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 забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF”

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

Керівник проекту
доктор філософії (PhD)
_____ Дрєєва Г.М.
«____» _____ 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
« 27 » лютого 2026 року

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

Руденку Денису Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF*

2. Керівник роботи *Дрєєва Ганна Миколаївна, 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 забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 27 » лютого 2026 р.

Підпис керівника
_____ Дрєєва Г.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 27 » лютого 2026 р.

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

АНОТАЦІЯ

Руденко Д.О. Програмне забезпечення системи забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF. 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 забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF.

Метою розробки є програмне забезпечення системи забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF.

Результат роботи – програмна реалізація системи забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF.

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, ЦОД, NVMe-oF

ABSTRACT

Rudenko D.O. Software for the system ensuring stable operation of a data center using the NVMe-oF protocol stack. 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 system ensuring stable operation of a data center using the NVMe-oF protocol stack.

The purpose of the development is software for the system ensuring stable operation of a data center using the NVMe-oF protocol stack.

The result of the work is a software implementation of the system ensuring stable operation of a data center using the NVMe-oF protocol stack.

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 PCs with Windows 10/11.

The program is developed in Python.

Keywords: computer engineering, data center, NVMe-oF

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП..... 3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 5
1.1 Призначення системи..... 5
1.2 Область застосування..... 5
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 7
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти..... 7
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 12
2.3 Розгорнута постановка завдання 14
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 16
3.1 Опис функціонування системи 16
3.2 Розробка структурної схеми..... 20
3.3 Розробка функціональної схеми 29
3.4 Розробка діаграми процесів..... 33
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 35
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 35
4.2 Захист розробленого програмного забезпечення..... 59
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 64
6 ОСНОВНІ ВИСНОВКИ..... 69
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 71

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

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

ВСТУП

Актуальність теми. За оцінками аналітиків, тільки 13% ємності зберігання перебуває усередині серверів або підключається до них прямо, що тоді як залишилися 87% розміщаються в зовнішніх системах зберігання. Відповідно, що випливає логічним кроком для NVMe, є реалізація NVMe поверх мережі. Із цією метою був розроблений набір протоколів NVMe over Fabrics, які дозволяють передавати команди NVMe по мережі й при цьому забезпечують порівнянні з NVMe затримки й інші характеристики

З поширенням твердотільних накопичувачів виникла потреба в більше ефективному протоколі доступу до них, тому що використовувані інтерфейси створювалися розраховуючи на жорсткі диски й не дозволяли повною мірою задіяти переваги нового різновиду пристроїв зберігання. У відповідь на цей запит галузь розробила протокол NVMe, що забезпечує підключення SSD-накопичувачів з використанням шини PCIe як транспорт і породжує набагато менші накладні витрати, чим його попередники.

Продажі жорстких дисків стабільно знижуються протягом от уже декількох років, у той час як попит на флеш-накопичувачі росте. Так, по оцінці сайту statista.com, в 2025 році на кожні два проданих жорсткі диски доводився один SSD (395 млн проти 190 млн), а всього за два роки до цього співвідношення було 4,5 до 1 (470 млн проти 105 млн). NVMe здобуває все більшу популярність як інтерфейс для SSD завдяки головним своїм перевагам: малій затримці при передачі даних, великої пропускної здатності й високих показників IOPS. Наприклад, якщо для SATA максимальне значення становить 200 000 IOPS, те, за даними організації NVM Express, для NVMe-пристроїв воно може перевищувати 1 000 000 IOPS.

Однак, по оцінках аналітиків, тільки 13% всієї ємності зберігання перебуває усередині серверів або підключається до них прямо, що тоді як

залишилися 87% розміщаються в зовнішніх системах зберігання. Відповідно, що випливає логічним кроком для NVMe є реалізація NVMe поверх мережі, як це було зроблено у свій час для SCSI (FCP, iSCSI). Із цією метою був розроблений протокол NVMe over Fabrics (NVMe-oF), що дозволяє передавати команди NVMe по мережі й одночасно забезпечує порівнянні з NVMe затримки й інші характеристики.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF.

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

– Огляд існуючих систем забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF.

– Дослідження системи забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF.

– Програмна реалізація системи забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF, 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

Система призначена для забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF. З точки зору операційної системи, диск, підключений через NVMe/TCP, та диск iSCSI представляють себе як необроблені блокові пристрої. Коли ви використовуєте вбудовані інструменти ОС для керування ними (iscsiadm для iSCSI та nvme для NVMe/TCP), їхня поведінка та набори функцій значною мірою порівнянні. Наприклад, прямі знімки на рівні сховища не підтримуються жодним із підходів.

Звітування про вільний та використаний простір ідентичне для дисків, підключених через будь-який протокол, а резервні копії PVE обробляються абсолютно однаково.

Ви також можете використовувати рівень QCOW2 поверх будь-якого типу диска. Основна відмінність полягає в інтеграції управління: тип пулу сховищ iscsi може бути вбудовано підтримуваний PVE, тоді як сховищем NVMe/TCP наразі потрібно керувати вручну.

На вищих рівнях управління, зокрема під час використання плагінів, що надаються постачальниками сховищ, ви можете розблокувати додаткові функції та можливості за допомогою будь-якого з цих варіантів.

1.2 Область застосування

Областю застосування є центри обробки даних. Дата-центр або ЦОД (Центр обробки даних) це високотехнологічне приміщення, простір якого заповнене телекомунікаційним устаткуванням і іншими пристроями, завдяки яким збір, зберігання й обробка різної інформації відкриває можливості для

користувачів у роботі всесвітньої мережі Інтернет. Відмітною рисою властивою стабільним дата-центрам є надійність будівлі, внутрішня планування й сучасна інфраструктура, що максимально відповідає вимогам стабільного й безпечного використання ЦОД його ключовими клієнтами.

Послуги дата-центра головним чином використовують корпоративні замовники, для успішного рішення завдань бізнесу, оскільки завдяки можливостям ЦОД, клієнти реально заощаджують фінанси компанії, оскільки немає необхідності обладнати спеціальне приміщення для зберігання серверного устаткування й наймати дорогий IT-персонал.

Розрізняють ЦОД насамперед по технічних можливостях і видам їхніх послуг. Корпоративні дата-центри, що індивідуально обслуговують інтереси великої компанії ділять пальму першості з комерційними центрами обробки даних, завдання яких полягає в обслуговуванні різних по своїх масштабах організацій і приватного бізнесу.

Кожний дата-центр має власний ряд інфраструктур, завдання кожної з яких сугубо індивідуальні:

– Інформаційна – забезпечує основні функції ЦОД, з яких виділимо обробку й зберігання ваших даних.

– Телекомунікаційна – забезпечує надійний зв'язок і передачу даних, між окремими серверами і їхніми користувачами.

– Інженерна інфраструктура – гарантує стабільне й безперебійне функціонування всіх систем серверної площадки.

На базі віртуальних платформ до не менш важливих можливостей функціонала ЦОД необхідно віднести ряд хмарних рішень, таких як: поштовий сервер, резервне копіювання даних (бекап) а також хмарні АТМ.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF, 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

Багато хто зіштовхуються із проблемою вибору – чи розміщати свої сервера на території України або ж звернутися до фірм, що пропонують закордонний колокейшн. У цьому випадку варто враховувати місце розташування основної клієнтської бази – якщо вона перебуває в інших країнах, те й сервер краще розмістити за рубежем. Якщо ж клієнти зосереджені на Україні, сервер також зручніше розташувати тут.

Устояні забобони схиляють велику кількість людей до необґрунтованої впевненості в тому, що закордонні варіанти розміщення сервера вигідніше й зручніше у всьому – у ціні, своєчасному і якісному технічному обслуговуванні, повній гарантії захисту від аварійних ситуацій і форс-мажорних обставин. На жаль, привабливі картини очікувань стрімко втрачають барвистість і вицвітають при першому ж знайомстві з реальністю – на практиці за рубежем клієнт одержує по-справжньому значну відмінність лише в розмірі абонентської плати при тому, що аптайм та інші характеристики залишаться приблизно на тому же рівні.

Як вибрати надійний центр обробки даних (ЦОД)

Навіть якщо клієнт зупинив вибір на одному із кращих ЦОД світу з бездоганною репутацією, зовсім незайвим буде просто відвідати його. А вже якщо договір укладається з маловідомим дата-центром, відвідування стає обов'язковим кроком. Можна просто поговорити із представником і простежити за його реакцією й відповідями на елементарні питання, але краще запитати співробітників про можливість потрапити безпосередньо в ЦОД. Якщо колектив відповідає на таке прохання категоричною відмовою й не бажає навіть видали

показати вам демонстраційну частину ЦОД, то варто задуматися – чи проста це міра безпеки, або ж реальне технічне оснащення дата-центра не збігається з обіцянками й умістом сертифікатів? Не варто радуватися й вільному входу в ЦОД – прямій доступ несумісний з поняттям схоронності інформації.

Нав'язлива стовідсоткова гарантія якості – ознака неправди завжди й скрізь

Шанс одержання неякісної послуги ніколи не зводиться до нуля – імовірність стати обманутим існує скрізь. Тільки відвідування й порівняння навіч обіцяного з фактичним може дати певні гарантії. Але вітчизняні клієнти найчастіше не відвідують закордонні дата-центри, цілком поклавшись на їхній фото, думки про їх і інтуїтивні відчуття. Використання послуг вітчизняних ЦОД дає значно більше гарантій – при бажанні можна без праці домовитися із представниками дата-центра про візит і особисто скласти власну неупереджену думку перед ухваленням остаточного рішення.

Якщо сервер перебуває за тисячі кілометрів від співробітників клієнта, то залишається покладатися тільки на безвідмовність його роботи – у випадку появи несправностей максимально швидке усунення поломки є попросту неможливим. Варто пам'ятати, що серед закордонних провайдерів лише одиниці мають україномовних співробітників у службі підтримки й з'являється ще одна перешкода – мовний бар'єр. Тому ймовірність співробітництва із закордонними провайдерами найкраще підкріплювати наявністю технічно грамотних співробітників, що володіють іноземними мовами. Тільки підписання угоди про рівень якості на значну суму може гарантувати швидке усунення виниклих несправностей, у противному випадку іноді обігу можуть навіть ігноруватися.

Не варто боятися понапрасну.

Масове переміщення серверів в іноземні ЦОД часто зв'язують із діяльністю українських державних органів контролю, які найчастіше можуть вилучати устаткування й перешкоджати роботі серверів по найменшому приводі. Але навіть за тисячі кілометрів від границь України неможливо одержати

стовідсоткову гарантію схоронності – віддалення серверів в інших країнах така ж звичайна процедура, просто за рубежем вона стане ледве більше складною й тривалою через подолання додаткових законодавчих бар'єрів. Якщо клієнт хоче одержати максимально можливу гарантію схоронності серверів у будь-якій країні, то його діяльність споконвічно повинна бути законної й прозорої.

Фінансове питання розміщення сервера за кордоном

Прийнявши остаточне рішення про розміщення сервера за кордоном, клієнт повинен бути готів переборювати додаткові труднощі, першої з яких стає покупка й доставка. Устаткування здобувається на місці установки й коштує дорожче або ж привозиться з іншої країни й обходиться дешевше, але додається значна сума за доставку й оформлення в митних органах. Варто пам'ятати, що закордонні ЦОД можуть відмовитися від розміщення серверів, які куплені на сумнівних ресурсах і не мають сертифікатів або ліцензій. Установлюючи таке устаткування, всю відповідальність і значно зрослі ризики клієнт бере на себе. Список витрат поповниться додатковою графою відрядної суми власного співробітника (поїздка в закордонний дата-центр із метою покупки устаткування) або величезної суми, що одержать співробітники ЦОД за виконання тої ж роботи.

Ще один спосіб – оренда сервера безпосередньо в даті-центрі. У цьому випадку додаткові витрати вносяться тільки за установку й настроювання, а основною складністю залишається зворотний зв'язок з іншомовною техпідтримкою, особливості якої унеможливлюють миттєву реакцію на виникнення несправностей. А поки іноземний фахівець відповість на запит і відправиться на усунення проблеми, сервер буде простоювати або виконувати свої функції не повною мірою.

