до бази даних у Python примітивний та недостатньо розвинений у порівнянні з іншими популярними технологіями.

– Помилки виконання. Це мова з динамічною типізацією, тому тип даних змінної може змінюватись у будь-який час. Змінна, що містить ціле число, у майбутньому може містити рядок, що може призвести до помилок виконання.

Застосування Python:

– Для аналізу даних. Дані стали цінним активом у будь-якій сучасній галузі, і більшість компаній зацікавлені у збиранні, обробці та аналізі релевантних даних, щоб витягти з них цінну інформацію для бізнесу. І тут Python виходить за межі будь-якої конкуренції. Python особливо цінна тим, що крім великої стандартної бібліотеки надає величезний набір додаткових модулів,

розроблених спеціально для аналітичних цілей. Найвідоміші бібліотеки Python для аналізу даних – це pandas і NumPy . Ці інструменти дозволяють робити з вашими даними майже все, наприклад, очищати і аналізувати їх, вивчати статистику або візуалізувати приховані тенденції у ваших даних.

– Для візуалізації даних. Візуалізація даних – це окрема частина аналізу даних, яка допомагає нам подавати інформацію, необроблену чи очищену, у більш змістовній формі. Тут Python знову входить у гру, пропонуючи широкий спектр інструментів візуалізації даних. Найпопулярніші з них – matplotlib і заснований на ній seaborn. Використовуючи їх, ми можемо створювати буквально всі види візуалізації: від найпростіших до складніших.

– Для машинного навчання. Машинне навчання (ML) є основою більшості завдань науки даних. Він є областю штучного інтелекту, пов’язаною з використанням алгоритмів, що дозволяють машинам вивчати закономірності та тенденції на основі історичних даних, щоб робити прогнози на основі невідомих даних. – Використовуючи методи ML, ми можемо створювати моделі, які можуть точно передбачити швидкість відтоку клієнтів компанії, оцінити ризик виникнення у людини певного захворювання, визначити оптимальне розташування автомобілів таксі й т.д. За допомогою Python ми можемо побудувати модель ML, використовуючи лише три рядки коду.

– Для розробки програмного забезпечення. Крім свого багатостороннього застосування в галузях науки про дані, Python використовується на кожному етапі розробки програмного забезпечення, включаючи контроль складання, автоматичну безперервну компіляцію, прототипування, відстеження помилок, тестування та обслуговування програмного забезпечення. За допомогою цієї мови можемо створювати аудіо- або відеопрограми на основі методів штучного інтелекту, машинного навчання, API (інтерфейсів прикладного програмування), GUI (графічних інтерфейсів) або будь-якого іншого типу програмного забезпечення.

– Для веброзробки. У той час як для створення візуальної частини вебсайту ми переважно будемо використовувати такі мови, як HTML, CSS та JavaScript, для його невидимої частини ми часто вибираємо Python. Серед масштабних вебсайтів та програм, створених за допомогою цієї мови, варто згадати Google, Facebook, Instagram, YouTube, Dropbox та Reddit.

– Для автоматизації задач/скриптингу. Це відмінний інструмент для написання програм для автоматизації різних завдань, що повторюються. Цей процес називається скриптингом. Зокрема, можна робити скрипти для роботи з файлами та папками. Наприклад, можна створювати, перейменовувати, перетворювати, розділяти, об’єднувати або видаляти файли, перевіряти їх на наявність помилок. Ви також можете використовувати автоматизацію Python для пошуку та завантаження інформації з Інтернету, заповнення та надсилання онлайн-форм та надсилання регулярних повідомлень або електронних листів.

Яким фахівцям потрібно володіти Python:

– Фахівець з даних.

– Аналітик даних.

– Інженер даних.

– Інженер з машинного навчання.

– Журналіст даних.

– Архітектор даних.

– Повний стек веб-розробника.

– Backend-розробник.

– DevOps-інженер.

– Інженер-програміст.

Можемо зробити висновок, що Python ще довго буде популярною мовою, хоч і має низку недоліків. Цю мову використовують для створення вебсайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу, аналізу даних, машинного навчання, інженерії даних та для багатьох інших областей. Це перспективна і затребувана навичка, яка необхідна у всіх галузях.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи Software Define Network побудованих з використанням комутаторів Agema.

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 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) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Віртуалізація поширюється сьогодні не тільки на сервери, але й на мережі, і на системи зберігання – цього вимагає бізнес, не готовий платити за невикористовувані IT-ресурси, що так ще вимагають для свого розгортання недозволено багато часу. Вся стратегія компанії буде тепер будуватися навколо SDN і SDDC (Програмно-визначаємі ЦОД Software-Defined Data Center), за допомогою яких можна автоматизувати стандартні функції начебто створення віртуальних машин і розподілу ресурсів зберігання. Засоби SDN дозволяють ефективніше й простіше управляти хмарними конфігураціями. Для корпоративного сектора це можливість керування й оптимізації складною інфраструктурою, а для середнього й малого бізнесу – інструмент ефективної роботи з публічними хмарами. зрушення, Що Відбувається, парадигми споживання з апаратних і програмних продуктів на сервіси вимагає для своєї реалізації нових рішень.

Глобальний ринок масштабованих програмно-визначених мереж становив 29,11 млрд доларів США у 2025 році та, за прогнозами, зросте з 37,00 млрд доларів США у 2026 році приблизно до 320,26 млрд доларів США до 2035 року, зростаючи зі середньорічним темпом зростання 27,10% з 2026 по 2035 рік. Ринок масштабованих програмно-визначених мереж переживає безпрецедентне зростання, зумовлене зростаючим попитом на мережеву ефективність та зростаючою інтеграцією штучного інтелекту (ШІ) для інтелектуальної автоматизації.

Основні моменти ринку:

– Північна Америка домінувала на ринку, займаючи найбільшу частку ринку в 52,4% у 2025 році.

– Очікується, що Азіатсько-Тихоокеанський регіон зростатиме з найшвидшим середньорічним темпом зростання (CAGR) на рівні 26,5% у період з 2026 по 2035 рік.

– За типом/технологією сегмент SD-WAN зайняв найбільшу частку ринку – 36,8% у 2025 році.

– За типом/технологією очікується, що сегмент гібридних SDN зростатиме зі стійким середньорічним темпом зростання (CAGR) на рівні 24,4% у період з 2026 по 2035 рік.

– За компонентами, сегмент SDN-рішень/контролерів займав основну частку ринку в 57,8% у 2025 році.

– За компонентами, сегмент послуг (інтеграція, консалтинг, підтримка) готовий до значного зростання зі значним середньорічним темпом зростання (CAGR) у 24,3% між 2026 і 2035 роками.

– За кінцевими користувачами, сегмент підприємств забезпечив найбільшу частку ринку – 48,7% у 2025 році.

– Очікується, що сегмент постачальників послуг з точки зору кінцевих користувачів зростатиме зі вражаючим середньорічним темпом зростання (CAGR) у 24,6% між 2026 і 2035 роками.

– За моделлю розгортання, сегмент локальних рішень зайняв найбільшу частку ринку – 65,4% у 2025 році.

– За моделлю розгортання, хмарний (SaaS) сегмент зростає з двозначним середньорічним темпом зростання (CAGR) на рівні 25,1% у період з 2026 по 2035 рік.

– За застосуванням, сегмент мереж центрів обробки даних займав найбільшу частку ринку – 52,4% у 2025 році.

– За застосуванням, прогнозується, що сегмент хмарних обчислень зростатиме зі значним середньорічним темпом зростання (CAGR) на рівні 25,0% у період з 2026 по 2035 рік.

Огляд ринку

Масштабовані програмно-визначені мережі стосуються програмованих мережевих архітектур, які відокремлюють площину керування від площини даних, що дозволяє централізований мережевий аналіз та контроль політик незалежно від базового обладнання. Таке архітектурне роз'єднання дозволяє адміністраторам визначати, змінювати та забезпечувати поведінку мережі за допомогою програмного забезпечення, покращуючи видимість, узгодженість та операційний контроль у складних середовищах.

Платформи SDN підтримують динамічний розподіл ресурсів, віртуалізацію мережі та багатокористувацьку архітектуру, що робить їх добре придатними для масштабного розгортання в центрах обробки даних, корпоративних мережах та телекомунікаційних інфраструктурах. Абстрагуючи мережеві функції, SDN забезпечує швидше налаштування, спрощене керування конфігурацією та ефективнішу маршрутизацію трафіку на основі попиту в режимі реального часу.

Завдяки автоматизації та централізованому застосуванню політик, масштабована SDN покращує гнучкість мережі, оптимізацію продуктивності та економічну ефективність. Ці можливості особливо важливі для хмарних робочих навантажень, програмно-визначених центрів обробки даних та середовищ з високим трафіком, які потребують швидкого масштабування, надійності та гнучкої оркестрації мережі.

Як штучний інтелект впливає на зростання індустрії масштабованих програмно-визначених мереж?

З розвитком технології штучного інтелекту, інтеграція штучного інтелекту стимулює інновації та прискорює зростання на ринку масштабованих програмно-визначених мереж, посилюючи безпеку, прогнозну аналітику та можливості автоматизації мережі. Штучний інтелект та машинне навчання зменшують залежність від ручного налаштування мережі, постійно аналізуючи моделі

трафіку, поведінку пристроїв та показники продуктивності у великих та складних мережевих середовищах.

Штучний інтелект забезпечує прогнозне обслуговування, виявляючи ранні ознаки апаратного збою, зниження продуктивності та відхилення конфігурації до того, як відбудеться переривання обслуговування. Моделі машинного навчання виявляють аномалії та загрози безпеці в режимі реального часу, дозволяючи контролерам SDN ізолювати уражені сегменти, застосовувати політики та автоматично усувати проблеми, покращуючи стійкість мережі та час безвідмовної роботи.

Аналітика на базі штучного інтелекту покращує прогнозування перевантажень та оптимізацію трафіку, динамічно коригуючи політики маршрутизації на основі пріоритету програм та стану мережі. Ці можливості сприяють переходу до більш автономних, самооптимізуючих мереж, які можуть адаптуватися до змін робочих навантажень, знижувати операційні ризики та підтримувати масштабоване розгортання в центрах обробки даних, корпоративних мережах та телекомунікаційних інфраструктурах.

Перспективи ринку масштабованих програмно-визначених мереж

– **Огляд зростання галузі**: Очікується, що між 2025 і 2030 роками галузь зазнає прискореного зростання. Зростання ринку масштабованих програмно-визначених мереж зумовлене зростаючим впровадженням хмарних сервісів та віртуалізованих середовищ, швидким розширенням мереж 5G та Інтернету речей, а також зростаючою потребою в автоматизації мереж. Крім того, очікується, що зростаючий попит на безпечні, гнучкі та масштабовані рішення для підключення стимулюватиме зростання ринку в найближчі роки.