Також бар'єром стають складності з бухгалтерською звітністю, адаптацією документації й підтвердженням проведення численних оплат, які можуть затримуватися внаслідок помилок електронних платіжних систем. У цьому випадку на допомогу приходять досвідчені посередники, послуги яких вимагають чергових додаткових витрат. І вибір посередника – не менш відповідальний крок,

адже серед таких фірм кількість шахраїв на порядок більше. Додатковою складністю стають рамки трафіку – у багатьох закордонних країнах не існує доступних безлімітних тарифних планів. А всі виниклі проблеми – додатковий простій сервера. Тільки якщо клієнт добре представляє ймовірність виникнення й має налагоджені способи врегулювання всіх вищевказаних проблем, можна сміло затверджувати, що іноземні ЦОД йому підходять.

Плюси розміщення сервера на Україні:

1. При яскраво вираженій орієнтації убік розміщених на Україні споживачів місцеві дата-центри можуть гарантувати канали з підвищеною надійністю й швидкістю.

2. Вартість. Ціна розміщення сервера на Україні набагато нижче, а поточний курс валют тільки підкріплює цю тенденцію.

3. Підтримка. Природно, на території України клієнт одержить допомогу зручною мовою.

4. Устаткування. Матеріально-технічна база для усунення будь-яких несправностей або швидкої заміни завжди перебуває під рукою клієнта.

5. Безкоштовний сервіс.

Плюси розміщення сервера в Європі:

1. Поліпшені канали при орієнтації на європейських споживачів.

2. Складна процедура віддалення сервера. Однак, варто пам'ятати, що якщо за кордоном винесли рішення про віддалення сервера, всі бар'єри лише ненадовго затримають відповідні органи, а не зупинять їх.

Остаточний вибір варто робити тільки після детального аналізу всіх існуючих переваг і недоліків вітчизняних і закордонних дата-центрів. Під час аналізу варто зберігати тверезість розуму й строгий розрахунок, а особливо важливо дистанціюватися від всіх забобонів і стереотипів про вітчизняний і закордонних дата-центрах, які можуть вплинути на остаточне рішення.

Уже давно пройшло той час, коли великі українські інтернет-проекти розташовувалися в дата-центрах Росії або Європи. Сьогодні в Україні з кожним

роком будується усе більше нових ЦОД, які укомплектовані по останньому слову техніки. У даному рейтингу ми постаралися зібрати найбільш велика дата-центри України, які мають широкий і сучасний набір послуг, а також мають гарні відкликання й багато клієнтів.

Mirohost.net

Компанія "Mirohost" працює на українському ринку вже майже 15 років, а з огляду на ще й загальний капітал із проектом "Imena.ua", вона є однією з найбільших українських хостинг-компаний. Устаткування дуже стабільно й відповідає найвищим світовим стандартам. Для кожного клієнта, що замовляє хостинг строком на 1 рік, даний провайдер безкоштовно реєструє домен. Служба підтримки працює добре, але Livechat, на жаль, відсутній. Ціни в даного провайдеру звичайно не найнижчі, але така якість коштує цих грошей.

Freehost.com.ua

Freehost" – це один з найбільших провайдерів на українському ринку хостингу. Дана компанія має досить надійне й сучасне устаткування. Був період, коли з'явилися скарги на швидкість виконання php-скриптів, але вони припинилися. Так само наші тестери відзначають інтерфейс, що змінився, що став більше зручним, але інформація подана менш зрозуміло, чим в інших провайдерів.

Colocall.net

Компанія "Дзвін", що надає послуги під брендом "Colocall" є старожилом українського хостинг-ринка. На сьогодні має дуже велику базу клієнтів і пропорційно зростаюча кількість послуг. Негативним моментом експерти порахували високі ціни.

Prohoster.info

Компанія надає безліч різних пакетів віртуального й VPS хостингу. Захист серверів віртуального хостингу на дуже високому рівні. Також співвідношення ціни і якості в VPS серверів відмінне, та й техпідтримка досить кваліфікована. З іншого боку, виділені сервера слабенькі й забезпечувати безпека як виділених,

так і VPS серверів потрібно самостійно. Тому, можна сказати, що віртуальний хостинг підійде будь-якому користувачеві, а VPS і виділені сервери більше пасують професіоналам, тому що для їхнього адміністрування потрібні деякі пізнання в цій області.

Fornex.com

Навігація по сайті зручна й знайти тарифи хостингу дуже легко – посилання на них відразу впадає в око. Відразу хочеться відзначити те, що хостер докладно описав які саме послуги входять у кожний тарифний план. Вартість же хостингу стартує від 1,14 долара на місяць, що доступна фактично кожному користувачеві із самим маленьким не прибутковим проектом. Також хостинг-провайдер за додаткову плату надає інші досить корисні послуги, які не всі провайдери можуть вам запропонувати (базове адміністрування VPS, наприклад).

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

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

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

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

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

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

Python підтримує динамічну типізацію, тобто тип змінної визначається тільки під час виконання. Тому замість «присвоювання значення змінної» краще говорити про «зв'язування значення з деяким ім'ям». У Python є убудовані типи: бульові, рядки, Unicode-рядки, цілі числа довільної точності, числа із плаваючої коми, комплексні числа й деякі інші. З колекцій Python підтримує кортежі

(*tuples*), списки, словники (асоціативні масиви) і, починаючи з версії 2.4, безлічі. Всі значення в Python є об'єктами, у тому числі функції, методи, модулі, класи.

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

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

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи забезпечення стабільної роботи ЦОД за

допомогою стеку протоколів NVMe-oF.

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

Fibre Channel

Fibre Channel (або скорочено FC, у перекладі з англійського fibre channel – волоконний канал) – клас протоколів, що використовуються для високошвидкісної (гігабітної) передачі даних. Застосовується в основному для мереж зберігання даних. Перебуває у відомстві Технічного комітету T11 (входить до складу Міжнародного комітету зі стандартам у сфері IT (InterNational Committee for Information Technology Standards – INCITS).

Первісне сімейство Fibre Channel застосовувалося винятково в суперкомп'ютерах, потім перейшло в мережі зберігання даних, там FC застосовується як стандартна можливість підключення до система зберігання даних. Одержав широке корпоративне використання.

Fibre Channel Protocol (FCP) – це транспортний протокол (аналогічно TCP в IP-мережах), що використовує SCSI у мережах FC. Являє собою базу для побудови мереж зберігання даних.

Fibre Channel уперше з'явився в 1988 році, але тільки в 1994 році був офіційно затверджений як стандарт, що спрощує інтерфейс HIPPI. У застарілому протоколі застосовувався масивний 50-парний кабель із громіздкими коннекторами.

Первісними цілями інтерфейсу Fibre Channel по виходу на ринок були підвищення дальності дії й спрощення схеми підключення ліній передачі. Примітно, що перед розроблювачами не стояло завдання збільшити швидкість роботи.

Коли Fibre Channel почав вести серйозну конкуренцію на ринку, головним суперником нового інтерфейсу став SSA (Serial Storage Architecture) від IBM, з

яким довгий час розверталася серйозна боротьба. Зрештою, ринок вибрав Fibre Channel, таким чином, IBM втратили контроль над технологіями зберігання даних наступного покоління.

Чому покупець вибрав Fibre Channel?

Справа в тому, що в процесі розробки інтерфейсу, основна концентрація зусиль довелася на спрощення підключення, збільшення відстані дії, однак і не був зроблений акцент на збільшення швидкості.

Пізніше, розроблювачі додали до пріоритетних завдань по розвитку FC ще й можливість підключення дискових накопичувачів SCSI, що забезпечують більше високу швидкість при збільшеному числі підключених пристроїв. Це зробило Fibre Channel практично недосяжним навіть для великих конкурентів. Крім того, пізніше в Fiber Channel була додана підтримка будь-якої кількості протоколів "верхнього рівня" (включаючи ATM і IP) з SCSI, що виступає як пріоритетний.

RDMA: погляд зсередини

Ріст популярності кластерних систем як середовища для високо-продуктивних обчислень або HPC (High Performance Computing) виводить на перший план завдання забезпечення ефективної взаємодії платформ, що утворять кластер.

Провідне місце тут займає технологія RDMA (Remote Direct Memory Access), що узагальнює поняття прямого доступу до пам'яті від передачі даних у межах локальної платформи до взаємодії декількох систем у межах кластера.

Щоб зрозуміти зміст цієї технології, спробуємо розглянути контролер NIC (Network Interface Controller) поза контекстом мережних протоколів (TCP), винятково як машину, що вміє виконувати дві дії:

– приймати дані з мережі й передавати їх в оперативну пам'ять;
– читати дані з оперативної пам'яті й передавати їх у мережу.

Тоді, для декількох комп'ютерів, зв'язаних у мережу, сукупність їхніх контролерів NIC можна розглядати як якийсь узагальнений контролер DMA, що відрізняється від звичайного контролера DMA тим, що здатно взаємодіяти не тільки з локальною, але й з віддаленою пам'яттю.

Як працює DMA у межах кластера

Отже, ніж же потрібно доповнити традиційну операцію копіювання блоку пам'яті, якщо джерело й одержувач такого копіювання перебувають на різних платформах одного кластера.

Запит містить інформацію ідентифікуючий буфер-джерело й адреса всередині нього, буфер-одержувач і адреса всередині нього, а також довжину переданого блоку. Передача такого повного набору параметрів є властивістю тегованих запитів, що адресують конкретні діапазони адрес у конкретних буферах:

– Data Sink Steering Tag [32 bits] – номер, під яким у межах кластера зареєстрований буфер-одержувач.

– Data Sink Tagged Offset [64 bits] – зсув початку записуваного блоку відносно почала буфера-одержувача.

– RDMA Read Message Size [32 bits] – розмір блоку, що пересилається.

– Data Source Steering Tag [32 bits] – номер, під яким у межах кластера зареєстрований буфер-джерело.

– Data Source Tagged Offset [64 bits] – зсув початку блоку, що читається, відносно початку буфера-джерела.

Такий набір з п'яти параметрів відрізняється від звичного канонічного набору трьох параметрів операції копіювання (адреса джерела, адреса одержувача, довжина блоку) тільки тим, що поряд з використанням 64-бітних адрес (зсувів) використовуються номери буферів (теги). Причина такого ускладнення очевидна: взаємодіють кілька платформ, у кожної платформи свій адресний простір, тому для ідентифікації будь-якого об'єкта, знання однієї тільки адреси недостатньо.

Буфери, що є логічно безперервними, можуть бути фрагментованими у фізичних адресних просторах платформ. Це треба з логіки роботи віртуальної пам'яті в операційній системі.64-бітні зсуви Tagged Offsets задають не фізичні, а логічні (віртуальні) адреси, які необхідно піддати сторінкової трансляції й дефрагментації за допомогою пристроїв Scatter-Gather, щоб, в остаточному підсумку, об'єктом RDMA-операцій контролера RNIC стала пам'ять користувальницьких додатків, як це пропонує модель передачі даних Direct Data Placement (DDP).

Очевидно, технологія RDMA, також як і традиційний стік мережних протоколів, забезпечує передачу даних між обчислювальними платформами. Чим же вона краще традиційного підходу до обміну інформацією з локальної мережі?

У випадку використання тегованих запитів, кожна із платформ, що утворять кластер має інформацію, необхідної для дистанційної адресації пам'яті іншої платформи. Роздільні фізично адресні простори систем, що утворять кластер, можуть бути об'єднані в єдинологічний адресний простір.

Схема адресації, використовувана драйверами RDMA, гарантує, що на момент ініціювання операції передачі даних, відомі кінцеві цільові адреси як буфера-джерела так і буфера-одержувача. Обоє цих буфери перебувають в адресному просторі користувальницьких додатків. Така форма організації передачі даних, що дозволяє усунути необхідність у транзитних буферах і додаткових операціях копіювання даних, називається Zero-Copy. Нескладно помітити, що необхідність втручання центрального процесора в ході виконання такої операції мінімізована, тому що вся інформація, що задає базові адреси й розміри копіюємих блоків, програмується на стороні, що генерує запит у момент його генерації.

3.2 Розробка структурної схеми

NVMe over Fabrics являє собою мережний протокол, використовуваний для взаємодії між хостом і системою зберігання по мережі (мережна фабрика). Як найближчі аналоги можна привести FCP і iSCSI, однак на відміну від них він забезпечує набагато меншу затримку – кілька мікросекунд. У результаті різниця між локальним і віддаленим сховищем стає практично непомітною.

Розробка NVMe-oF почалася в 2014 році з метою поширення підтримки протоколу блокового доступу NVMe на мережні середовища Ethernet, Fibre Channel і InfiniBand. З його допомогою планувалося збільшити відстань до підключених пристроїв і підсистем NVMe, довівши його до типових для ЦОД дистанцій, а крім того, забезпечити додаткову затримку при доступі по мережі не більше 10 мкс у порівнянні з локальними пристроями NVMe усередині сервера.

Перша специфікація NVMe-oF версії 1.0 була прийнята в 2016 році й передбачає використання Fibre Channel і RDMA як транспорт. У випадку RDMA використовується одна із трьох технологій – InfiniBand, RDMA over Converged Ethernet (RoCE) і Internet Wide Area RDMA Protocol (iWARP). У версії 1.1, що повинна з'явитися в другій половині цього року, очікується також підтримка TCP. Крім цих мережних середовищ, у майбутньому можлива підтримка мережних технологій наступного покоління Gen-Z.

NVMe-o забезпечує прив'язку NVMe до мережного середовища, використовуваної як транспорт. Прив'язка (binding) покликана гарантувати надійну передачу даних, команд і відповідей між хостом і підсистемою NVMe по мережному середовищу з мінімальною додатковою обробкою. Одні середовища передачі краще підходять для окремих додатків, інші гірше, тому прив'язка до конкретного середовища може виключати або обмежувати ті або інші функції залежно від можливості транспорту.

Невеликий набір команд NVMe спрощує реалізацію протоколу в мережному середовищі. Приблизно на 90% протокол NVMe-oF збігається з

локальним протоколом NVMe: ті ж команди уведення-виводу й адміністративні команди, такі ж адресні простори (простору імен – у термінології NVMe), регістри й властивості, асинхронні події й т.д. Основні розходження стосуються використовуваних ідентифікаторів, механізмів виявлення, реалізації черг і передачі даних. Однак ці подробиці цікаві головним чином розроблювачам драйверів.

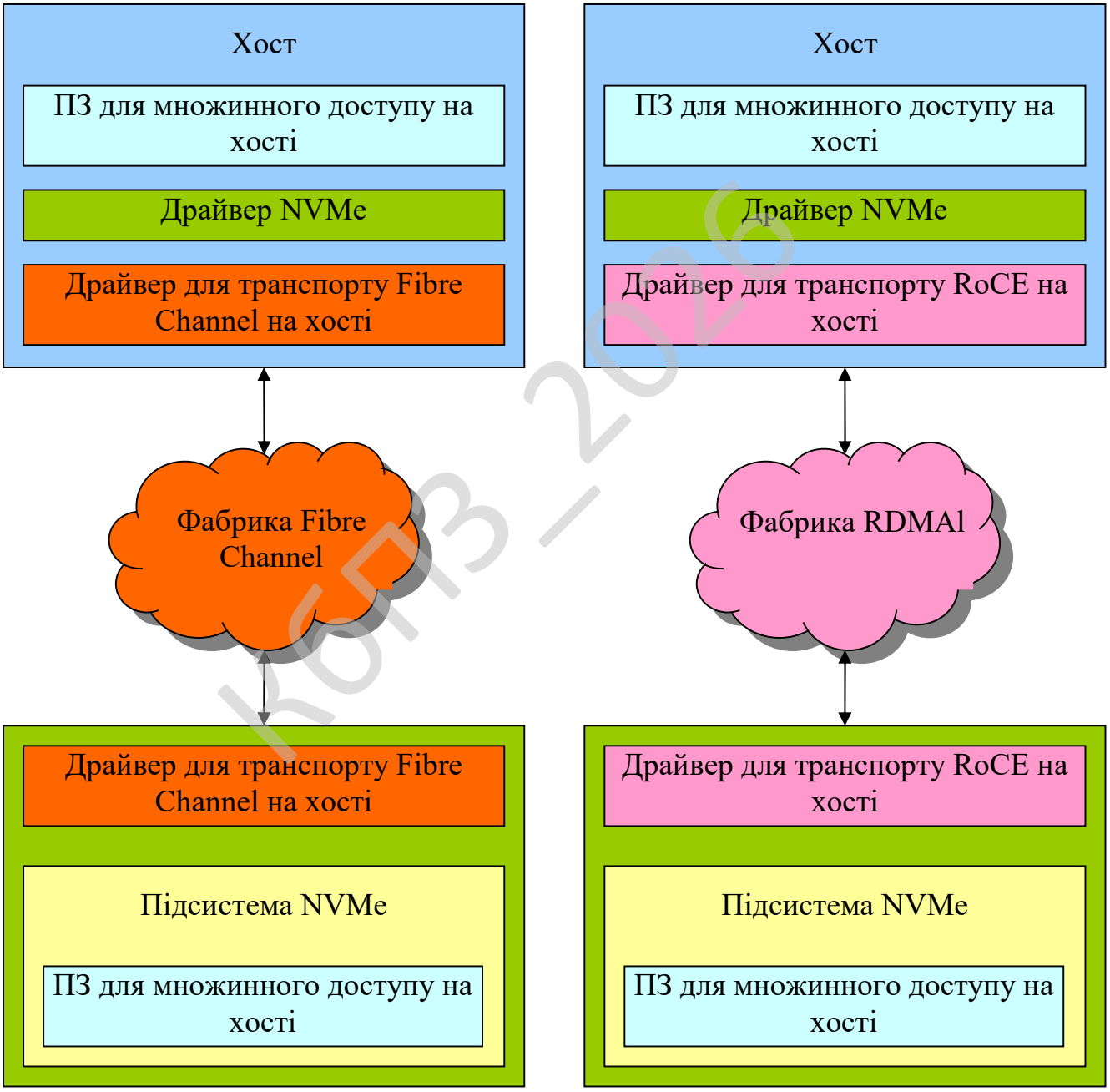

Рисунок 3.1 – Структурна схема системи

Головна відмінність NVMe-oF від NVMe полягає в механізмі передачі команд. NVMe записує запити й відповіді в поділювану область пам'яті на хост-комп'ютері за допомогою PCIe, тоді як NVMe-oF реалізує обмін повідомленнями для передачі запитів і відповідей по мережі між хостом і цільовим пристроєм зберігання. Для транспорту через мережу команди й дані містяться в так звані капсули, кожна з яких може вміщати кілька команд або відповідей. Формат капсул не залежить від того, яка конкретно мережне середовище використовується.

За допомогою капсул можна відправити кілька невеликих повідомлень в одному, що дозволяє поліпшити ефективність передачі й скоротити затримку. Капсули містять запису із чергами запитів (Submission Queue Entry) або відповідей (Completion Queue Entry), а також деякий обсяг даних, метаданих або списків фрагментації-дефрагментації (Scatter-Gather List, SGL).

Для ідентифікації віддалених цільових пристроїв використовуються звичні угоди про кваліфіковані імена. Схема NVMe Qualified Name (NQN) аналогічна iSCSI Qualified Name (IQN). У березні 2018 року NVM Express додав до протоколу NVMe-oF нову функцію за назвою «асиметричний доступ до простору імен» (ANA) і тим самим забезпечив можливість організації уведення-виводу по різних маршрутах між безліччю хостів і просторами імен на цільових пристроях.

NVMe-o висуває тверді вимоги до мережного середовища для доставки команд NVMe. Як вказується в інформаційному документі NVM Express, мережна фабрика повинна вносити затримку не більше 10 мкс, масштабуватися до десятків тисяч пристроїв, надавати альтернативні маршрути між хостом і цільовим пристроєм і т.і. Від мережі потрібне забезпечення надійної доставки даних без втрати кадрів (пакетів) через перевантаження, а від мережних адаптерів – здатність самостійно резервувати області пам'яті для використання їхніми додатками, щоб дані уведення-виводу могли бути передані прямо апаратному адаптеру (нульове копіювання – zero copy). Все це порушує питання про вибір мережного середовища, найбільш підходящої для реалізації NVMe-oF.

NVMe поверх Fibre Channel

Мережа Fibre Channel використовується переважно для передачі протоколу SCSI. Властиво, Fibre Channel Protocol (FCP) являє собою верхнерівневий протокол для передачі SCSI по Fibre Channel. Однак у мережі FC можуть застосовуватися й інші протоколи, зокрема FICON, а тепер до них додається ще й NVMe over Fibre Channel (FC-NVMe).

Кращим середовищем для реалізації NVMe-oF є мережа з контролем потоків на основі кредитів для забезпечення передачі пакетів без втрат. На відміну від Ethernet, де для цього необхідні додаткові засоби, такі як DCB, Fibre Channel споконвічно підтримує відповідні убудовані механізми. Завдяки їм відправник завжди знає, який кредит йому виділений, тобто скільки пакетів може обробити одержувач, що дозволяє уникнути втрат «зайвих» пакетів.

За реалізацію FC-NVMe відповідає комітет T11 Міжнародної комісії по IT-стандартах (International Committee for Information Technology Standards, INCITS), що займається розробкою стандартів для Fibre Channel. Крім властиво FC, дана реалізація може працювати й по Ethernet – точніше, по Fibre Channel over Ethernet (FCoE). Роботи над першою версією стандарту FC-NVMe були завершені в серпні 2017 року. При розробці другої версії стандарту, FC-NVMe-2, комітет T11 має намір сфокусуватися на таких моментах, як корекція послідовних помилок і арбітраж черг.

Розроблювачі приділяли велику увагу скороченню числа непотрібних повідомлень і параметрів при транспорті NVMe через мережу зберігання Fibre Channel з виконанням всіх вимог специфікації NVMe-oF. Існуючі мережі зберігання на базі FC п'ятого й шостого поколінь можуть підтримувати FC-NVMe з мінімальними необхідними змінами. Зокрема, FC-NVMe повинен підтримуватися операційною системою СЗД і мережними адаптерами.

Завдяки скороченому набору команд і підтримці безлічі черг, FC-NVMe забезпечує кращу продуктивність і меншу затримку (у порівнянні з SCSI), а також паралельне уведення-вивід при читанні й записі даних на SSD. Як показали

тести, проведені аналітичною компанією Demartek, що спеціалізується на дослідженнях рішень для зберігання даних і мережних технологій, в NVMe over Fibre Channel показник IOPS на 58% вище, а тривалість затримки на 34% менше, ніж у протоколу SCSI Fibre Channel.

Fibre Channel залишається де-факто стандартом для мережного доступу до пристроїв зберігання: по даним IDC, за допомогою FC здійснюється доступ до більш ніж 20 тис. петабайт даних – ніяка інша технологія щодо цього з нею не зрівняється. Головна практична перевага FC-NVMe у порівнянні з альтернативами полягає в тому, що для нього не потрібно радикальної модернізації мережі: підтримка NVMe у мережі Fibre Channel на базі устаткування 5-го й 6-го покоління досягається відновленням програмного забезпечення. При цьому до однієї й тієї же інфраструктурі Fibre Channel можна підключати системи зберігання з підтримкою як FCP, так і FC-NVMe, чим забезпечується поступовий перехід на нову технологію.

NVMe поверх RDMA

Альтернативою Fibre Channel є використання RDMA. У свою чергу, RDMA як транспорт може задіяти InfiniBand, Ethernet (точніше, конвергентний Ethernet – RoCE) і протокол TCP (iWARP).