– **Глобальна експансія**: Кілька провідних гравців на ринку масштабованих програмно-визначених мереж (SDN), включаючи Cisco Systems, Intel, Hewlett Packard Enterprise (HPE), Juniper Networks, Huawei та IBM, активно розширюють свою географічну присутність за допомогою стратегічних ініціатив, таких як запуск нових послуг, партнерства/співпраця та придбання, щоб

задовольнити постійний попит на автоматизацію мереж та хмарну інтеграцію. Наприклад, у травні 2025 року Hewlett Packard Enterprise (HPE) оголосила про розширення свого портфоліо дротових та бездротових продуктів HPE Aruba Networking, а також про нові розподілені комутатори послуг HPE Aruba Networking CX 10K, які оснащені вбудованими програмованими блоками обробки даних (DPU) від AMD Pensando для розвантаження служб безпеки та мережі, щоб звільнити ресурси для обробки складних робочих навантажень штучного інтелекту.

– **Основні інвестори**: Кілька великих компаній та стратегічних інвесторів активно працюють на ринку масштабованих програмно-визначених мереж (SDN), включаючи провідні технологічні компанії, венчурні компанії та мережеві фірми, такі як VMware, Juniper Networks, Hewlett Packard Enterprise (HPE), Nokia Corporation, Huawei Technologies Co., Ltd., Cisco та IBM. Ці інвестиції стимулюють інновації завдяки пропонованим передовим продуктам.

3.2 Розробка структурної схеми

Аналітика типу/технології

SD-WAN: Цей сегмент займав найбільшу частку ринку – 36,8% у 2025 році. Зростання сегмента зумовлене широким впровадженням хмарних сервісів, зростаючим попитом на економічно ефективні мережеві рішення та зростаючою потребою в покращеній безпеці та централізованому управлінні мережею. Традиційні глобальні мережі (WAN) часто мають проблеми із затримкою та поганим користувацьким досвідом для хмарних додатків, тоді як SD-WAN долає ці проблеми, забезпечуючи прямий доступ до хмари у філіях.

Гібридні SDN: Очікується, що цей сегмент зростатиме зі вражаючим середньорічним темпом зростання (CAGR) на рівні 24,4% у період з 2026 по 2035 рік, завдяки його здатності інтегрувати традиційні мережеві архітектури з можливостями програмно-визначених мереж. Такий підхід надає підприємствам

та постачальникам послуг гнучкість для підтримки міграції в хмару, розгортання 5G, підключення до Інтернету речей та автоматизації мереж на основі штучного інтелекту без повної заміни застарілої інфраструктури. Моделі гібридних SDN дозволяють організаціям поступово модернізувати свої мережі, накладаючи централізоване керування, програмованість та автоматизацію політик на існуюче обладнання.

Аналітика режиму розгортання

Локальні розгортання: Цей сегмент домінує в секторі масштабованих програмно-визначених мереж, займаючи частку 65,4%, завдяки суворим законам про конфіденційність даних та зростаючій потребі в контролі даних. Локальний режим розгортання є кращим там, де конфіденційність даних та відповідність вимогам є критично важливими. Локальні розгортання дозволяють повний контроль даних та безпеки для управління конфіденційними робочими навантаженнями, особливо у фінансах, охороні здоров'я та державних організаціях.

Хмарні рішення (SaaS): Цей сегмент є найшвидше зростаючим у галузі масштабованих програмно-визначених мереж, зі середньорічним темпом зростання 25,1%. Модель розгортання хмарних рішень (SaaS) пропонує різні переваги, такі як економічна ефективність (оплата за використання), гнучкість та масштабованість. Хмарні рішення підтримують віддалену роботу, безперешкодну інтеграцію з зростаючими гібридними/багатохмарними середовищами та усувають необхідність у великих початкових капіталовкладеннях. Хмарні рішення (SaaS) забезпечують безперешкодну інтеграцію та управління в різних хмарних екосистемах.

Статистика кінцевих користувачів

Підприємства: Цей сегмент домінує в секторі масштабованих програмно-визначених мереж, займаючи мажоритарну частку в 48,7%, що пов'язано зі зростаючою потребою в гнучкості, автоматизації, економічній ефективності та високому рівні безпеки для підтримки цифрової трансформації, впровадження

хмарних технологій, гібридних моделей роботи та Інтернету речей. Кілька підприємств у різних секторах, таких як фінанси, охорона здоров'я та роздрібна торгівля, широко впроваджують хмарні технології (SaaS) для модернізації, підвищення операційної ефективності та покращення взаємодії з клієнтами.

Постачальники послуг: Цей сегмент є найшвидше зростаючим на ринку масштабованих програмно-визначених мереж, зі середньорічним темпом зростання 24,6%. Програмно-визначені мережі дозволяють телекомунікаційним постачальникам ефективно керувати зростаючим трафіком даних, що включає централізоване керування, автоматизацію, управління трафіком та масштабованість мережі. Крім того, очікується, що зростаюча потреба в гнучкості, ефективності, управлінні трафіком та підтримці постійних потреб, таких як 5G, хмарні сервіси та Інтернет речей, буде стимулювати зростання сегмента протягом прогнозованого періоду.

Аналітика компонентів

SDN-рішення/контролери: Цей сегмент домінує на ринку масштабованих програмно-визначених мереж, займаючи 57,8% частки. Зростання сегмента підтримується зростаючим попитом на централізоване керування, віртуалізацію мережі та автоматизацію. Крім того, зростаюча потреба в оптимізації витрат та плавній інтеграції з хмарними сервісами, Інтернетом речей та мережами 5G сприяє зростанню сегмента протягом прогнозованого періоду.

Послуги (інтеграція, консалтинг, підтримка): Цей сегмент є найшвидше зростаючим сегментом на ринку масштабованих програмно-визначених мереж і, як очікується, зростатиме зі середньорічним темпом зростання (CAGR) на рівні 24,3% через зростаючу важливість централізованого контролю, зростаючу потребу в управлінні трафіком і сплеск мережевих загроз, що спонукають компанії вживати заходів безпеки. Послуги включають інтеграцію, консалтинг і підтримку. Послуги відіграють життєво важливу роль в інтеграції з хмарою та задоволенні потреб цифрової трансформації.

Аналітика додатків

Мережі для центрів обробки даних: Цей сегмент домінує на ринку масштабованих програмно-визначених мереж, займаючи частку 52,4%. Зростання сегмента мереж для центрів обробки даних зумовлене кількома факторами, такими як зростаючий попит на гнучке, автоматизоване, економічно ефективне, масштабоване та ефективне управління мережею для хмарних обчислень та віртуалізації. Масштабовані програмно-визначені мережі можуть керувати складними мережевими потоками та автоматизувати операції. Швидке зростання хмарних обчислень та віртуалізації значно збільшує попит на SDN для ефективного управління динамічними робочими навантаженнями та ресурсами.

Хмарні обчислення: Цей сегмент є найшвидше зростаючим сегментом ринку масштабованих програмно-визначених мереж, з темпами зростання 25,0% завдяки зростаючому впровадженню хмарних технологій. Кілька постачальників хмарних послуг широко використовують SDN для оптимізації своєї інфраструктури. Крім того, швидка цифрова трансформація дозволяє більшій кількості підприємств переходити в хмару, а програмно-визначені мережі стають вирішальними для управління гібридними та багатохмарними конфігураціями.

Регіональні аналітичні дані

Обсяг ринку масштабованих програмно-визначених мереж у Північній Америці оцінюється в 15,25 мільярда доларів США у 2025 році та, за прогнозами, досягне приблизно 168,14 мільярда доларів США до 2035 року зі середньорічним темпом зростання 27,13% з 2026 по 2035 рік.

Північна Америка домінує на ринку масштабованих програмно-визначених мереж, займаючи 38% частки у 2025 році, що підтримується зрілою екосистемою хмарних обчислень, активним впровадженням корпоративних IT та масштабними інвестиціями в цифрову інфраструктуру. Регіон отримує вигоду від широкого розгортання гіпермасштабних центрів обробки даних, передових корпоративних мереж та раннього впровадження програмованих мережевих

архітектур, які покладаються на SDN для централізованого керування, автоматизації та масштабованості.

Сполучені Штати та Канада сприяють цьому домінуванню завдяки розширенню інфраструктури 5G, розгортанню периферійних обчислень та значним інвестиціям у програмно-визначені центри обробки даних. Телекомунікаційні оператори, постачальники хмарних послуг та великі підприємства все частіше використовують SDN для управління складними середовищами з високим трафіком, підтримки сегментації мережі та забезпечення послуг з низькою затримкою.

Такі високоцінні сектори, як оборонна, аерокосмічна, автомобільна та критично важлива інфраструктура, ще більше підсилюють попит на масштабовані SDN-рішення. Ці галузі потребують високозахищених, стійких та програмованих мереж для підтримки автономних систем, розробки ADAS, критично важливих комунікацій та обробки даних у режимі реального часу. Паралельно зі зростанням фінансування досліджень та розробок, державними програмами цифрової модернізації та оборонними контрактами, що підтримуються ШІ, прискорюється впровадження SDN-платформ на базі штучного інтелекту, що підтримує лідерство Північної Америки на ринку масштабованих програмно-визначених мереж.

Розмір ринку масштабованих програмно-визначених мереж у США оцінюється в 11,44 мільярда доларів США у 2025 році та, як очікується, досягне майже 126,94 мільярда доларів США у 2035 році, прискорившись зі стійким середньорічним темпом зростання (CAGR) у 27,21% між 2026 і 2035 роками.

Сполучені Штати трансформують ринок масштабованих програмно-визначених мереж. Сполучені Штати є основним фактором розвитку ринку масштабованих програмно-визначених мереж у північноамериканському регіоні. У країні спостерігається значне розширення центрів обробки даних для хмарних обчислень, штучного інтелекту та машинного навчання, що стимулює попит на FPGA для прискорення робочих навантажень та управління величезними

обсягами даних. Країна є світовим лідером в інноваціях і є домівкою для великих компаній, що займаються розробкою FPGA, таких як AMD/Xilinx, Microchip Technology, Lattice, Intel та інші. Крім того, очікується, що зростання інвестицій у центри обробки даних, аерокосмічну та оборонну галузь, а також зростаюче впровадження штучного інтелекту, автономних транспортних засобів (ADAS) та технологій 5G сприятимуть зростанню ринку протягом прогнозованого періоду.

Азіатсько-Тихоокеанський регіон є регіоном, що найшвидше розвивається на ринку масштабованих програмно-визначених мереж, де зареєстровано середньорічний темп зростання (CAGR) становить 26,5%, що зумовлено швидким розширенням цифрової інфраструктури та агресивними ініціативами з модернізації мережі. Такі країни, як Китай, Японія, Південна Корея, Індія та Тайвань, значно інвестують у хмарні центри обробки даних, віртуалізацію телекомунікацій та цифровізацію корпоративних мереж, створюючи високий попит на масштабовані архітектури SDN.

Зростання регіону зумовлене головним чином масштабним розгортанням мереж 5G, де SDN відіграє центральну роль у розподілі мережі, оркестрації трафіку та наданні послуг з низькою затримкою. Телекомунікаційні оператори в Азіатсько-Тихоокеанському регіоні все частіше впроваджують SDN для управління щільними, гетерогенними мережами та підтримки швидкого зростання мобільного трафіку даних, периферійних обчислень та приватного розгортання 5G.