Метод прямого доступу до пам'яті (Remote Direct Memory Access, RDMA) одержав широке поширення у високопродуктивних обчисленнях, де спочатку він був реалізований поверх InfiniBand. За допомогою RDMA комп'ютери в мережі можуть обмінюватися даними, що перебувають в основній пам'яті, без участі процесора, кешу й операційної системи. Виключення ЦПУ із процесу переміщення даних дозволяє істотно скоротити затримку й підвищити продуктивність і ефективність обміну.

Як і FC-NVMe, Infiniband забезпечує передачу даних без втрат, передбачає убудовані механізми підтримки необхідної якості обслуговування й реалізує контроль потоків на базі кредитів. Однак Infiniband має порівняно вузьку область застосування – переважно в HPC і серверних кластерах. Для його впровадження

більшості підприємств прийдеться переходити на нову технологію, що обмежує потенційну область поширення NVMe over InfiniBand.

Протокол Ethernet – найбільше широко використовуваний протокол у сучасних центрах обробки даних. З появою усе більше швидких варіантів Ethernet зростає потреба в протоколах з низькою затримкою, таких як RDMA. Завдяки версії RoCE для конвергентного Ethernet технологія RDMA стала використовуватися в хмарах, корпоративних мережах і мережах зберігання. RoCE був розроблений InfiniBand Trade Association і іноді розглядається як реалізація Infiniband поверх Ethernet.

Теоретично NVMe-oF на базі Ethernet може бути реалізований із застосуванням будь-яких комутаторів Ethernet, однак насправді для цього необхідно устаткування, що здатне ефективно справлятися з такого роду трафіком. Щоб використовувати RoCE, потрібні комутатори з підтримкою протоколу Data Center Bridging (DCB), які забезпечують передачу кадрів без втрат і контроль пріоритетних потоків (Priority Flow Control, PFC).

На даний момент є дві версії RoCE, причому між собою вони несумісні через використання різних транспортних механізмів. RoCEv2 є маршрутизуюємою версією і як транспорт використовує протокол UDP, а не Ethernet. Однак з нею, як і у випадку RoCEv1 безпосередньо, рекомендується використовувати Converged Ethernet, тобто DCB.

Варіант iWARP одержав значно менше поширення, хоча був запропонований більше 10 років тому. Стандартизований в 2007 році Internet Engineering Task Force, він базується на TCP/IP, опирається на стандартні механізми розвантаження TCP, підтримує маршрутизацію пакетів і може використовуватися поверх будь-якої мережі Ethernet. Однак, як показують порівняльні тестування RoCE і iWARP на швидкостях 10 і 40 Гбіт/с, перший забезпечує меншу затримку й більшу пропускну здатність для повідомлень всіх розмірів. Такий результат не викликає подиву, оскільки у випадку iWARP додається як мінімум ще один проміжний рівень.

Використання TCP обмежує застосовність iWARP (безвідносно NVMe). У своєму вихідному варіанті TCP погано підходить для передачі трафіку від систем зберігання через використання механізмів повільного старту. Останні версії TCP, наприклад TCP для центрів обробки даних (Data Center TCP, DCTCP), мають спеціальні функції для оптимізації роботи в ЦОД, однак DCTCP не застосовується в глобальних мережах. Необхідність задіяти різні стеки протоколів TCP/IP для різних сценаріїв використання стримує впровадження iWARP.

Fibre Channel по TCP

Впровадження NVMe-oF у випадку RoCE і iWARP може потенційно зажадати заміни всіх мережних серверних карт на плати з підтримкою RDMA. Настільки масштабна модернізація навряд чи економічно оправдана, особливо для таких гігантів, як Facebook. Тому власники великих ЦОД готові погодитися з деяким збільшенням затримки заради можливості збереження існуючої інфраструктури. Це породило інтерес до реалізації NVMe поверх TCP з використанням типової мережі Ethernet.

Протокол TCP с необхідними вдосконаленнями дозволяє забезпечити цілком прийнятні затримки, при цьому немає необхідності в застосуванні спеціальних мережних плат і комутаторів з підтримкою DCB. Компанія Pavilion, виробник флеш-масивів з підтримкою NVMe-oF, зрівняла характеристики NVMe-oF, коли як транспорт використовувалися RoCE і TCP. При доступі до сховища від 1 до 20 клієнтів середня затримка запису у випадку TCP склала 183 мкс, а у випадку RoCE – 107 мкс.

Таким чином, TCP-NVMe виявився на 71% повільніше. Однак і така затримка була у два рази менше, ніж при використанні iSCSI у тій же конфігурації. Ці результати цілком прийнятні, тим більше що тестування проводилося в масштабі центра обробки даних, а не окремої стійки (між клієнтом і СЗД могло перебувати до шести транзитних комутаторів). Після настроювання затримку вдалося скоротити ще більше.

Компанія Solarflare Communications, постачальник устаткування Ethernet, розробила безкоштовний плагін для стека TCP/IP на Linux з підтримкою NVMeo/TCP (він може працювати як на клієнті, так і на цільовій системі зберігання). Після схвалення стандарту на TCP-NVMe цей плагін буде доданий у ядро Linux. Таким чином, будь-який бажаючий зможе створити систему зберігання з доступом по NVMe-oF практично з будь-якого місця.

По даним SolarFlare, при використанні плагіну затримки збільшуються всього на 5-10% у порівнянні з використанням NVMe-oF на базі RDMA, при цьому не виникає проблем із сумісністю устаткування. (Як відзначалося вище, RoCE і iWARP несумісне, тому на обох кінцях з'єднання повинні бути встановлені однакові мережні карти. Поки тільки компанія Marvell оголосила про випуск NIC з одночасною підтримкою RoCE і iWARP.)

Поки просуванням NVMe over TCP займаються переважно стартапи, серед яких Pavilion, Lightbits і Solarflare, але інтерес до даної технології проявляють і великі гравці. Так, ізраїльський стартап Lightbits заявляє, що він працює с Intel і Facebook над специфікацією NVMe over TCP. А програмне забезпечення Kumoscale компанії Toshiba підтримує нестандартизовану версію протоколу.

Прив'язки до транспорту TCP планується додати в стандарти NVMe поряд із прив'язками до Fibre Channel і RDMA, що дозволить використовувати NVMe-oF в існуючих мережах ЦОД. Цим займається технічна робоча група NVMe (NVMe Technical Working Group). Прийняття стандарту на NVMe поверх TCP очікується до кінця поточного року. Він буде підтримувати всі функції й архітектурні особливості NVMe і NVMe-oF.

Стандартизація NVMe over TCP повинна спростити (і здешевити) реалізацію NVMe-oF в існуючих мережах Ethernet. Даний транспорт для NVMe представляє цілком життєздатну альтернативу іншим різновидам, якщо тільки не потрібно забезпечити максимально низьку затримку.

Перспективи NVMe-oF

NVMe over Fabrics робить тільки перші кроки – навіть не всі стандарти ще прийняті. В 2017 році підтримка NVMe-oF була додана в ядро Linux і з'явилася в

цілому ряді дистрибутивів цієї ОС. Однак на швидке й тим більше широке поширення NVMe-oF навряд чи можна розраховувати. Наднизька затримка потрібна далеко не всім додаткам, так що спочатку NVMe-oF буде реалізовуватися тільки для тих додатків, для яких це дає реальну вигоду,?- наприклад, для аналітики в реальному часі, біржової торгівлі, машинного навчання.

Це переважно висококритичні застосунки, але впровадження такої технології, що ще не пройшла серйозне обкатування, чревате підвищеними ризиками. У кожного виробника свій підхід до реалізації NVMe-oF у тих або інших рішеннях, що потенційно може привести до проблем сумісності. До того ж побудова системи, що повною мірою використовувала б можливості NVMe-oF, поки неможливо з технічних причин – всі переваги NVMe-oF можна буде задіяти з появою PCIe 4.0.

4-портові накопичувачі NVMe здатні приймати дані зі швидкістю понад 20 Гбіт/с при записі й видавати зі швидкістю близько 25 Гбіт/с при читанні. Щоб підтримувати такий темп, для кожного накопичувача в системі потрібно встановити адаптер 25Gb або 32GFC і виділити на хості по 8 ліній PCIe. Побудова системи без вузьких місць і перепідписки – непросте завдання.

Підтримка різних варіантів транспорту для NVMe-oF, з одного боку, надає волю вибору, з іншого боку, порушує питання про кращий вибір. Прихильники RDMA упевнені, що саме ця технологія буде найбільш затребуваний як спосіб реалізації NVMe-oF. Так, згідно із прогнозами G2M Research, у найближчі два-три роки у світі буде продаватися в три рази більше Ethernet-адаптерів з підтримкою NVMe-oF, чим всіх інших. У той же час прихильники FC-NVMe, заявляючи про необґрунтованість таких прогнозів, пророкують NVMe-oF на базі Ethernet таку ж долю, яка осягла технологію FCoE.

Fibre Channel залишається кращою технологією для реалізації мереж зберігання. І Gartner, і IDC згодні з тим, що основна частка (70–80%) флеш-масивів підключаються до FC SAN. Це переважно мережі Fibre Channel 5-го або

6-го покоління, які вже готові до передачі трафіку NVMe-oF на відміну від звичайних мереж Ethernet, де для підтримки NVMe-oF потрібна модернізація апаратного забезпечення. До того ж, роблячи ставку на Ethernet, потім прийде вибирати між двома несумісними варіантами – RoCE і iWARP, а точніше, трьома, оскільки між RoCEv1 і RoCEv2 сумісності теж немає.

Із прийняттям стандарту NVMe over TCP стане доступний дешевий і простий спосіб реалізації NVMe-oF по Ethernet.

3.3 Розробка функціональної схеми

Функціональна схема системи зображена на рисунку 3.2. З рисунку видно, що розроблена система складається з наступних блоків:

– Моніторинг трафіку ЦОД за допомогою стеку протоколів NVMe-oF.

– Робота з файлами.

– Монітор з'єднань.

– Статистика подій.

– Робота з ресурсами мережі ЦОД за допомогою стеку протоколів NVMe-oF.

– Робота з сесіями.

– Функції для роботи з мережею ЦОД за допомогою стеку протоколів NVMe-oF.

Розглянемо детальніше кожний з блоків.

Моніторинг трафіку ЦОД за допомогою стеку протоколів NVMe-oF включає в себе:

– Визначення підключених інтерфейсів.

– Визначення вхідного та вихідного трафіку ЦОД за допомогою стеку протоколів NVMe-oF.