Азіатсько-Тихоокеанський регіон також є глобальним центром виробництва споживчої електроніки, проектів розумних міст та впровадження Інтернету речей. Зростаюче розгортання підключених пристроїв, систем промислової автоматизації та програм на базі штучного інтелекту вимагає гнучких, програмованих та централізовано керованих мереж, що прискорює впровадження програмно-визначених мереж (SDN) на підприємствах та у постачальників послуг. Очікується, що ця конвергенція розширення 5G, впровадження хмарних технологій та зростання Інтернету речей підтримуватиме

потужний імпульс для масштабованих програмно-визначених мереж у всьому регіоні протягом прогнозованого періоду.

Китай переживає значне зростання. У країні добре зарекомендувала себе присутність відомих напівпровідникових компаній. Країна має великий обсяг виробництва споживчої електроніки, особливо пристроїв Інтернету речей та розумного дому, смартфонів, дронів, ігрових консолей, телевізорів та камер (з передовою обробкою зображень/відео). Зростання країни підтримується потужною цифровою інфраструктурою, швидким розгортанням мереж 5G, зростаючим розширенням автомобільної промисловості, підтримуючими урядовими ініціативами та високим рівнем впровадження нових технологій, таких як штучний інтелект, машинне навчання та Інтернет речей (IoT).

Європейський регіон займає значну частку на ринку масштабованих програмно-визначених мереж, що підтримується раннім і широким впровадженням передових цифрових мережевих технологій у багатьох галузях. Такі країни, як Німеччина, Франція та Велика Британія, є лідерами у впровадженні SDN завдяки значним інвестиціям у розгортання 5G, штучний інтелект та високопродуктивну обчислювальну інфраструктуру, що вимагає програмованих, низькозатримкових та високомасштабованих мережевих архітектур.

Стійке зростання витрат на дослідження та розробки, особливо в автомобілебудуванні, промисловій автоматизації та розумному виробництві, прискорює попит на мережі з підтримкою SDN, які можуть підтримувати обмін даними в режимі реального часу, периферійні обчислення та безпечне підключення. Європейські державні структури, що сприяють цифровій трансформації, разом із суворими вимогами до захисту даних та надійності мережі, ще більше посилюють впровадження централізованих та політично керованих моделей управління мережами.

Крім того, швидке розширення аерокосмічних та оборонних програм по всій Європі стимулює попит на високозахищені програмно-визначені мережі,

здатні підтримувати критично важливі комунікації, середовища моделювання та розподілені системи управління. Зростаюча потреба в програмованості, стійкості та сумісності мереж між цивільними та оборонними програмами продовжує позиціонувати Європу як ключовий ринок для масштабованих програмно-визначених мережевих рішень.

Ринок масштабованих програмно-визначених мереж у Німеччині переживає значне зростання, зумовлене головним чином надійною напівпровідниковою інфраструктурою, широким розгортанням мережевої інфраструктури 5G та швидким технологічним прогресом, таким як штучний інтелект, машинне навчання та Інтернет речей.

Зростання регіону також значною мірою зумовлене зростанням попиту в різних галузях, таких як автомобільна (системи допомоги водієві, інформаційно-розважальні системи, автономне водіння, об'єднання датчиків), 5G та телекомунікації (високошвидкісні мережі 5G), побутова електроніка (смартфони, камери, телевізори та ПК), аерокосмічна та оборонна промисловість (радар, обробка сигналів, управління польотами та безпілотні літальні апарати), а також промисловість/IoT (промислова автоматизація).

Очікується, що ринок масштабованих програмно-визначених мереж зростатиме значними темпами в регіоні Близького Сходу та Африки, що підтримуватиметься прискоренням розвитку цифрової інфраструктури та зростанням інвестицій у гіпермасштабовані та колокаційні центри обробки даних. Такі країни, як Об'єднані Арабські Емірати та Саудівська Аравія, впроваджують архітектури з підтримкою SDN для підтримки хмарних сервісів, платформ розумних міст та урядових програм цифрової трансформації.

Зростання також пояснюється зростаючим розгортанням мереж 5G, які потребують програмно-визначеного керування для оркестрації трафіку, сегментації мережі та управління затримками. Телекомунікаційні оператори та підприємства в регіоні впроваджують SDN для модернізації застарілих мереж та покращення масштабованості в розподілених середовищах.

Зростаючий попит з боку таких секторів, як автомобілебудування, побутова електроніка, енергетика та логістика, у поєднанні зі зростанням витрат на дослідження та розробки та швидким впровадженням технологій віртуалізації, посилює впровадження SDN. Ці фактори разом підтримують стабільне розширення масштабованих програмно-визначених мережевих рішень у регіоні Близького Сходу та Африки протягом прогнозованого періоду.

Південно-Африканська республіка переживає вражаюче зростання. Зростання країни зумовлене зростанням цифрової трансформації та модернізацією інфраструктури, особливо в галузі телекомунікацій та обробки даних. Очікується, що поява технології 5G, зростання попиту на споживчу електроніку, зростання попиту на високопродуктивні обчислення та різке збільшення державного фінансування сприятимуть зростанню ринку в країні. Крім того, очікується, що зростання впровадження передових технологій, таких як штучний інтелект, хмарні обчислення, автономні транспортні засоби (ADAS) та Інтернет речей, разом зі значним зростанням центрів обробки даних, сприятиме розширенню ринку масштабованих програмно-визначених мереж протягом прогнозованого періоду.

Площина даних та площина керування розділені в новому підході до управління мережею, який називається програмно-визначеною мережею (SDN). Це дозволяє одній людині керувати мережами з одного місця та спрощує їх програмування. Ця робота є оглядом еволюції SDN, архітектури та її інтеграції з віртуалізацією мережевих функцій (NFV) та її застосування в IT-операціях, включаючи управління центрами обробки даних, мережеву безпеку та нові технології, включаючи 5G. Дослідження дозволяє синтезувати основні характеристики, такі як централізоване управління, автоматизація, масштабованість та покращена безпека, а також представити практичні переваги, такі як гнучкість, економічна ефективність та спрощене адміністрування мережі. По-друге, у роботі розглядаються технічні проблеми, такі як масштабованість контролерів та сумісність зі старими системами, а також вразливість

централізованих систем, які перешкоджають їх широкому впровадженню. Огляд нещодавньої літератури пропонує різноманітні рішення, такі як системи аналізу трафіку та енергоощадні моделі, аж до безпеки на основі блокчейну та розподіленого керування, але більшість з них орієнтовані на одну точку. Цей огляд, завдяки об'єднанню цих висновків, показує, що в дослідженнях існують прогалини, і що для підтримки стійких та інтелектуальних IT-інфраструктур необхідні цілісні, адаптивні та сумісні структури SDN.

Система, яка розробляється у даній роботі, побудована на базі Agema Delta AG9064 64X100G QSFP28 SWITCH Broadcom Tomahawk2.

AG9064 з Broadcom Tomahawk2 забезпечує 64 порти 100GbE QSFP28 з низькою затримкою. 2RU порти 64X100GE QSFP28 високої щільності. Пристрій попередньо завантажений ONIE (Open Network Install Environment), готовий підтримувати вибір клієнта модульних розширюваних мережевих операційних систем (NOS), 3-річна гарантія на обладнання.

AG9064 – це високопродуктивний 2RU комутатор Spine наступного покоління високої щільності, призначений для розгортання магістральних мереж Enterprise Campus, центрів обробки даних та постачальників послуг. Він має шістдесят чотири порти 100GbE QSFP28, які забезпечують пропускну здатність 6,4 Тбіт/с, яку можна налаштувати на багатошвидкісний 40/100GbE з бажаною швидкістю тканини. Окрім широкої пропускної здатності, AG9064 підтримує можливості програмування, забезпечує нові функції та протоколи, орієнтовані на майбутнє, а також має комплексні можливості функцій другого та третього рівнів, включаючи STP, VLAN, QinQ, Trunk, QoS, LAG, повну маршрутизацію IPv4/IPv6, RIP, OSPF, IGMP, DVMRP, PIM-DM, PIM-SM, ACL тощо. AG9064 також спрямований на мережу наступного покоління та підтримує розширені функції, такі як VxLAN, L2GRE, NVGRE, GENEVE, NSH, MPLS, TRIL PPB-TE L2/L2 у тунелюванні L3, віртуалізацію серверів VN-Tag, VEPA, керування потоком PFC та ETS, щоб відповідати вимогам мережі DCB. Щоб забезпечити чіткий шлях до програмно-визначеного центру обробки даних, AG9064

підтримує ONIE для автоматичного встановлення змінних мережевих операційних систем.

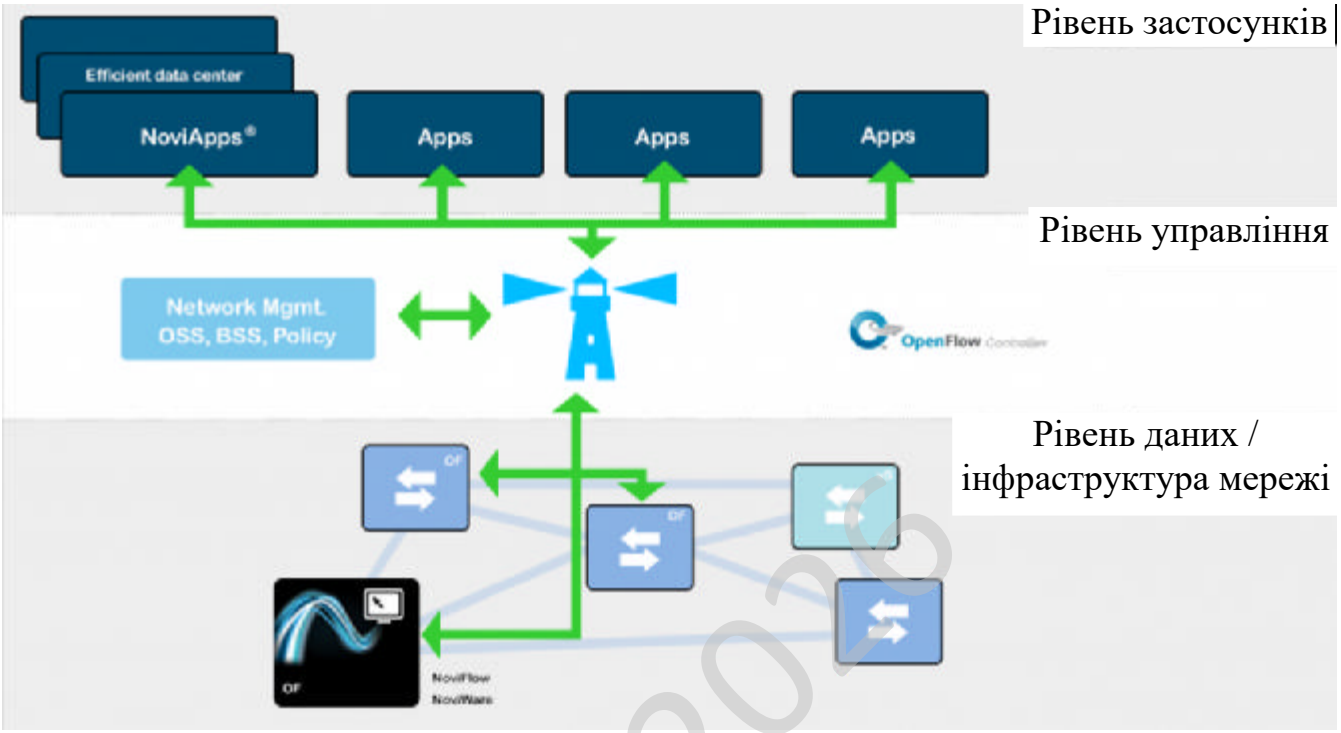

Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

На рисунку 3.2 наведена функціональна схема системи. Нижче перераховані функціональні блоки і їхнє призначення:

– **NFVI** (Network Function Virtualization Infrastructure): **інфраструктура віртуалізації мережних функцій**, у якій розгортаються й працюють віртуальні мережні функції VNF. NFVI фізично складається зі стандартного обчислювального устаткування загального призначення COTS (Commercial Off The Shelf: сервери, сховища, комутатори). У цьому фізична інфраструктура NFVI відрізняється від фізичної інфраструктури, у якій працюють фізичні мережні функції, – це звичайне, «монолітне» мережне устаткування.

– **VNF** (Virtualized Network Functions): **віртуалізовані мережні функції**, які працюють в інфраструктурі NFVI.

– **PNF** (Physical Network Functions): **фізичні мережні функції**, які реалізовані на виділеному фізичному встаткуванні й спеціалізованому програмному забезпеченні. Це устаткування й ПЗ успадковані від традиційної архітектури операторської мережі.

– **VIM** (Virtualized Infrastructure Manager): **менеджер віртуалізованої інфраструктури**, що управляє обчислювальними, зберігаючими й мережними ресурсами NFVI (серверами, системами зберігання й комутаторами). Він також може управляти **фізичною інфраструктурою** PNF.

– **VNFM** (VNF Manager): **менеджер віртуальних мережних функцій,** що відповідає за керування «життєвим циклом» **VNF**: розгортання, модифікація, масштабування, зчіпка для створення мережних послуг і застосунків, вимикання й деактивація).

– **Загальний Оркестратор EEO** (End-to-End Orchestrator), **функція оркестрації мережних послуг**, що відповідає за керування «життєвим циклом» мережних послуг. Функція оркестрації має також під-функції, що ставляться до різних аспектів оркестрації. Оркестрація VNF виконується NFVO (NFV Orchestrator), що є частиною загального оркестратора.

– **Каталоги й репозиторії,** збори файлів дескрипторів, шаблонів робочих процесів, скриптів надання послуг (provisioning scripts), та ін., які використовуються менеджером VNFM, загальним оркестратором і оркестратором послуг і застосунків роботи з VNF і для складання віртуальних мережних послуг NFV і SDN.

– **Функція забезпечення надійності послуг SA** (Service Assurance), що збирає попереджуючі повідомлення й моніторить дані. Застосунки усередині SA, або працюючі з SA, можуть використовувати ці дані для кореляції помилок, аналізу вихідних причин подій, аналізу впливу послуг, адміністрування SLA, моніторингу безпеки й аналітики, та ін.

– **SDN-контролери,** які можуть включати чотири види контролерів:

– **SDN-контролер дата-центру, DC SDN Controller**, що створює логічні мережі усередині дата-центра (у якому фізично перебуває інфраструктура NFVI).

– **Контролер глобальної мережі SDN WAN** (SDN WAN Controller), що організує логічні підключення через глобальну мережу WAN (Wide Area Network) між дата-центрам, у яких перебуває розподілена інфраструктура NFVI. Такі підключення називають DCI – Data Center Interconnect.

– **Контролер доступу SDN** (Access SDN Controller), відповідає за керування доменами провідного або бездротового доступу мережі оператора.

– **Контролери доменів**, специфічних для вендорских рішень (Domain/ Vendor-Specific Controller). Такі контролери можуть знадобитися у випадку обробки специфічних вендорських або технологічних доменів, при відсутності стандартних інтерфейсів або для цілей масштабування.

– **Оркестратор послуг і застосунків**, що взаємодіє з порталом користувача, і відповідає за надання каталогу послуг для цього порталу.

– **Портали**, які включають користувальницькі портали, де користувачі можуть замовляти, модифікувати й контролювати їхніх послуг, а також портал для операційного персоналу оператора (Ops portal).

– **EMS** (Element Management System) **система керування мережними елементами,** це успадкована система керування традиційними мережними елементами, побудованими на фізичному мережному встаткуванні.

– **OSS/BSS**, **системи керування операціями й бізнес-процесами** (Operations Support Systems and Business Support Systems), також успадковані від традиційної архітектури мережі оператора. OSS/BSS замовляють необхідні функцій, забезпечують роботу й надійність послуг, биллинг, формування тикетов про помилки, взаємодія з персоналом техпідтримки й виконують інші функції для операційних і бізнес-процесів мережі оператора.

Слід зазначити, що хоча ці системи показані усередині архітектури SDN/NFV, це найбільш проблемна область при реалізації проекту віртуалізації

мережі оператора зв'язку. Інтерфейси між OSS/BSS дотепер не пророблені досить детально. Частково це пояснюється розмаїтістю пропрієтарних рішень OSS/BSS від різних вендорів.

Рисунок 3.2 – Функціональна схема системи

Різні типи **каталогів і репозиторіев** включають:

– C1 – каталог VNF (наприклад, BNG, PCRF…).

– C2 – каталог мережних послуг SDN, наприклад, послуга IP VPN, послуга керування мережею через певні довжини хвиль (lambda service) для транспортної SDN (T-SDN) та ін.

– C3 – каталог мережних послуг, побудованих з VNF.

C1 використовується VNFM для керування життєвим циклом VNF. Файли дескрипторів повідомляють VNFM як створювати VNF, у т.ч. ідентифікувати віртуальні машини (VM) для роботи VNF, визначають необхідні логічні шляхи через мережу, властивості масштабування VNF, та ін.

C2 використовується SDN-контролером, причому, інформація про те, як управляти послугами, використовуючи моделі даних, перебуває в самому контролері.

C3 використовується загальним оркестратором. Дескриптор мережної послуги може містити інформацію про передатного графа VNF (VNF forwarding graph) і пов'язаних із цим графом дескрипторах, віртуальних линках, логічних мережних з'єднаннях через WAN, а також про необхідний PNF і скриптах конфігурації, які можуть знадобитися для для складання й активації повної мережної послуги.

Помітимо, що певні послуги можуть використовуватися рекурсивним образом. Наприклад, послуга, представлена загальним оркестратором, може опиратися на послугу з'єднання, представлену контролером WAN SDN, і може бути опублікована в каталозі C2 послуг SDN, а послуга, що опирається тільки на функціонала VNF, може бути опублікована в каталозі VNF C1.

В архітектурі є також різні репозиторії, які обновляються на основі активності EEO, VNFM, VIM і SA:

– Екземпляри VNF.

– Ресурси VNF.

– Екземпляри мережних послуг і ресурсів.

– Інвентарна інформація, наприклад, інформація про топологію мережних послуг.

– Репозиторій попереджуючих повідомлень із SA і інформація про роботу й продуктивність послуги.

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. 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 проходе 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).

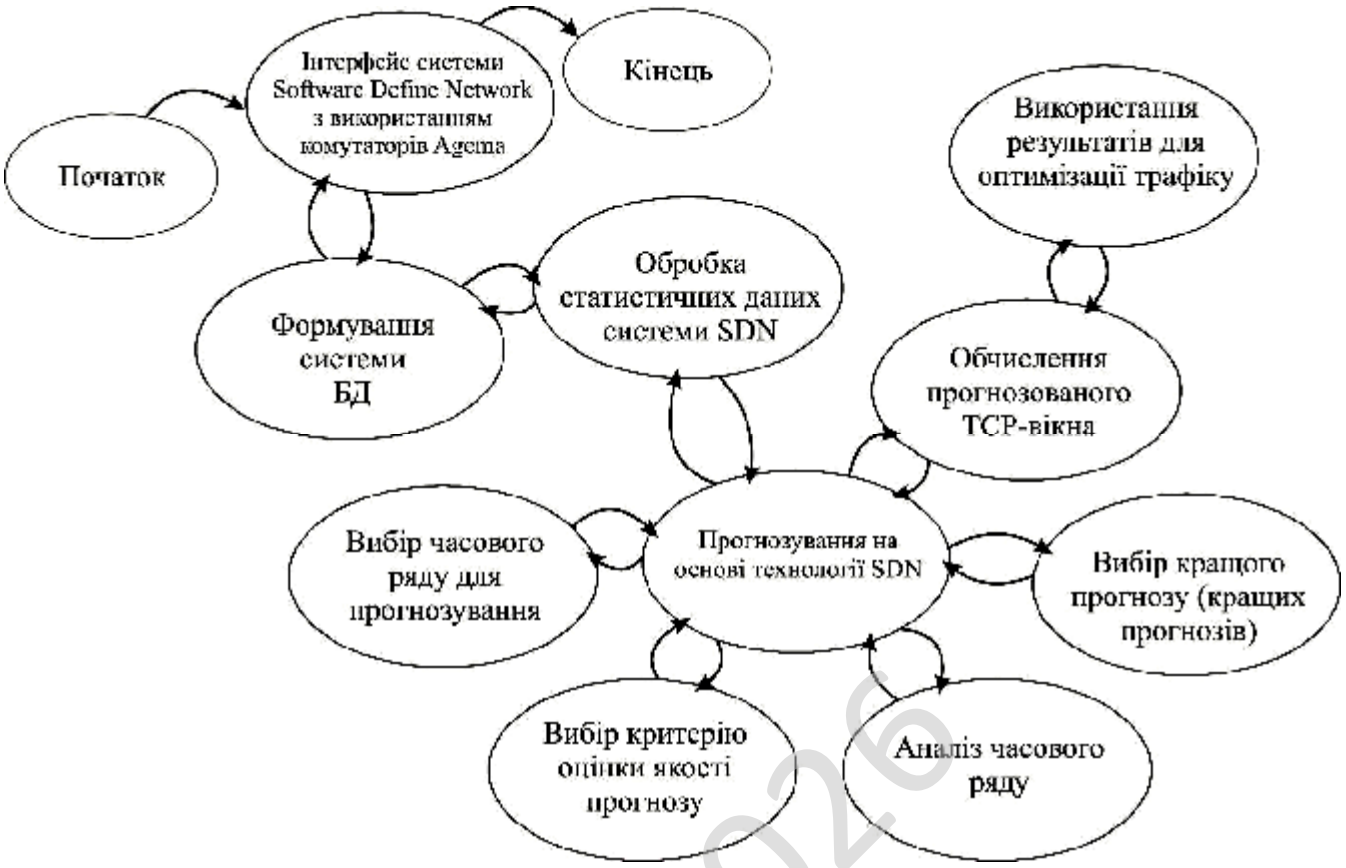

Рисунок 3.3 – Діаграма взаємодії процесів

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.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо.

Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються.

Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач. Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції. Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу.

Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується.

Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи. Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи). З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми.

Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

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. Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення.

UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

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

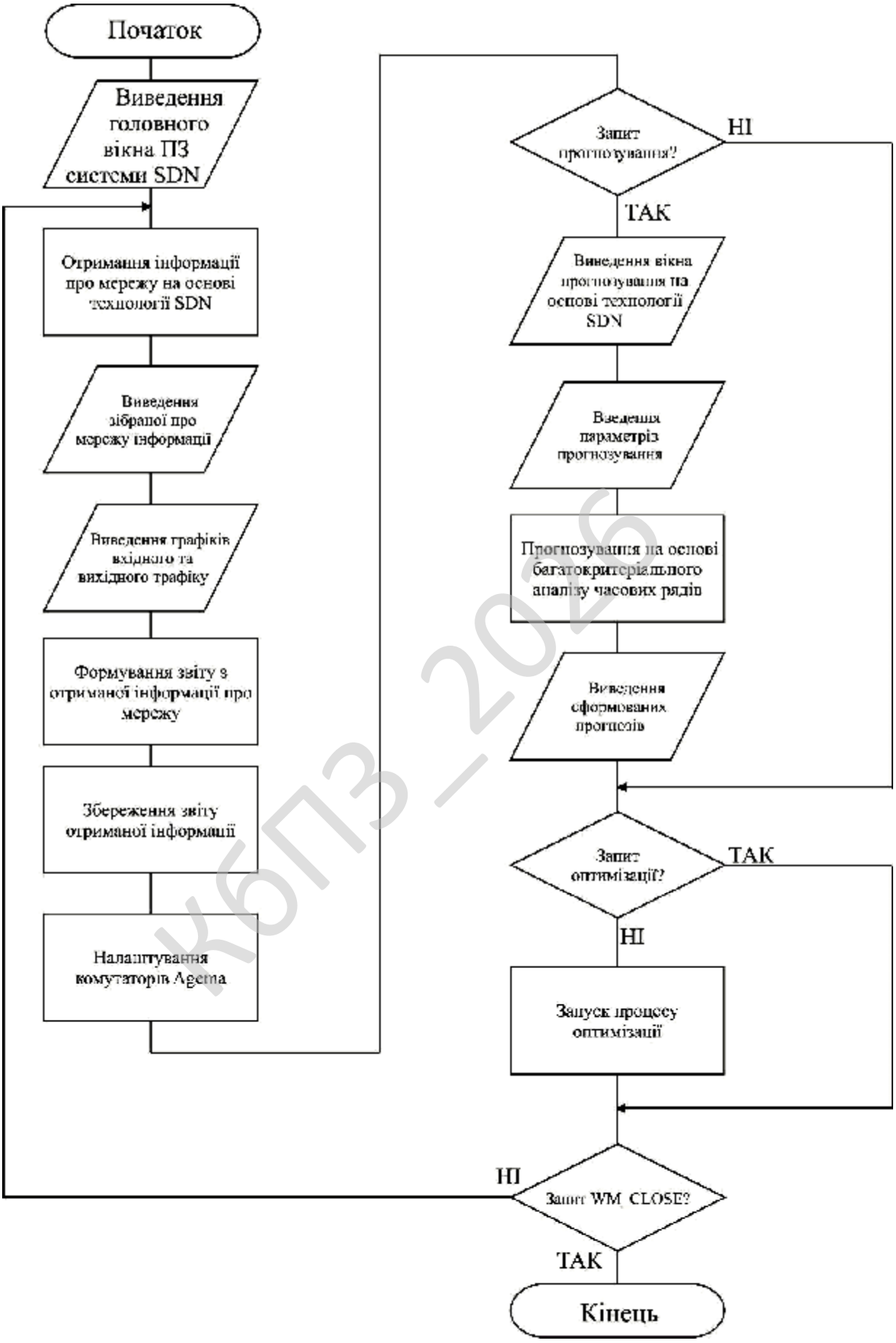

Рисунок 4.1 – Блок-схема основної програми

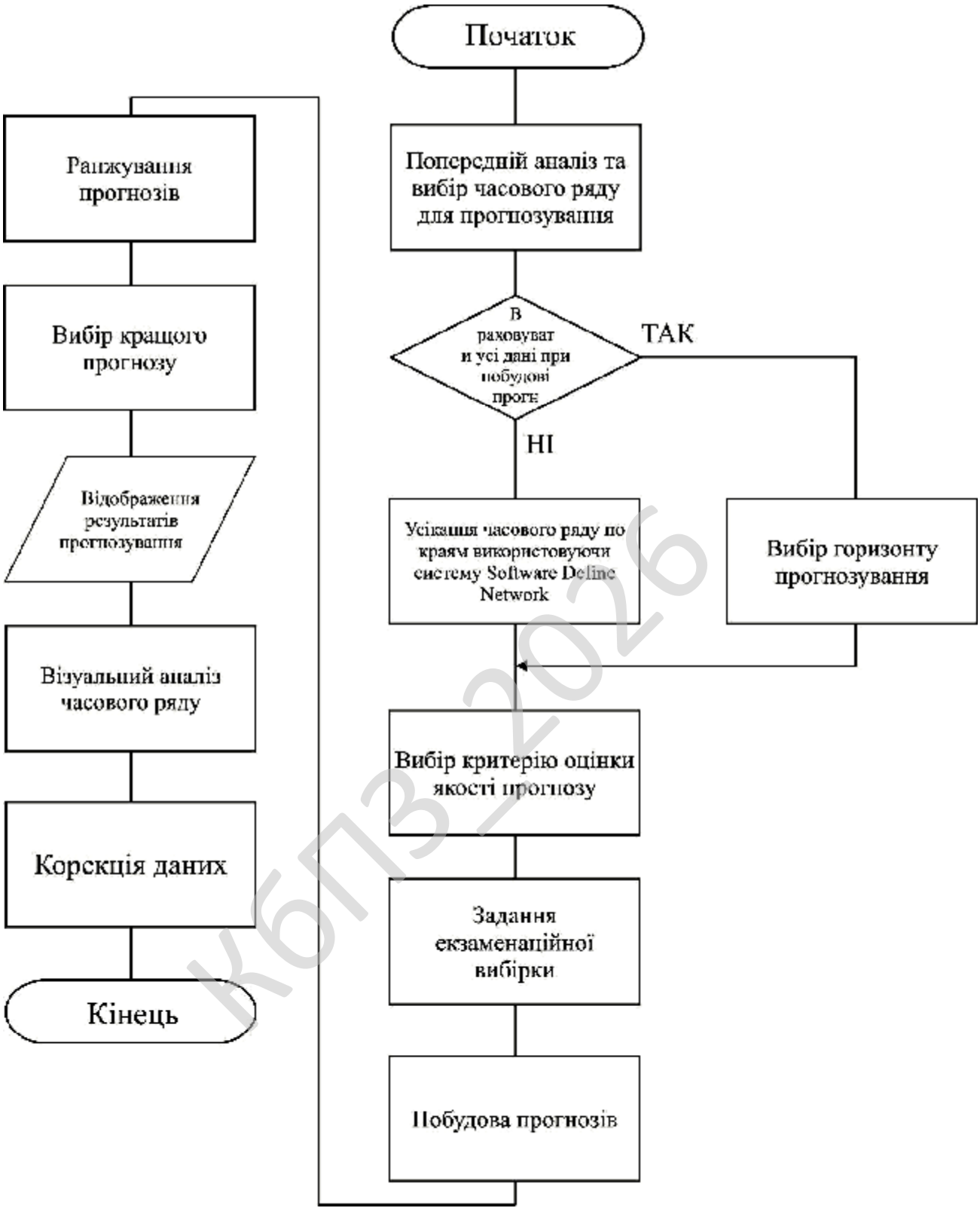

Рисунок 4.2 – Блок-схема роботи підпрограми

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.

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

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання.

Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою. У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати

діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого».

Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Управління вимогами це процес запису, аналізу, трасування, пріоритезації і узгодження вимог та контролю змін і доведення до їх зацікавлених сторін. Це безперервний процес протягом всього життя проекту. Вимога – якість, якій мають відповідати результати проекту (продукту або послуги).

Мета управління вимогами полягає в тому, щоб переконатися, що організація відповідає потребам і очікуванням своїх клієнтів, внутрішніх або зовнішніх зацікавлених сторін. Управління вимогами починається з аналізу і виявлення цілей і обмежень організації. Управління вимогами додатково включає в себе підтримку планування вимог, інтеграції вимог і організації роботи з ними (атрибути для вимог).

Управління вимогами передбачає спілкування між членами проектної групи і зацікавленими сторонами, і адаптацію до змін у вимогах протягом всього проекту. Щоб запобігти перетину поля одного класу вимог з іншим, постійні зв'язки між членами команди розробників є критичними. Наприклад, при розробці програмного забезпечення для внутрішнього використання у бізнесу можуть бути

настільки сильні потреби, що він може проігнорувати вимоги користувачів, або вважати, що створені сценарії використання покриють також і користувальницькі вимоги.

Відслідковування вимоги фактично означає документування всього життєвого циклу вимоги. Часто необхідно дізнатися першоджерело кожної вимоги. Для цього всі зміни вимог повинні бути задокументовані, щоб досягти стану повного відстеження. Відстежувати треба бути навіть використання реалізованих вимог.

Вимоги мають різні джерела, такі як ділова людина, що замовляє продукт, менеджер зі збуту і фактичний користувач. У всіх цих людей є різні вимоги до продукту. Використовуючи відслідковування вимог, реалізована в системі функція може бути простежена назад до людини або групі, яка замовляла її під час збору вимог. Ця особливість може, наприклад, використовуватися в процесі розробки для пріоритезації вимог, визначаючи, наскільки цінною є дана вимога для певного користувача.

Відслідковування може також використовуватися після розгортання продукту. Наприклад, коли вивчення використання системи показує, що якась функція не використовується, можна визначити навіщо вона була потрібна спочатку.

Завдання управління вимогами

На кожному етапі процесу розробки існують ключові методи і задачі пов'язані з управлінням вимогами. Для ілюстрації, розглянемо наприклад стандартний процес розробки з п'ятьма фазами: дослідженням, аналізом здійсненності, дизайном, розробкою та тестуванням і випуском.

Дослідження. Під час фази дослідження збираються перші три класи вимог від користувачів, бізнесу і команди розробників. У кожній області задають однакові питання: які цілі, які обмеження, які використовуються процеси та інструменти і так далі. Тільки коли ці вимоги добре зрозумілі, можна приступати до розробки функціональних вимог.