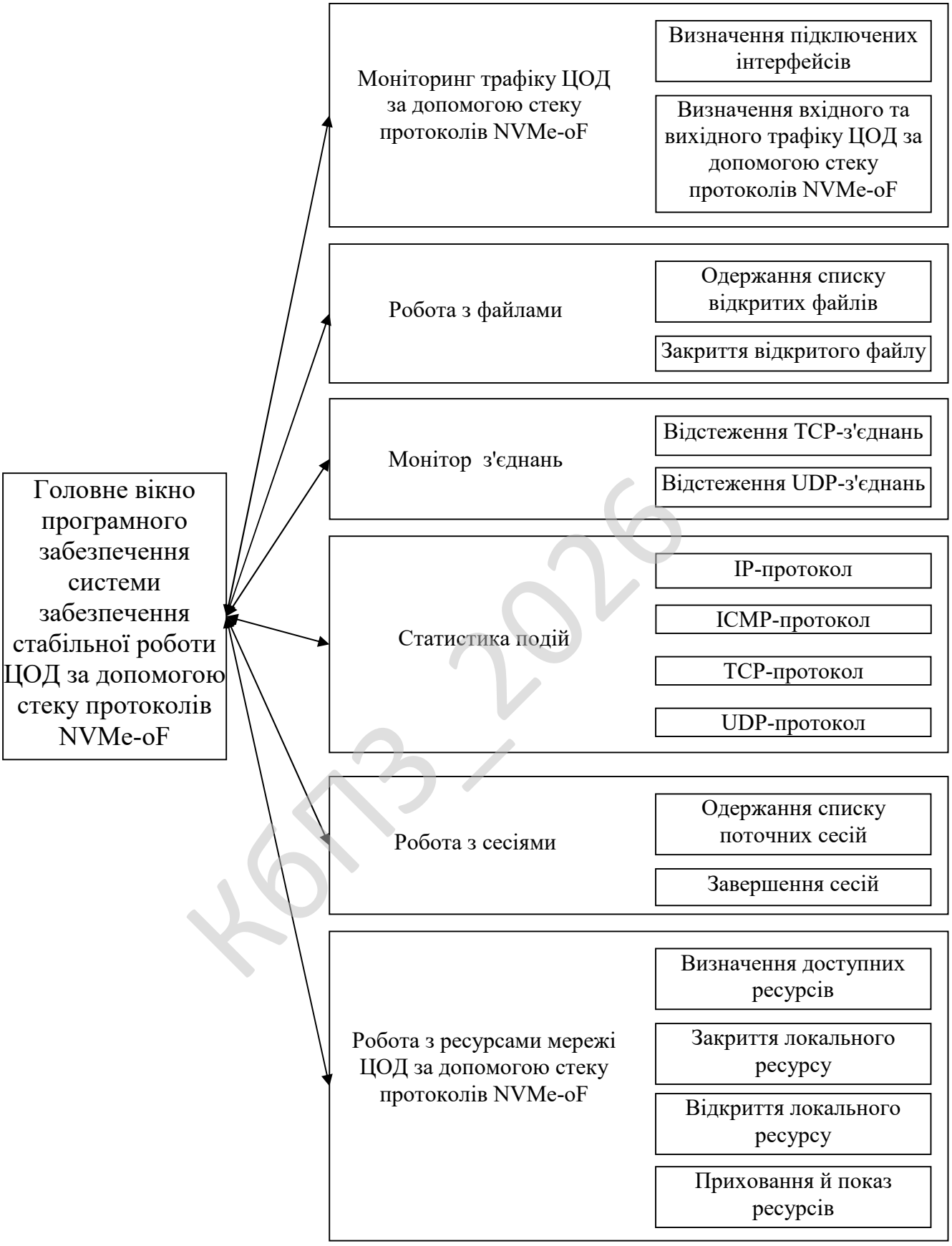

Рисунок 3.2 – Функціональна схема системи

Розроблена система відображає всі інтерфейси приєднані до комп'ютера, на якому запущена програма, їх MAC-адреси та вхідний і вихідний трафік на кожному з них.

Робота з файлами включає в себе:

– Одержання списку відкритих файлів.

– Закриття відкритого файлу.

Можна переглянути, які з Ваших файлів, що Ви відкрили для загального доступу, переглядають по мережі ЦОД за допомогою стеку протоколів NVMe-oF. Програма відобразить список файлів та користувачів, які їх переглядають. Також можна відкрити чи закрити файл.

Монітор з'єднань включає в себе:

– Відстеження TCP- з'єднань.

– Відстеження UDP- з'єднань.

Система фіксує всі підключення по TCP- та UDP-протоколу, та виводить їх на екран у форматі *IP-адреса:порт_призначення*.

Статистика подій включає в себе відстеження подій в наступних протоколах:

– IP-протокол.

– ICMP-протокол.

– TCP-протокол.

– UDP-протокол.

Статистика ведеться по цілому ряду параметрів. Наприклад для TCP-протоколу фіксується: Тип алгоритму повторної передачі, мінімальний тайм-аут, максимальний тайм-аут, максимальна кількість помилок з'єднання, активні з'єднання, пасивні з'єднання, невдалі спроби відкриття, скидання встановлених з'єднань, отримані сегменти, надіслані сегменти, повторно передані сегменти, помилки тощо.

Робота з ресурсами мережі ЦОД за допомогою стеку протоколів NVMe-oF включає в себе:

– Визначення доступних ресурсів ЦОД за допомогою стеку протоколів NVMe-oF.

– Закриття локального ресурсу ЦОД за допомогою стеку протоколів NVMe-oF.

– Відкриття локального ресурсу ЦОД за допомогою стеку протоколів NVMe-oF.

– Приховання й показ ресурсів ЦОД за допомогою стеку протоколів NVMe-oF.

Система відображає наявні у мережі ЦОД за допомогою стеку протоколів NVMe-oF ресурси у вигляді дерева. Пошук ресурсів ЦОД за допомогою стеку протоколів NVMe-oF можна здійснювати по заданим умовам: локальні чи глобальні ресурси; всі ресурси, тільки файли, чи тільки принтери, тощо.

Можна додавати до загальних ресурсів мережі ЦОД за допомогою стеку протоколів NVMe-oF свої власні, а також закривати їх потім.

Робота з сесіями включає в себе:

– Одержання списку поточних сесій.

– Завершення сесій.

Програма дозволяє переглянути список відкритих сесій, що включає в себе: назву сесії, користувача, що її розпочав, номер сесії, час роботи та час очікування.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

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.

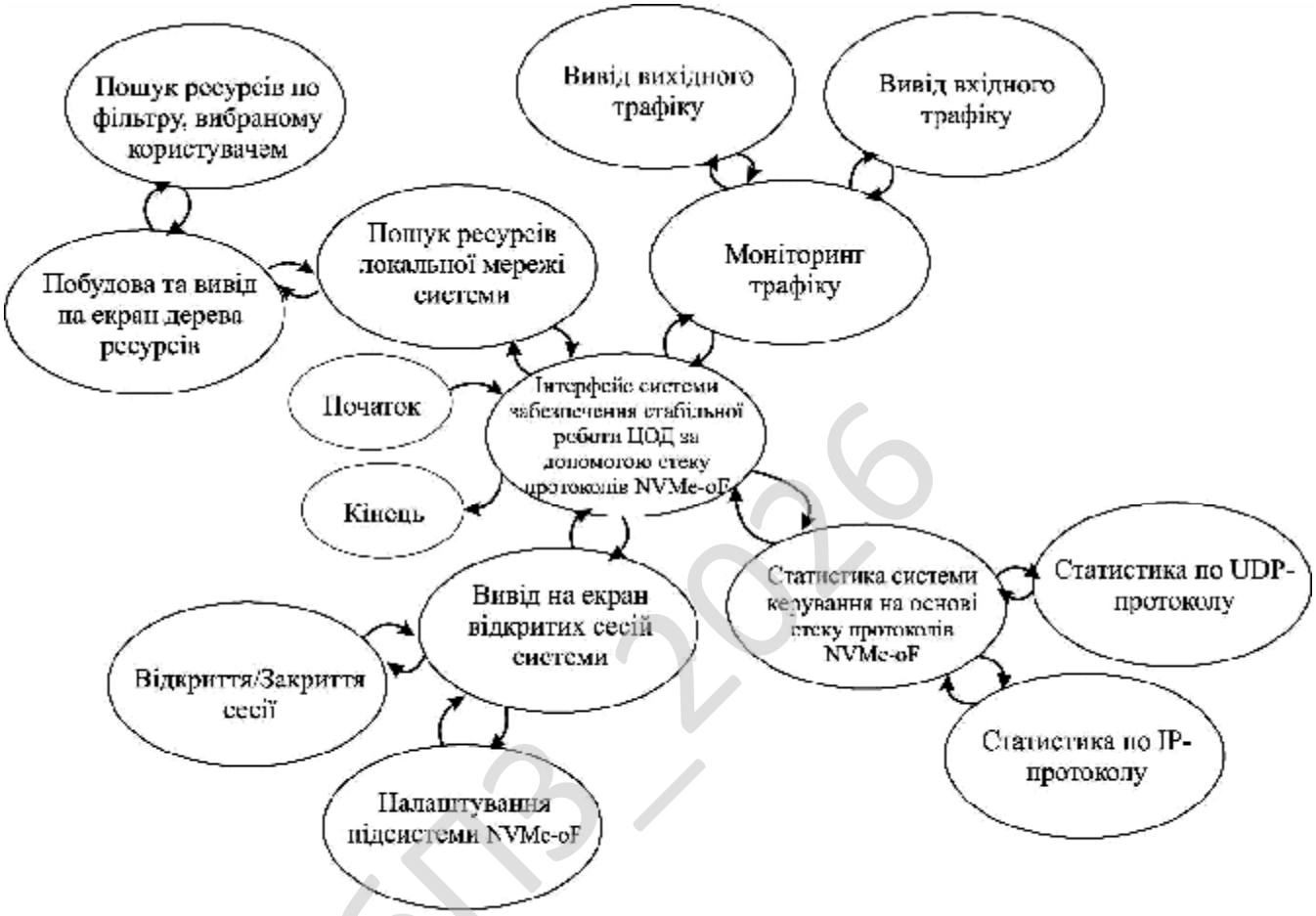

Рисунок 3.3 – Діаграма взаємодії процесів

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).

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.

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.

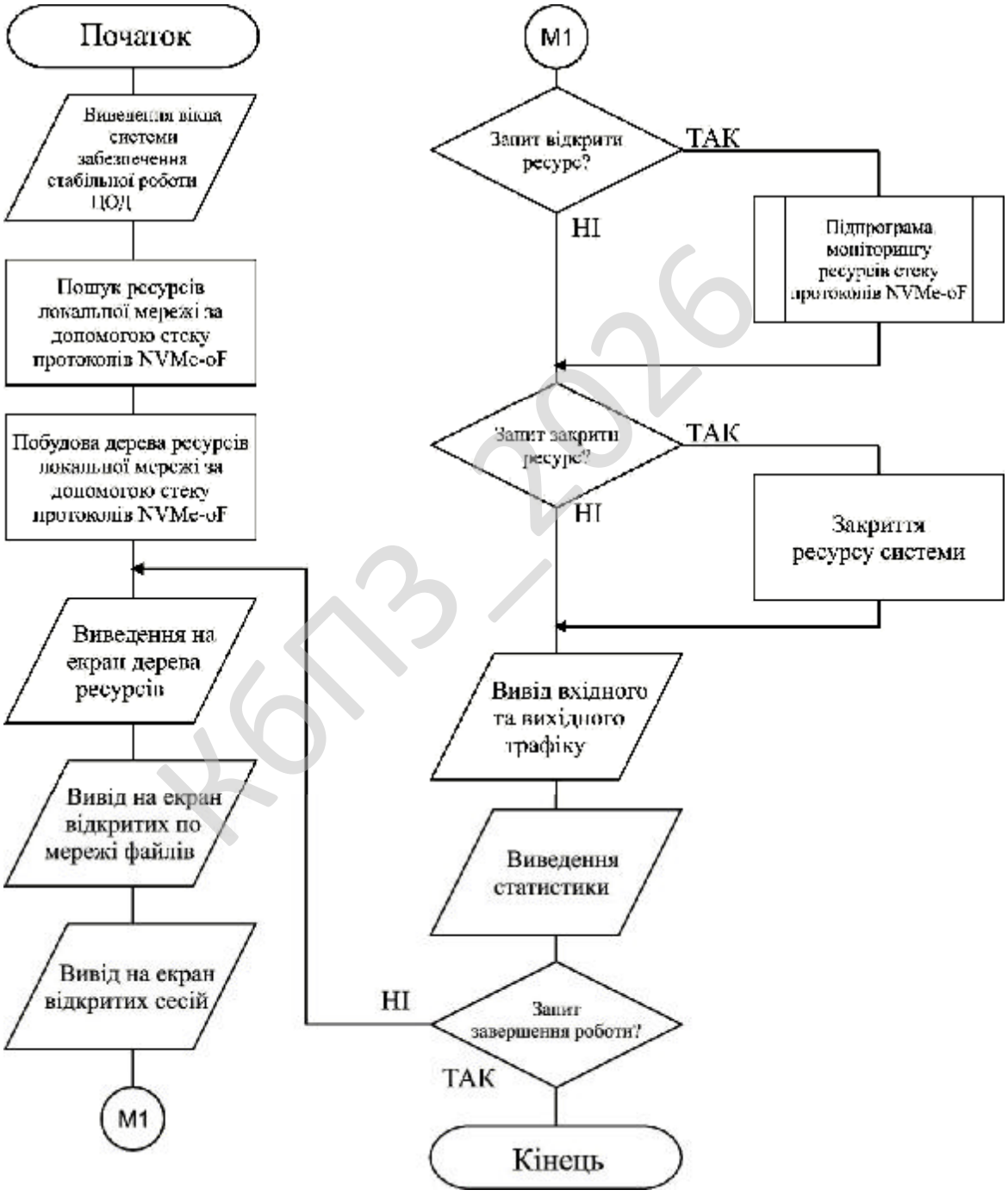

Рисунок 4.1 – Блок-схема основної програми

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.