Тут необхідне застереження: незалежно від того, як сильно група намагається це зробити, вимоги не можуть бути повністю визначені на початку проекту. Деякі вимоги змінюються, або тому що вони просто не були знайдені спочатку, або тому що внутрішні чи зовнішні сили торкаються проекту в середині циклу. Таким чином, учасники групи повинні спочатку погодитися, що головна умова успіху – гнучкість у мисленні та діях.

Результатом стадії дослідження є документ – специфікація вимог, схвалений усіма членами проекту.

Пізніше, в процесі розробки, цей документ буде важливий для запобігання розповзання меж проекту або непотрібних змін. Оскільки система розвивається, кожна нова функція відкриває світ нових можливостей, таким чином специфікація вимог прив'язує команду до оригінального бачення системи і дозволяє контрольоване обговорення змін.

У той час як багато організацій все ще використовують звичайні документи для керування вимогами, інші управляють своїми базовими вимогами, використовуючи програмні інструменти.

Ці інструменти керують вимогами використовуючи базу даних, і зазвичай мають функції автоматизації відстеження (наприклад, дозволяючи створювати зв'язки між батьківськими і дочірніми вимогами, або між тестами і вимогами), управління версіями, і управління змінами. Зазвичай такі інструментальні засоби містять функцію експорту, яка дозволяє створювати звичайний документ, експортуючи дані вимог.

Аналіз здійсненності

На стадії аналізу здійсненності визначається вартість вимог. Для користувальницьких вимог поточна вартість роботи порівнюється з майбутньою вартістю встановленої системи.

Задаються питання такі як: «Скільки нам зараз варті помилки введення даних?» Або, «Яка вартість втрати даних через помилки оператора пов'язаної з використовуваним інтерфейсом?». Фактично, потреба в новому інструменті часто розпізнається, коли подібні питання потрапляють до уваги людей, що займаються в організації фінансами.

Ділова вартість включає відповіді на такі питання як: «У якого відділу є бюджет на це?» «Який рівень повернення коштів від нового продукту на ринку?» «Який рівень скорочення внутрішніх витрат на навчання і підтримку, якщо ми зробимо нову, більш просту в використанні систему?»

Технічна вартість пов'язана з вартістю розробки програмного забезпечення та апаратною вартістю. «Чи є у нас потрібні люди, щоб створити інструмент?» «Чи потребуємо ми нове устаткування для підтримки нової системи?»

Подібні питання дуже важливі. Група повинна з'ясувати, чи буде новий автоматизований інструмент мати достатню ефективність аби перенести частину тягаря користувачів на систему і зекономити час людей.

Ці питання також вказують на основну суть управління вимогами. Людина і інструмент формують систему, і це розуміння особливо важливе, якщо інструмент – комп'ютер або новий додаток на комп'ютері. Людський розум вкрай ефективний у паралельній обробці та інтерпретації тенденцій з недостатніми даними. Комп'ютерний процесор ефективний у послідовній обробці і точному математичному обчисленні. Основна мета управління вимогами для програмного проекту полягала б у тому, щоб гарантувати, що автоматизована робота призначена «правильному» процесору.

Наприклад, «не змушуйте людину пам'ятати, де вона знаходиться в системі. Примусьте інтерфейс завжди повідомляти про місцезнаходження людини

в системі ». Або «не змушуйте людини вводити ті ж самі дані в два екрани. Примусьте систему зберігати дані і заповнювати їх де необхідно автоматично». Результатом стадії аналізу здійсненності є бюджет і графік проекту.

Дизайн. Припускаючи, що вартість точно визначена і переваги, які будуть отримані, є досить великими, проект може перейти до стадії проектування.

На стадії дизайну основна діяльність управління вимогами полягає в тому, щоб перевіряти чи відповідають результати дизайну документу вимог, щоб упевнитися, що робота залишається в межах проекту.

І знову, гнучкість є ключем до успіху. Ось класичний приклад змін проекту, які відмінно працювали. Проектувальники Форда на початку 1980-х очікували, що ціни на бензин піднімуться до 3,18 дол за галон до кінця десятиліття. На середині процесу дизайну автомобіля Ford Taurus, ціни встановилися приблизно на рівні 1,50 дол за галон. Колектив дизайнерів вирішив, що вони могли б створити більший, більш зручний, і більш потужний автомобіль, якщо б ціни на бензин залишилися низькими. Таким чином, вони перепроектувати автомобіль. Коли новий автомобіль вийшов, він встановив загальнонаціональні рекорди продажів.

У більшості випадків, однак, відступ від оригінальних вимог до такої міри не працює. Таким чином документ вимог стає ключовим інструментом, який допомагає команді приймати рішення про зміни дизайну.

Розробка та тестування. На стадії розробки і тестування, основна діяльність управління вимогами – це гарантувати, що робота і ціна залишаються в межах графіка і бюджету, і що створюваний інструмент дійсно відповідає вимогам. Основним інструментом, використовуваним на цій стадії, є створення прототипу і ітераційне тестування. Для програмного додатка користувацький інтерфейс може бути створений на папері і перевірений з потенційними користувачами, в той час як створюється основа програми. Результати цих тестів записуються в керівництві по дизайну користувацького інтерфейсу і передаються

колективу дизайнерів. Це економить їх час і робить їх завдання набагато простіше.

При розробці ПЗ було використано підходи ри́зик-ме́неджменту – це система управління ризиками, яка включає в себе стратегію та тактику управління, направлені на досягнення основних цілей. Ефективний ризик-менеджмент включає:

– систему управління;

– систему ідентифікації і вимірювання;

– систему супроводження (моніторингу та контролю).

Сучасна наука представляє ризик як вірогідну подію, в результаті настання якої можуть відбутися позитивні, нейтральні або негативні наслідки. Якщо ризик припускає наявність як позитивних, так і негативних результатів, він відноситься до спекулятивних ризиків. Якщо ж наслідки негативні, або відсутні взагалі, такий ризик іменується чистим.

Мета ризик-менеджменту – підвищення конкурентоспроможності господарюючих суб'єктів за допомогою захисту від реалізації чистих ризиків.

Теорія ризик-менеджменту ґрунтується на трьох базових поняттях: корисності, регресії і диверсифікації.

У 1738 швейцарський математик Даніель Бернуллі доповнив теорію вірогідності методом корисності або привабливості того або іншого результату подій. Ідея Бернуллі полягала в тому, що в процесі ухвалення рішення люди приділяють більше уваги розміру наслідків різних результатів, ніж їх вірогідність.

В кінці XIX століття англійський дослідник Ф. Гальтон запропонував вважати регресію або повернення до середнього значення універсальною статистичною закономірністю. Суть регресії трактувалася ним як повернення явищ до норми з часом. Згодом було доведено, що правило регресії діє в найрізноманітніших ситуаціях, починаючи з азартних ігор та розрахунку вірогідності виникнення нещасних випадків, і закінчуючи прогнозуванням коливань економічних циклів.

У 1952 аспірант Університету Чикаго Гарі Марковіц в статті «Диверсифікація вкладень» («Portfolio Selection») математично обґрунтував стратегію диверсифікації інвестиційного портфеля, зокрема, він показав, як шляхом продуманого розподілу вкладень мінімізувати відхилення прибутковості від очікуваного показника. У 1990 Г. Марковіцу присуджена Нобелівська премія за розробку теорії і практики оптимізації портфеля фондових активів.

Етапи ризик-менеджменту

У ризик-менеджменті прийнято виділяти декілька ключових етапів:

– на першому етапі відбувається виявлення ризику з супутньою оцінкою вірогідності його реалізації і масштабу наслідків;

– на другому етапі здійснюється розробка ризик-стратегії з метою зниження вірогідності реалізації ризику і мінімізації можливих негативних наслідків;

– на третьому етапі вибираються методи і інструменти управління виявленим ризиком;

– на четвертому етапі проводиться безпосереднє управління ризиком;

– на завершальному етапі оцінюються досягнуті результати і коректується ризик-стратегія.

За ключовий етап ризик-менеджменту вважається етап вибору методів і інструментів управління ризиком.

Методи і інструментарій ризик-менеджменту

Базовими методами ризик-менеджменту є відмова від ризиків, зниження, передача і ухвалення.

Ризик-інструментарій значно ширший. Він включає політичні, організаційні, правові, економічні, соціальні інструменти, причому ризик-менеджмент як система допускає можливість одночасного застосування декількох методів і інструментів ризик-управління.

Найбільш часто вживаним інструментом ризик-менеджменту є страхування. Страхування припускає передачу відповідальності за відшкодування передбачуваного збитку сторонній організації (страхової компанії).

Прикладами інших інструментів можуть бути відмова від надмірно ризикової діяльності (метод відмови), профілактика або диверсифікація (метод зниження), аутсорсинг витратних ризикових функцій (метод передачі), формування резервів або запасів (метод ухвалення).

Подійно-орієнтована архітектура (Event-driven architecture, надалі EDA) – шаблон архітектури програмного забезпечення, який призначений для створення подій, їх виявлення, споживання і реагування на них.

Подія може бути визначена як значна зміна стану. Наприклад, коли споживач купує автомобіль, стан автомобіля змінюється з "на продаж" до "продано".

Архітектура системи дилера автомобілів може трактувати цю зміну стану як подію, поява якої може стати відомою іншим програмам даної архітектури.

З формальної точки зору, те, що виробляється, публікується, поширюється, виявляється і споживається (як правило, асинхронно) є повідомленням, яке називають сповіщенням про подію (або нотифікацією), а не самою подією, яка є зміною стану, що викликає появу повідомлення.

Події не подорожують, вони просто відбуваються. Проте термін подія часто використовується метонімічно для позначення самого нотифікаційного повідомлення, що може призвести до певної плутанини.

Цей архітектурний шаблон може застосовуватися при проектуванні і реалізації ПЗ і систем, які передають події між слабкозв’язаними компонентами програмного забезпечення і сервісами (службами).

Подійно-орієнтована система як правило складається з емітерів подій (або агентів) і споживачів подій (або стоків).

Стоки несуть відповідальність за здійснення реагування на появу події. Реакція не завжди може бути повністю забезпечена самим стоком. Наприклад,

стік, може бути відповідальним лише за фільтрацію, трансформацію і відправку події до іншого компонента або він може забезпечити повністю самостійну реакцію на таку подію.

Перша категорія стоків може бути заснована на традиційних компонентах, таких як проміжне програмне забезпечення, орієнтоване на обробку повідомлень (message oriented middleware, MOM), в той час, як друга категорія стоків (самостійна реакція в режимі он-лайн) може вимагати більш придатної платформи (фреймворку) для виконання транзакцій.

Розробка ПЗ і систем в подійно-орієнтованій архітектурі дозволяє їм бути сконструйованими способом, який більш відповідає вимогам до їх створення, оскільки такі системи в більшій мірі пристосовуються до непередбачуваних і асинхронних середовищ.

Подійно-орієнтована архітектура (EDA) може доповнювати сервісно-орієнтовану архітектуру (SOA), оскільки сервіси (служби) можуть бути активовані тригерами, які ініціюються при настанні подій.

Ця парадигма особливо корисна, коли стік не забезпечує власного виконання будь-яких дій.