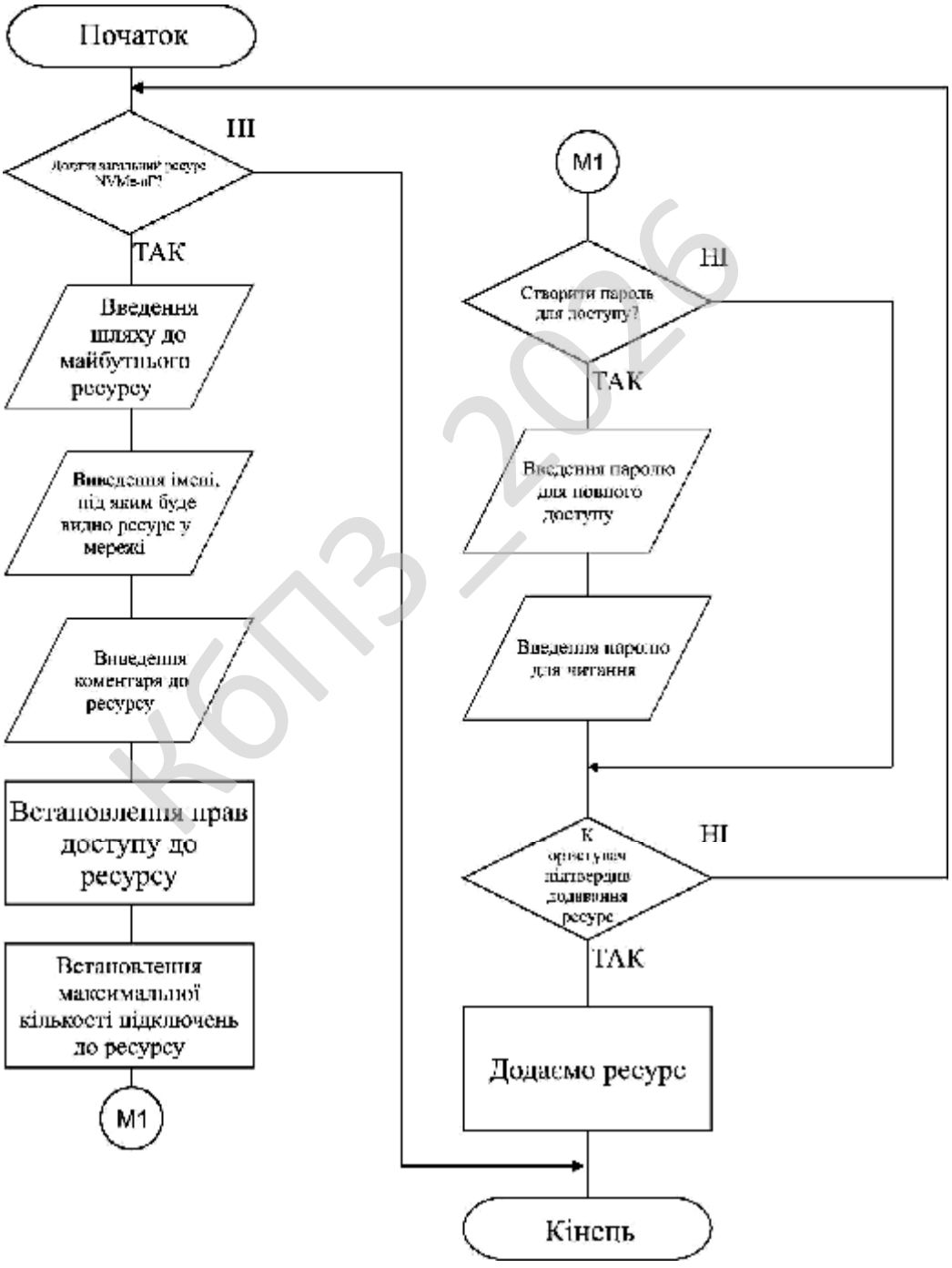

Рисунок 4.2 – Блок-схема роботи підпрограми

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.

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).

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.

Розглянемо розробку проекта в контрольованому середовищі (PRojects IN Controlled Environments – PRINCE) – це метод управління проектами. Метод включає в себе управління, контроль і організацію проекту. «PRINCE2» – це другий реліз зазначеного методу, що є зареєстрованим торговим знаком, державної торгово–промислової палати (Office of Government Commerce – OGC), незалежного підрозділу державного казначейства Сполученого Королівства.

PRINCE2 походить від більш раннього методу PROMPT та методу проектного управління PRINCE, який був розроблений у 1989 р. Центральним телекомунікаційним та комп'ютерним агентством (Central Computer and Telecommunications Agency – CCTA), як державний стандарт Великобританії з управління проектами у сфері інформаційних технологій (IT). Незабаром зазначений стандарт почали використовувати за межами IT сфери. У 1996 р. PRINCE2 був визнаний універсальним методом управління проектами. PRINCE2

став надзвичайно популярним і зараз де–факто є стандартом проектного управління в Великобританії.

Остання версія була випущена у 2009 р., як результат проекту оновлення Prince2:2009 державної торгово–промислової палати Великобританії.

PRINCE2:2009 Оновлення: З 2006 р. метод переглядався, а 16 Червня, 2009 р. був опублікований під назвою «Оновлення PRINCE2:2009». Назва «PRINCE2» (замість «PRINCE3» чи схожої) використано задля демонстрації збереження головних принципів методу.

Все ж таки, це найбільш суттєвий перегляд методу з 1996 р. з метою його адаптації до мінливого бізнес середовища, спрощення і «полегшення», а також кращої інтеграції з іншими методами державної торгово–промислової палати Великобританії (ITIL, P3O, P3M3, MSP, etc.). Головна різниця між релізом 2009 р. та більш ранніми релізами полягає в появі двох довідників замість одного. «Управління проектами, використовуючи PRINCE2» (Managing Projects Using PRINCE2) – оновлена методика управління, що призначена для менеджерів проектів, для яких управління проектами є щоденною діяльністю. «Стратегічне управління проектами, використовуючи PRINCE2» (Directing Projects Using PRINCE2) – новий документ для Керівника проектів і Ради управління проектам, тобто спонсорів/замовників проектів.

Екзамен як з Теоретичної так і з Практичної частини буде базуватися на новому довіднику «Управління проектами» та не буде включати матеріали з довідника «Стратегічне управління проектами». Прохідний бал для Теоретичного екзамену залишиться не змінним. Прохідний бал для Практичного екзамену збільшиться з 50 % до 55 %. Тривалість екзамену з Практичної частини буде зменшено з 3 годин до 2,5 годин.

PRINCE2 – це структурований підхід до управління проектами, який спрямований на управління проектами в рамках чітко визначеної організаційної структури. PRINCE2 описує процедури координації людей та завдань в проекті, я к створювати/планувати проект та що робити, якщо проект потребує внесення

змін через невідповідність фактичного стану виконання проекту плану його виконання. Кожний процес зазначено з ключовими вхідними та вихідними даними, а також з специфічними цілями та завданнями, які необхідно виконати, що загалом дає можливість контролю відхилень від плану.

Поділ методу на частини, що підлягають управлінню, забезпечує ефективний контроль ресурсів, завдяки чому можливо здійснювати контрольований та організований моніторинг проекту. PRINCE2 забезпечує єдину термінологію для всіх учасників проекту. Різноманітні ролі управлінців та зони відповідальності повністю описані та можуть бути адаптовані відповідно до складності проекту та компетенцій організації.

Помилки використання

Іноді PRINCE2 вважають не доцільним для дуже маленьких проектів через обсяг роботи, необхідної для створення та оновлення документів, протоколів та реєстрів. Досить часто це трапляється через нерозуміння, які частини PRINCE2 використовувати, адже PRINCE2 можливо повністю масштабувати.

Звичайно все це чудово, але сприйняття PRINCE2 погіршене тим, що для кожного ефективного проекту, який використовує методологію PRINCE2 існує певна кількість не ефективних проектів, які використовують, так звану, методологію PINO (PRINCE In Name Only – дослівно, PRINCE лише в назві).

Загальними проблемами є:

– Стадія початку проекту опускається/проштовхується і організація переходить відразу до створення Документу ініціювання проекту (Project initiation document – PID).

– Рада проекту не є ефективною – ескалації уходять в «чорну діру» тощо.

– Відсутній Опис продукту (Product description – PD), без якого управління якістю в PRINCE2 не працює.

– Припустимі відхилення не встановлюються, або встановлюються лише для часу виконання та вартості – чітко не визначено, в якій мірі повноваження

передані виконавцям або коли необхідно використовувати ескалацію (піднімати рівень прийняття рішень).

– Проект фактично складається з однієї стадії (або «фази» використовуються, щоб уникнути Огляду виконання стадії завершення (End Stage Review)).

– Документи ініціювання проекту фактично копі–паст (copy–paste – копіювати – вставити) Документу ініціювання проекту з останнього проекту – їм не слідують і вони створюються тому що «потрібно».

– Низький рівень залучення «бізнесу», адже PRINCE2 вважається «технічним» методом – виходи (продукти) проекту можуть не призвести до бажаних результатів.

– Успіхи та невдачі використання PRINCE2 були однією з причин створення Моделі зрілості PRINCE2 (PRINCE2 Maturity Model – P2MM).

Модель зрілості PRINCE2. Модель зрілості PRINCE2 описує набір Сфер Ключових Процесів (Key Process Areas – KPAs), які необхідні для ефективного впровадження та використання PRINCE2 в організації. Довідник PRINCE2 описує управління проектом і не включає опис процесів впровадження PRINCE2 в організації, в той час як P2MM описує саме процеси впровадження PRINCE2.

P2MM описує ключові практики, які додаються до процесів та складових PRINCE2 з метою забезпечення повторюваного використання методу (Рівень 2 KPAs), а також ключові практики необхідні для впровадження методу (Рівень 3 KPAs), як стандартного бізнес–процесу управління проектами, що включає: призначення власника (3.1), адаптацію методу (3.2), навчання (3.3), інтеграцію з іншими системами управління (3.4), механізм забезпечення якості (3.5).

Огляд методу

Блок–схема процесів PRINCE2. Стрілки означають потоки інформації. PRINCE2 – це процесуальний метод управління проектами на відміну від адаптивних методів, таких як Scrum. PRINCE2 2009 визначає 40 окремих активностей (завдань) і об'єднує їх у сім процесів.

Початок проекту

На цьому етапі визначаються учасники проектної команди та готується короткий опис проекту (опис цілей проекту та комерційне обґрунтування проекту). Відповідно до загальних засад методу на цьому ж етапі планується наступний етап проекту. Після завершення зазначених робіт Рада проекту має погодити наступний етап проекту – ініціювання проекту.

Головні завдання включають: призначення керівника проекту та менеджера проекту, визначення та призначення команди управління проектом, створення короткого опису проекту, визначення методу управління проектом, планування наступного етапу проекту (ініціювання).

Ініціювання проекту

Цей етап базується на результатах виконання завдань етапу початку проекту. Короткий опис проекту розширюється до бізнес плану (Business case). Точки контролю якості проекту узгоджуються з точками контролю стану виконання проекту. Створюється план виконання наступного етапу проекту. Результати етапу виносяться на погодження Ради проекту з метою погодження продовження роботи над проектом.

Головні завдання включають: планування показників якості, створення плану проекту, деталізація бізнес плану та ризиків, визначення точок контролю проекту, створення Документу ініціювання проекту (Project Initiation Document).

Стратегічне управління проектом

Цей процес визначає яким чином Рада проекту (що уособлює роль спонсору проекту) буде загалом контролювати проект. Рада проекту дозволяє/санкціонує етап ініціювання проекту, а також сам проект. Стратегічне управління також визначає яким чином Рада проекту має погоджувати план етапів, включаючи погодження будь–якого плану етапу, що може змінити існуючий план проекту у зв'язку з будь–якими змінами строків або ключових показників проекту. Також визначається, яким чином Рада проекту може змінити план проекту, способи закриття проекту.

Головні завдання включають: засоби погодження ініціювання проекту, погодження проекту, погодження етапу або вилучення етапу проекту, підтвердження завершення проекту.

Контроль

PRINCE2 передбачає поділ проекту на етапи, які підлягають контролю. Загалом визначено яким чином пакети завдань призначаються та погоджуються. Також визначається яким чином відслідковується статус виконання завдань і яким чином такий статус доповідається Раді проекту. Засоби для відслідковування та оцінювання виконання проекту пропонуються разом з способами застосування коригуючих дій. Також закріплюються методи ескалації визначеного кола проблем Раді проекту.

Головні завдання включають: призначення та погодження пакетів завдань, оцінювання статусу виконання завдань, звітування статусу виконання завдань, застосування коригуючи дій, ескалація проблем проекту, підтвердження завершення пакету завдань.

Управління межами етапів

Контроль виконання етапів проекту – це визначення переліку завдань, які мають бути виконані в межах етапу. Управління межами етапу визначає, що має бути виконано при завершені етапу.

При завершені етапу необхідно створити план виконання наступного етапу. Загальний план проекту, перелік ризиків проекту та бізнес план за необхідності переглядається та оновлюється. Також визначається перелік дій на випадок, якщо етап не відповідає показникам виконання. Загалом визначається, яким чином звітується про завершення етапу.

Головні завдання включають: планування етапу, оновлення плану проекту, оновлення бізнес плану проекту, оновлення ризиків проекту, звітування про завершення етапу, створення плану дій на випадок не виконання певних завдань проекту.

Завершення проекту

Визначаються завдання, які повинні бути виконані при завершені проекту. Проект має бути формально завершений (ресурси мають бути вивільнені для виконання інших завдань), результат виконання завдань має бути визначений, а сам проект має отримати формальну оцінку.

Головні завдання включають: завершення проекту, визначення результату виконання завдань, оцінку виконання проекту.

Розши́рювана мо́ва розмі́тки (Extensible Markup Language, скорочено XML) – запропонований консорціумом World Wide Web Consortium (W3C) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет. Є спрощеною підмножиною мови розмітки SGML.

XML-документ складається із текстових знаків, і придатний до читання людиною. Стандарт XML (Recommendation, перше видання від 10 лютого 1998, останнє, четверте видання 29 вересня 2006) визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів.

Цей формат достатньо гнучкий для того, аби бути придатним для застосування в різних галузях. Іншими словами, запропонований стандарт визначає метамову, на основі якої шляхом запровадження обмежень на структуру та зміст документів визначаються специфічні, предметно-орієнтовані мови розмітки даних. Ці обмеження описуються мовами схем (Schema), такими як XML Schema (XSD), DTD або RELAX NG. Прикладами мов, заснованих на XML, є: XSLT, XAML, XUL, RSS, MathML, GraphML, XHTML, SVG, а також XML Schema.

Основні поняття. Коректність.

Коректний документ (well-formed document) відповідає всім синтаксичним правилам XML. Документ, що не є коректним, не може називатись XML-

документом. Сумісний синтаксичний аналізатор (Conforming parser) не повинен обробляти такі документи. Зокрема, коректний XML-документ має:

1. Лише один елемент у корені.

2. Непорожні елементи розмічено початковим та кінцевим тегами (наприклад, <пункт> Пункт 1</пункт>). Порожні елементи можуть позначатися «закритим» тегом, наприклад <IAmEmpty />. Така пара еквівалентна <IAmEmpty></IAmEmpty>.

3. Один елемент не може мати декілька атрибутів з однаковою назвою. Значення атрибутів перебувають або в одинарних ('), або у подвійних (") лапках.

4. Теги можуть бути вкладені, але не можуть перекриватись. Кожен некореневий елемент мусить повністю перебувати в іншому елементі.

5. Документ має складатися тільки з правильно закодованих дозволених символів Юнікоду. Єдиними кодуваннями, які обов'язково має розуміти XML-процесор, є UTF-16 та UTF-8. Фактичне та задекларованe кодування (character encoding) документа мають збігатись. Кодування може бути задекларовано ззовні, як у заголовку «Content-Type» при передачі по протоколу HTTP, або в самому документі використанням явної розмітки на самому початку документа. У разі відсутності інформації про кодування, документ має бути в кодуванні UTF-8 (або його підмножині ASCII).

Валідність

Документ називається валідним (valid), якщо він є коректним, містить посилання на граматичні правила та повністю відповідає обмеженням, вказаним у цих правилах (DTD або XML Schema або іншому подібному документі).

Синтаксичний аналізатор

Синтаксичним аналізатором (часто парсер, від parser) називається програма або компонент, що читає XML-документ, проводить синтаксичний аналіз та відтворює його структуру. Якщо синтаксичний аналізатор перевіряє документ на валідність, то такий аналізатор називають валідатором (validating).

Назви елементів чутливі до регістру літер. Наприклад, наступна пара елементів правильна: <Step> … </Step> у той час як ця – ні: <Step> … </step>.

Правильний вибір назв для XML-елементів підкреслюватиме значення даних у створеній мові розмітки. Це сприятиме полегшенню роботи людей з такими документами, зберігаючи можливості для комп'ютерної обробки даних.

Вибір змістовних назв передає семантику елементів та атрибутів для людини, без посилання на зовнішню документацію. Однак це може призвести до надмірності розмітки, що ускладнює редагування й збільшує розмір файлів.

Правильний вибір назв для XML-елементів підкреслюватиме значення даних у створеній мові розмітки. Це сприятиме полегшенню роботи людей з такими документами, зберігаючи можливості для комп'ютерної обробки даних. Вибір змістовних назв передає семантику елементів та атрибутів для людини, без посилання на зовнішню документацію. Однак це може призвести до надмірності розмітки, що ускладнює редагування й збільшує розмір файлів. XML-документи мають як фізичну, так і логічну структуру.