Подійно-орієнтована SOA (SOA-2) розвиває архітектури SOA і EDA для забезпечення більш глибокого і надійного рівня сервісів за рахунок використання раніше невідомих причинно-наслідкових зв'язків, щоб сформувати новий шаблон подій. Цей новий шаблон бізнес-аналітики дає поштовх до небаченого раніше зростання рівня автоматизації підприємства за рахунок привнесення додаткової цінної інформації в описану раніше модель діяльності.

Обчислювальна техніка та сенсорні пристрої (сенсори, датчики, контролери) можуть виявляти зміни стану об'єктів або умов і створювати події, які потім можуть бути оброблені сервісом (службою) або системою.

Подія може складатися з двох частин: заголовка події та тіла події. Заголовок події може включати в себе інформацію таку як, наприклад, назва події, часова мітка події і тип події. Тіло події – це частина, яка описує факт, що стався в

дійсності. Тіло події не слід плутати з шаблоном або логікою, яка може бути застосована як реакція на саму події.

Архітектура, керована подіями, складається з чотирьох логічних рівнів (шарів). Вона починається з виявлення факту, його технічного подання у формі події і закінчується непустою множиною реакцій на цю подію.

Генератор подій.

Першим логічним шаром є генератор подій, який виявляє факт і представляє цей факт подією. Оскільки фактом може бути практично все, що може бути сприйнято, то ним може бути і генератор подій. Наприклад, генератором може бути клієнт електронної пошти, система електронної комерції або певний тип датчика.

Перетворення різних даних, отриманих від датчиків, в єдину стандартизовану форму даних, які можуть бути оцінені, є основною проблемою при розробці та реалізації цього шару. Однак, враховуючи, що подія є строго декларативною, можна легко застосовувати будь-які операції трансформації, тим самим усуваючи необхідність забезпечення високого рівня стандартизації.

Канал подій.

Канал подій – це механізм, через який інформація від генератора подій передається до обробника подій (event engine) або стоку.

Це може бути з'єднання TCP/IP або вхідний файл будь-якого типу (простий текст, формат XML, e-mail тощо).

В один і той же час може бути відкрито кілька каналів подій. Як правило, оскільки обробник подій повинен працювати в режимі, наближеному до реального часу, канали подій зчитуються асинхронно. Події зберігаються в черзі, очікуючи наступної обробки механізмом обробки подій.

Механізм обробки подій.

Механізм обробки подій (event processing engine) є місцем, де подія ідентифікується і вибирається відповідна реакція на нього, яка потім виконується. Це також може призвести до породження ряду тверджень. Якщо подія, яка

надійшла до механізму обробки подій, є наприклад такою «Запаси продукту ID досягли нижнього допустимого рівня», це може ініціювати, наприклад, такі реакції як «Замовити продукт ID» і «Сповістити персонал».

Наступна подійно-орієнтована дія (післядія).

Щодо того, як можуть проявлятися наслідки події, слід відмітити, що вони можуть проявитись багатьма різними способами і у різноманітних формах (наприклад, повідомлення електронної пошти, надіслане комусь, або ПЗ, що виводить деяке попередження на екран). Залежно від рівня автоматизації, який забезпечується стоком (механізмом обробки подій), ці дії можуть виявитись зайвими.

Є три основні стилі обробки подій: простий, потоковий і складний. Часто ці три стилі використовуються спільно у розвинутій подійно-орієнтованій архітектурі.

Проста обробка подій.

Проста обробка подій стосується подій, які безпосередньо належать до специфічних вимірних змін умов. У випадку простої обробки подій, мають справу з появою відомих подій, що ініціюють післядію (післядії). Проста обробка подій зазвичай використовується для управління потоком робіт в реальному часі, скорочуючи тим самим час затримки і вартість робіт.

Наприклад, прості події можуть створюватись (породжуватись) датчиком, що виявляє зміну тиску в шині або температуру навколишнього середовища.

Обробка потоку подій.

При обробці потоку подій (event stream processing, далі ESP) відбуваються як звичайні, так і відомі події. Звичайні події (заявки, передачі RFID) перевіряються на те, чи є вони відомими, і передаються інформаційним передплатникам. Обробка потоку подій зазвичай використовується для управління потоком інформації в реальному часі і на рівні підприємства, що дозволяє своєчасно приймати рішення.

Обробка складних подій.

Обробка складних подій (Complex event processing (CEP)) дозволяє за шаблонами простих і звичайних подій проводити аналіз того, чи наступила складна подія. Обробка складних подій полягає в оцінюванні взаємного впливу подій і в наступному виконанні дій. При цьому, типи подій (відомих або звичайних) можуть перетинатись, а події можуть виникати протягом тривалого періоду часу.

Кореляція подій може бути причинною, тимчасовою або просторовою. CEP вимагає використання складних інтерпретаторів подій, визначення і підбору шаблонів подій, а також відповідних кореляційних методів.

Обробка складних подій зазвичай використовується для виявлення і реагування на аномальну поведінку, загрози і можливості у бізнесі.

Незважаючи на те що я працював над ПЗ один в реалізації програми я використовував підходи пришвидшення розробки на основі методологій Agile – Extreme Programming.

Екстремальне програмування (Extreme Programming, далі XP) це методологія розробки програмного забезпечення, найпопулярніша серед так званих гнучких методологій. Має на меті поліпшення якості програмного забезпечення та чутливість до змін у вимогах замовників. Як вид гнучких методологій, XP радить часті "випуски" програми у коротких циклах розробки, що має на меті поліпшити продуктивність праці та покращити можливості виконання вимог замовника що змінюються. Авторами даної методології є Кент Бек, Ворд Каннінгем, Мартін Фаулер та інші.

Інші елементи екстремального програмування включають в собі: парне програмування, проведення обширної перевірки сирцевого коду, модульне тестування всього коду, уникання створення функціональності до того як вона дійсно необхідна, простота та ясність коду, очікування на зміну вимог замовників з плином часу та коли вимоги до продукту стають ясніші, досить часте спілкування із замовником та між самими програмістами.

Назва методології походить від ідеї застосувати корисні методи і практики розробки програмного забезпечення, піднявши їх до "екстремальних" рівнів.

Критики XP зауважують на потенційні недоліки цієї методології – нестабільні вимоги, незадокументовані компроміси конфліктів користувачів, відсутність загального документу дизайну програми.

Технологія екстремального програмування була розроблена Кентом Беком, Уардом Каннінґхемом та Роном Джеффріесом під час роботи над Chrysler Comprehensive Compensation System (C3). У 1996 Кент Бек став лідером проекту і почав вдосконалювати методи розробки, що застосовувалися в роботі над проектом. Свій метод він виклав у книзі «Extreme Programming Explained», котру було видано у жовтні 1999. Після купівлі Крайслера компанією Даймлер–Бенц проект C3 було скасовано у лютому 2000.

Хоча саме екстремальне програмування є відносно новим, багато її практик вже існували і використовувались протягом певного часу; однак, методологія підносить "найкращі практики" до екстремального рівня. Для прикладу, практика по плануванню і написанню тестів перед написанням кожної маленької частини коду було використано раніше в проекті НАСА "Меркурій". Для зменшення часу на розробку ПЗ деякі формальні документи тестування (такі як приймальне тестування) писались паралельно (або й раніше) з написанням самого ПЗ. Незалежна група тестування НАСА може писати процедури тестування базуючись на формальних вимогах до продукту до того як програмне забезпечення розроблене та інтегроване в систему. В XP ця концепція піднесена до "екстремального рівня" завдяки написанню автоматичних тестів які перевіряють поведінку навіть малих частинок коду, а не тільки значних функціональних частин ПЗ.

Посібник Extreme Programming Explained: Embrace Change описує Екстремальне Програмування, як:

– Спроба примирити гуманність і продуктивність.

– Механізм для соціальної зміни.

– Шлях до удосконалення.

– Стиль розвитку.

Дисципліна розробки програмного забезпечення.

Головною метою Екстремального Програмування є скорочення вартості неочікуваних змін. У традиційних методах розробки (на кшталт SSADM) вимоги до розвитку системи визначаються на початку роботи над проектом, і часто виправляються пізніше. Це означає, що вартість проекту через зміни буде більшою за заплановану (традиційна особливість для програмного забезпечення, що проектується).

XP використовується для скорочення вартості змін, завдяки представленню простих значень, принципів і методів. При використанні екстремального програмування, проект повинен стати гнучкішим щодо змін.

Extreme Programming Explained описує екстремальне програмування як дисципліну розробки програмного забезпечення яка змушує людей створювати високоякісне ПЗ якомога швидше.

XP намагається зменшити ціну зміни вимог до ПЗ завдяки малим циклам розробки, а не одним довгим циклом. Екстремальне програмування сприймає зміни до вимог як звичайні, неминучі та бажані аспекти розробки ПЗ, і ці зміни мають бути очікуваним. Основна ідея полягає в тому що неможливо розробити самодостатній пакет вимог до ПЗ, зміни в вимогах – неминучі.

Екстремальне програмування також вводить набір практик та принципів на основі методології гнучкої розробки програмного забезпечення.

Екстремальне програмування описує чотири базові активності що виконуються при розробці програмного забезпечення: написання коду, тестування, слухання та дизайн.

Написання коду. Прихильники XP заявляють що єдиним дійсно важливим результатом розробки ПЗ є код: без готового коду нема продукту.

Тестування. Методологія екстремального програмування заявляє, що якщо дрібне тестування може перевірити незначну частину функціональності, то багато дрібних тестів можуть перевірити набагато більше частинок і продукт в цілому.

Основні прийоми XP. Дванадцять основних прийомів екстремального програмування (за першим виданням книги Extreme programming explained) можуть бути об'єднані в чотири групи:

1. Короткий цикл зворотного зв'язку (Fine scale feedback).

1.1. Розробка через тестування (Test driven development).

1.2 Гра в планування (Planning game).

1.3. Замовник завжди поруч (Whole team, Onsite customer).

1.4 Парне програмування (Pair programming).

2. Безперервний, а не пакетний процес.

2.1 Безперервна інтеграція (Continuous Integration).

2.2 Рефакторинг (Design Improvement, Refactor).

2.3 Часті невеликі релізи (Small Releases).

3. Розуміння, що поділяється всіма учасниками.

3.1 Простота (Simple design).

3.2 Метафора системи (System metaphor).

3.3 Колективне володіння кодом (Collective code ownership) або обраними шаблонами проектування (Collective patterns ownership).

3.4 Стандарт кодування (Coding standard or Coding conventions).

4. Соціальна захищеність програміста (Programmer welfare), а саме 40 годинний робочий тиждень (Sustainable pace, Forty hour week).

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SHACAL-2, який шифрує дані 256-бітними блоками з використанням 512-бітного ключа. Допускається використання ключів менших розмірів (не менш 128 біт), які доповнюються бітовими нулями до 512 біт.

Шифруємий блок даних ділиться на 8 фрагментів по 32 біта (які позначені буквами *A*...*H*). Алгоритм виконує 64 раунду перетворень, у кожному з яких дані фрагменти обробляються в такий спосіб:

$$T = H_i + S_1(E_i) + Ch(E_i, F_i, G_i) + M_i + K_i,$$

$$H_{i+1} = G_i,$$

$$G_{i+1} = F_i,$$

$$F_{i+1} = E_i,$$

$$E_{i+1} = D_i + T,$$

$$D_{i+1} = C_i,$$

$$C_{i+1} = B_i,$$

$$B_{i+1} = A_i,$$

$$A_{i+1} = T + S_0(A_i) + Maj(A_i, B_i, C_i).$$

де T – тимчасова змінна.

Використовувані функції визначені в такий спосіб:

$$S_0(x) = (x >>> 2) \oplus (x >>> 13) \oplus (x >>> 22),$$

$$S_1(x) = (x >>> 6) \oplus (x >>> 11) \oplus (x >>> 25),$$

$$Ch(x, y, z) = (x \,\&\, y) \oplus (x' \,\&\, z),$$

$$Maj(x, y, z) = (x \,\&\, y) \oplus (x \,\&\, z) \oplus (y \,\&\, z),$$

де >>> – операція побітового циклічного зрушення вправо.

Константи, що модифікують, M_i (i = 0…63) наведено нижче (одна за одною від M_0 до M_{63}):

428A2F98	71374491	B5C0FBCF	E9B5DBA5
3956C25B	59F111F1	923F82A4	AB1C5ED5
D807AA98	12835B01	243185BE	550C7DC3
72BE5D74	80DEB1FE	9BDC06A7	C19BF174
E49B69C1	EFBE4786	0FC19DC6	240CA1CC
2DE92C6F	4A7484AA	5CB0A9DC	76F988DA
983E5152	A831C66D	B00327C8	BF597FC7
C6E00BF3	D5A79147	06CA6351	14292967
27B70A85	2E1B2138	4D2C6DFC	53380D13
650A7354	766A0ABB	81C2C92E	92722C85
A2BFE8A1	A81A664B	C24B8B70	C76C51A3
D192E819	D6990624	F40E3585	106AA070
19A4C116	1E376C08	2748774C	34B0BCB5
391C0CB3	4ED8AA4A	5B9CCA4F	682E6FF3
748F82EE	78A5636F	84C87814	8CC70208
90BEFFFA	A4506CEB	BEF9A3F7	C67178F2

Фрагменти розширеного ключа $K_0…K_{63}$ обчислюються в процесі процедури розширення ключа, що виконується в такий спосіб:

Етап 1. 512-бітний вихідний ключ шифрування ділиться на 16 фрагментів по 32 біта $K_0…K_{15}$...

Етап 2. Інші фрагменти розширеного ключа $K_{16}…K_{63}$ обчислюються з перших 16 фрагментів у такий спосіб:

$$K_i = O_1(K_{i-2}) + K_{i-7} + O_0(K_{i-15}) + K_{i-16},$$

де функції O_0 і O_1 визначені так:

$$O_0(x) = (x >>> 7) \oplus (x >>> 18) \oplus (x >> 3),$$

$$O_1(x) = (x >>> 17) \oplus (x >>> 19) \oplus (x >> 10),$$

де >> – операція побітового зрушення (не циклічного) вправо.

Раунди розшифрування алгоритму виконуються у зворотній послідовності:

$$T = A_{i+1} + S_0'(B_{i+1}) + Maj'(B_{i+1}, C_{i+1}, D_{i+1}) + 2,$$

$$H_i = T + S_1'(F_{i+1}) + Ch'(F_{i+1}, G_{i+1}, H_{i+1}) + M_i' + K_i' + 4,$$

$$G_i = H_{i+1},$$

$$F_i = G_{i+1},$$

$$E_i = F_{i+1},$$

$$D_i = E_{i+1} + T' + 1,$$

$$C_i = D_{i+1},$$

$$B_i = C_{i+1},$$

$$A_i = B_{i+1}.$$

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у бакалаврської дипломної роботі система Software Define Network побудованих з використанням комутаторів Agema. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Функціональних кнопок ПЗ; Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

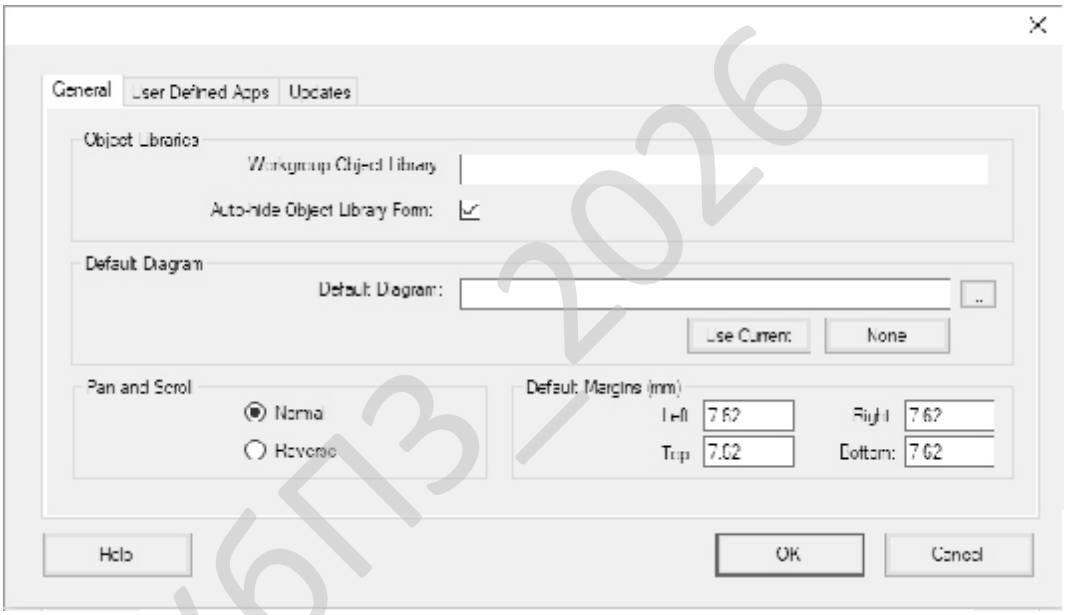

Рисунок 5.1 – Головне вікно ПЗ

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

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

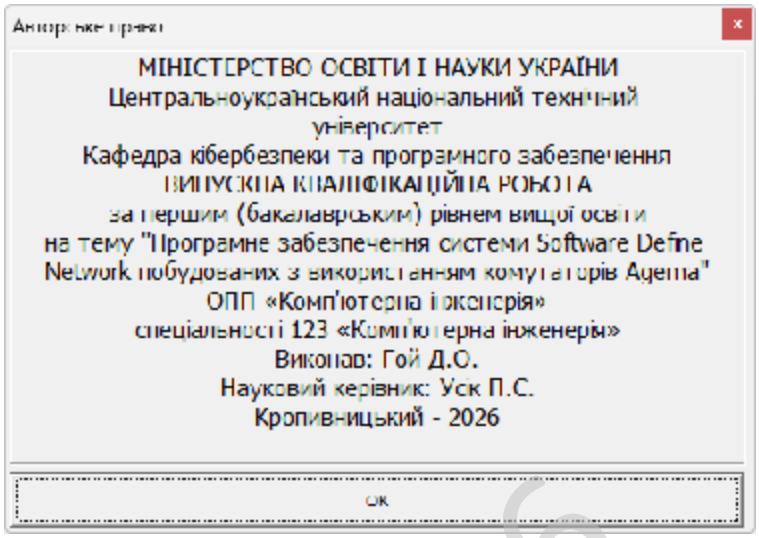

Рисунок 5.2 – Авторське право

Програмно-визначаєма мережа або програмно-конфігуруєма мережа (англ. software-defined networking, SDN) – мережа передачі даних, у якій рівень керування мережею відділений від пристроїв передачі даних і реалізується програмно. Одна з форм віртуалізації мережі. Ключові принципи програмно-визначаємих мереж – поділ процесів передачі й керування даними, централізація керування мережею за допомогою уніфікованих програмних засобів, віртуалізація фізичних мережних ресурсів.

Протокол OpenFlow, що реалізує незалежний від виробника інтерфейс між логічним контролером мережі й мережним транспортом, є однієї з реалізацій концепції програмно-визначаємої мережі й уважається рушійною силою її поширення й популяризації. Принципи програмно-визначаємих мереж сформулювали в 2006 році фахівці Берклі й Стенфорда. В 2009 році технології програмно-визначаємих мереж з'явилися в списку 10 швидкозростаючих технологій, який визначає щорічно MIT Technology Review, після чого стали

(поряд із протоколом Openflow) об'єктом пильної уваги не тільки в академічних дослідженнях, але й з боку комерційного сектора.

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.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– 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.

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.

Обрано умови розповсюдження – Freeware. Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи Software Define Network побудованих з використанням комутаторів Agema.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем Software Define Network побудованих з використанням комутаторів Agema.

– Досліджена система Software Define Network побудованих з використанням комутаторів Agema.

– На основі отриманих результатів досліджень створена програмна реалізація системи Software Define Network побудованих з використанням комутаторів Agema.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання Software Define Network побудованих з використанням комутаторів Agema.

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 створені 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 Software Define Network побудованих з використанням комутаторів Agema. Це 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 SHACAL-2.

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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 1». Cisco Press. 2020. – 848 p.

2. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 2 Premium Edition eBook and Practice Test». Cisco Press. 2020. – 624 p.

3. Scott Jernigan «CompTIA Network+ Certification All-in-One Exam Guide, Eighth Edition». 2022. – 976 p.

4. Doug Lowe «Networking For Dummies 12th Edition». 2020. – 480 p.

5. Ramon Nastase «Computer Networking: The Beginner's guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.

6. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

7. 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.

8. 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.

9. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

10. 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.

11. 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.

12. 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.

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

14. 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.

15. 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.

16. 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.

17. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

18. 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.

19. 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.

20. 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.

21. 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.

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

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. Smirnov O. Kuznetsov A., Zaicenko 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. Smirnov A.A., Kuzne tsov 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.

39. 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

40. 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.

41. 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.

42. Смірнов О.А., Конопліцька-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.

43. 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.

44. Смірнов О.А., Дрєєва Г.М., Дрєєв 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.

45. Смірнов О.А., Конопліцька-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.

46. 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).

47. Смірнов О.А., Дрєєва 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

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

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

50. Смірнов О.А., Стасєв Ю.В. Бараннік В.В. Захист інформації в автоматизованих системах управління. Навчальний посібник – Харків: ХУПС, 2015. – 264 с.

51. Смірнов О.А., Мелешко Є.В., Семенов С.Г. Методи та засоби обробки сигналів і даних в інформаційних системах. Навчальний посібник. – Кіровоград: КНТУ 2012. – 250 с.

52. Смірнов О.А., Євсеєв С.П., Жукарев В.Ю., Король О.Г., Сорокін В.Є., Мелешко Є.В. Технології і стандарти комп'ютерних мереж. Навчальний посібник. – Кіровоград: КНТУ 2012. – 454 с