Приклад XML-документа:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<mediawiki xmlns="http://www.mediawiki.org/xml/export-0.3/" xml:lang="uk">
  <page>
    <id>1529</id>
    <revision>
      <id>4382</id>
      <timestamp>2006-09-18T22:11:53Z</timestamp>
      <minor />
      <comment>Interwiki</comment>
      <text xml:space="preserve">{{Wikipedia}}
      </text>
    </revision>
  </page>
</mediawiki>
```

Фізична структура

Сутності (Entity). Головною сутністю є зміст документа. Інші можливі сутності вказуються за допомогою. Посилання на сутності (&назва; в самому документі, та, наприклад %назва; у визначені його типу) можуть слугувати в ролі

позначень спеціальних символів, посилань на спеціальні символи або окремих документів чи фрагментів тексту.

XML-декларація, в ній вказується версія XML, кодування та інша допоміжна інформація.

Декларація типу документа може застосовуватись для того, аби додавати нові типи сутностей та визначати логічну структуру документа.

Логічна структура

XML-документ має ієрархічну логічну структуру, і може представлятись у вигляді дерева. Вузлами цього дерева можуть бути: елементи, фізична структура яких складається із коректної пари відкриваючого та закриваючого тегів (<Назва-тега>) та (</Назва-тега>), або тега порожнього елемента (<Назва-тега />).

Атрибути, що мають вигляд пар ключ-значення (назва атрибута="значення атрибута") і знаходяться або у відкриваючому, або у порожньому тезі (подібно до метаданих).

Вказівки щодо обробки документа (Processing Instruction) (<?Обробник параметр ?>).

Коментарі (<!-- Текст коментаря -->).

Текст, або у вигляді простого тексту, або фрагментів CDATA (<![CDATA[довільний текст]]>).

XML-документ повинен мати лише один кореневий елемент. Решта елементів є піделементами цього кореневого елемента. Деякі веб-браузери здатні безпосередньо відображати XML-документи. Це може досягатись шляхом застосування таблиці стилів (Stylesheet). Вказані у таблиці стилів операції можуть призводити до перетворення XML-документа в інший, відмінний від XML формат.

Коректність XML-документів

Залишивши назви, дозволену ієрархію, та значення елементів і атрибутів відкритою та можливою бути визначеною в спеціалізованих схемах або визначеннях типу документа (DTD).

XML утворює синтаксичну основу для створення спеціалізованих, заснованих на XML мовах розмітки даних.

Загальний синтаксис таких документів стабільний і наперед визначений – документи мають відповідати базовим вимогам XML, гарантуючи те, що довільне програмне забезпечення з підтримкою XML буде здатне щонайменше зчитати і відтворити відносну структуру інформації, що міститься в них.

Схема лише доповнює синтаксичні правила множиною обмежень. Зазвичай схеми обмежують назви елементів та атрибутів, дозволені типи значень і допустиму ієрархію елементів, наприклад, дозволяючи лише елементу з назвою «народження» містити під-елемент з назвою «місяць» та з назвою «день», і кожен із них мусить містити лише літери. Обмеження, вказані в схемі, можуть також включати присвоєння певних типів даних для впливу на те, як обробляється інформація; наприклад, дані елемента «місяць» можна визначити як такі, що містять лише місяць, як це визначено відповідно до використаної мови схем.

Коректний XML-документ, що відповідає обмеженням схеми або DTD, називається валідним.

DTD (Document Type Definition). Найдавнішим форматом схем для XML є успадкований від SGML формат визначення типу документа (Document Type Definition, DTD). У той час, як через включення до стандарту XML 1.0 DTD став поширеним форматом схем, він має такі обмеження:

1. Відсутність нових можливостей XML, із найважливішою з посеред них простори назв.

2. Брак виразності. Деякі формальні аспекти XML-документів неможливо відобразити в DTD.

3. Використовується спеціалізований, заснований не на XML синтаксис для опису схем.

DTD все ще використовується в багатьох програмах, оскільки він вважається найпростішим форматом для аналізу та збереження.

XML Schema. На заміну DTD було розроблено нову мову схем – XML Schema (буквально – XML-схема), скорочено позначається як XSD (від XML Schema Definition). XSD набагато потужніші за DTD в описанні заснованих на XML мов. Вони використовують багатий набір типів даних, підтримують детальніші обмеження на структуру документів, та повинні оброблятись складнішими системами. XSD побудовано на основі XML, що робить можливим використання звичайних інструментів XML для їхньої обробки, хоча реалізації XSD вимагають набагато більше аніж просто можливість читати XML.

Серед недоліків XSD називають такі:

1.Специфікація дуже велика, що робить її складною для розуміння та реалізації.

2. Оснований на XML синтаксис додає надмірності мові, що ускладнює читання та запис XSD.

3. Валідація відносно схеми може бути дорогим додатком до синтаксичного аналізу XML-документів.

4. Можливості моделювання дуже обмежені, без можливості впливу значень атрибутів на вміст елементів.

5. Модель отримання типів даних є дуже обмеженою, зокрема в тому, що отримання шляхом розширення є рідко коли корисним.

6. Механізми ключа/посилання на ключ/унікальності не враховують тип даних.

7. Концепція PSVI (Post Schema Validation Infoset) не має стандартного представлення або прикладного програмного інтерфейса, що працює проти незалежності від реалізації, якщо не виконується повторна валідація.

RELAX NG є іншою поширеною мовою схем для XML. Вперше RELAX NG було визначено стандартом OASIS, а тепер – міжнародним стандартом ISO (як частина DSDL). Ця мова схем має два формати: оснований на XML, та компактний, не-XML. Компактний синтаксис призначений для покращення можливості читання та написання схем, однак, оскільки існує точно визначений

спосіб перетворення компактного формату в оснований на XML, і навпаки, не втрачаються переваги від використання стандартних XML-інструментів. RELAX NG має простіші системи для визначення та валідації у порівнянні з XML Schema, що робить її привабливішою для використання та реалізації. Також існує можливість використання модулів роботи з типом даних; наприклад, автор схеми RELAX NG може вказати, що значення XML-документа мають відповідати визначенням типам даних у форматі XML Schema Datatypes.

ISO DSDL та інші мови схем

Стандарт ISO DSDL (Document Schema Description Languages, мови описання схем документів) об'єднує широке коло малих мов схем, кожна із яких призначена для розв'язання окремих проблем.

До DSDL належить RELAX NG із повним та компактним синтаксисом, мова припущень Schematron, та мови для визначення типів даних, обмежень на літери, перейменування та розкривання мнемонік, та основане на просторах назв перенаправлення фрагментів документів у різні валідатори.

Мови DSDL все ще не мають підтримки як у XML Schema, і є, певною мірою, реакцією видавців на брак можливостей XML Schema для видавничої справи.

Деякі мови схем не тільки описують структуру певного формату XML-документів, а ще і мають обмежені можливості впливу на обробку документів цього формату.

Як DTD, так і XSD мають цю можливість; наприклад, вони можуть визначати значення для атрибутів «за замовченням». Натомість, як RELAX NG так і Schematron таких можливостей не мають.

Обробка XML-документів

До XML-обробки відносяться форматування, синтаксичний аналіз, редагування, перевірка коректності і перетворення в інші формати.

До традиційних технологій обробки XML-документів належать такі три технології:

1. Написання програм на мові програмування із використанням API SAX.

2. Написання програм на мові програмування із використанням API DOM.

3.Застосування механізму перетворення та фільтра.

До новіших технологій, що почали здобувати поширення останнім часом належать: Активний аналіз. Зв'язування даних.

Простий API для XML (SAX)

Простий програмний інтерфейс для XML (Simple API for XML, SAX) є основаним на подіях інтерфейсом лексичного аналізу. Відповідно до цієї моделі, документ аналізується послідовно, а вміст документа передається на обробники подій аналізатора користувача. SAX є порівняно швидким та легким для реалізації проте складним з точки зору задачі отримання інформації із різних частин XML-документів, оскільки розробник аналізатора мусить дбати про відстеження поточної частини документа. Запропонований SAX підхід краще підходить до ситуацій, коли певний тип інформації завжди обробляється однаково, попри те, в якій частині документа вона знаходиться.

Об'єктна модель документа (DOM)

Об'єктна модель документа (Document Object Model, DOM) є програмним інтерфейсом, який дозволяє здійснювати обхід цілого документа так, наче він є деревом, вузлами якого є об'єкти, що відтворюють зміст документа. Документ DOM може створюватись синтаксичним аналізатором або користувачами (з деякими обмеженнями).

Типи даних вузлів DOM-дерев є абстрактними; реалізації мають власні, специфічні для мов програмування типи даних. Реалізації DOM мають тенденцію до інтенсивного використання пам'яті, оскільки зазвичай перед початком роботи документ має бути повністю завантажений, оброблений та перетворений на дерево об'єктів.

Перетворення документів

Extensible Stylesheet Language фільтр в родині XSL може перетворювати XML-документи на інші XML-документи, для перегляду на екрані або друку.

XSL-FO є декларативною, заснованою на XML мовою для макетування сторінок. XSL-FO процесор може перетворювати XSL-FO документ в інший, не заснований на XML формат, такий як PDF.

XSLT є декларативною, основаною на XML мовою описання перетворення документів. XSLT-процесор може використовувати XSLT-стиль в ролі інструкції для перетворення дерева даних, представленого одним XML-документом на інше дерево, що може потім бути серіалізоване в XML, HTML, простий текст, або інший, підтримуваний процесором, формат.

XQuery є розробленою консорціумом W3C мовою для написання запитів, створення та перетворення XML-даних.

XPath є подібною до DOM моделлю дерева та мовою описання шляхів для вибору даних в XML-документах. XSL-FO, XSLT та XQuery використовують XPath. XPath також містить бібліотеку додаткових функцій.

Активний аналіз

З точки зору активного аналізу (Pull parsing) XML-документ розглядається як послідовність елементів, які зчитуються послідовно, використовуючи шаблон проектування ітератор.

Такий підхід дозволяє створення рекурсивних аналізаторів, у яких структура коду відображає структуру аналізованих XML-документів, проміжні результати аналізу можуть бути використані і розміщені у вигляді локальних змінних в підпрограмах, що виконують аналіз, передані як параметри до підпрограм нижчого рівня або повернені до підпрограм вищого рівня.

До прикладів активних аналізаторів належать StAX у мові програмування Java, SimpleXML у PHP та System.Xml.XmlReader у .NET.

Активний аналізатор створює ітератор, що послідовно обходить різні елементи, атрибути та дані в XML-документі. Код, що використовує цей «ітератор», може перевіряти поточний елемент (аби дізнатись, наприклад, чи є цей елемент стартовим, кінцевим або текстовим), та дізнаватись про його атрибути (локальна назва, простір назв, значення XML-атрибутів, зміст тексту

тощо) і може пересунути ітератор на наступний елемент. Таким чином, код аналізатора може зчитувати інформацію із документа під час обходу.

Підхід рекурсивного спуску сприяє зберіганню даних у вигляді типізованих локальних змінних в коді аналізатора, в той час як SAX, наприклад, зазвичай вимагає, аби аналізатор явно зберігав проміжні дані в стеку елементів, що є вищими елементами від того, який зараз аналізується. Код активного аналізатора може бути більш прямолінійним та зрозумілим і простішим для підтримки, аніж код SAX аналізатора.

Зв'язування даних

Іншим підходом до обробки XML-документів є зв'язування даних (Data binding). Відповідно до цього підходу, XML-дані доступні у вигляді спеціальних, строго типізованих структур даних.

4.2 Захист розробленого програмного забезпечення

Дані які використовуються у даній роботі захищаються алгоритмом ДСТУ 7624:2014 («Калина»). При розробці національного стандарту ДСТУ 7624:2014 (блоковий шифр «Калина» і режими його роботи) було ухвалене рішення забезпечити прозорість проектування й використовувати консервативний підхід із застосуванням добре досліджених конструкцій, що забезпечують запас стійкості для безпечного застосування алгоритму в умовах істотного прогресу криптоаналітичних технік і засобів обробки даних.

Національний стандарт підтримує розмір блоку й довжину ключа шифрування 128, 256 і 512 біт (довжина ключа дорівнює розміру блоку або у два рази перевищує його), забезпечуючи нормальний, високий і надвисокий рівень стійкості (зараз це єдиний у світі стандарт блокового шифрування, що підтримує 512-бітові симетричні ключі). Різні варіанти забезпечують гнучкість вибору параметрів для розроблювачів систем криптографічного захисту, що дозволяє

одержати як найвищий рівень швидкодії, так і найбільший запас стійкості перетворення.

Високорівнева конструкція використовує добре досліджену Square-подібну SPN-структуру, застосовувану в алгоритмах AES/Rijndael, Whirlpool, «Стрибог», «Коник» і багатьох інших. Циклове перетворення побудоване на базі таблиць підстановки (S-блоків) і множення на МДР-матрицю над кінцевим полем, забезпечуючи необхідні криптографічні властивості. Застосування саме такої конструкції дозволяє забезпечити доказову стійкість до диференціального, лінійному й ін. видам криптоаналізу, одночасно забезпечуючи ефективну реалізацію на широкому спектрі програмних і програмно-апаратних платформ. При виборі розміру МДР-матриці був прийнятий в увагу розмір кешу L1 сучасних і перспективних процесорів, що дозволило оптимізувати швидкодія програмної реалізації шифру.

Альтернативний варіант, який розглядався при розробці циклової функції, – ARX перетворення (Addition-rotation-xor). Цей підхід реалізований у шифрі SPECK (розроблений Агентством національної безпеки США й переданий у відкритий доступ), у блокових алгоритмах, на основі яких побудовано сімейство геш-функцій SHA-0,1,2 і ін. Перевагою походу є компактність і швидкодія перетворення. Але, у той же час, є й істотний недолік, пов'язаний з відсутністю методів, що дозволяють виконати строге аналітичне обґрунтування криптографічної стійкості таких розв'язків. Навіть із наймогутнішими у світі можливостями для аналізу, США кілька раз були змушено модифікувати свої стандарти гешування через знайдені уразливості: з 1993 по 1995 рр. діяв SHA-0, з 1995 по 2001 рр. застосовувався SHA-1, з тих пор використовується SHA-2. Через питання, що піднімаються, до стійкості й цієї версії, у США з 2008 по 2012 рр. був проведений міжнародний конкурс SHA-3. До теперішнього часу розроблений проект стандарту FIPS 202, що описує нову криптографічну геш-функцію.

Таким чином, консервативний і прозорий підхід до проектування нового національного стандарту України обумовив вибір добре перевіреної конструкції

на базі S-блоків і лінійного перетворення, для якої можливо забезпечити доказову стійкість до різних видів криптоаналізу.

При порівнянні характеристик S-блоків і інших симетричних перетворень можна відзначити, що саме національний стандарт України забезпечує найбільшу нелінійність булевих функцій, що дає додатковий запас стійкості до лінійного криптоаналізу. Ще більше значення нелінійності для взаємо- однозначної підстановки можна забезпечити, застосовуючи, наприклад, аффінно-тквівалентні статечні функції в кінцевому полі, але такі перетворення, використані в AES, Camellia і ін. алгоритмах, ставлять шифр під погрозу реалізації алгебраїчної атаки (цей метод криптоаналізу був успішно застосований проти шифру Keeloq, використовуваного в системах автомобільної безпеки).

У якості схеми розгортання ключів була запропонована нова конструкція з наступними властивостями:

– забезпечення криптографічної стійкості до відомих методів аналізу, відсутність «слабких» ключів, які можуть погіршити властивості перетворення;

– зручність програмної й програмно-апаратної реалізації (для формування циклових ключів застосовуються тільки операції, використовувані при шифруванні);

– висока обчислювальна складність відновлення ключа шифрування по одному або декільком цикловим ключам.

Остання властивість забезпечує додатковий захист до атак на реалізацію, коли зловмисник намагається атакувати інженерні розв'язки (вимірюючи споживаний пристроєм струм, навмисне викликаючи збої в роботі через навмисний перегрів шифратора та ін.). Ця особливість є істотною перевагою для ряду додатків, зокрема, при реалізації шифрування на смарт-картах, USB-токенах та ін., коли ключ прошитий у пристрої й повинен бути захищений від зовнішнього доступу (наприклад, у модулях доступу до платних цифрових ТБ-каналівм і ін.)

Кількість циклів шифрування залежить від довжини ключа: 10 циклів для 128-бітового, 14 циклів для 256-бітового й 18 циклів для 512-бітового ключа шифрування.

У порівнянні з іншими алгоритмами на основі Square-подібної SPN-структури, блоковий шифр «Калина» має наступні істотні конструктивні відмінності:

– початкове й кінцеве забілювання з використанням модульного додавання (264) для підвищення складності криптоаналітичних атак;

– застосування чотирьох різні S-блоків (замість одного) із властивостями для захисту від алгебраїчних атак, і при порівнянні характеристик з іншими блоковими й потоковими шифрами забезпечують найбільшу нелінійність булевих функцій (104), що дає додатковий запас стійкості перетворення;

– збільшений розмір МДР-перетворення, що поліпшує криптографічні властивості й дозволяє оптимізувати швидкодія на сучасних 64-бітових платформах;

– нову односпрямовану схему формування циклових ключів, що забезпечує захист від атак, ефективність програмної й програмно-апаратної реалізації, разом з додатковою стійкістю до методів аналізу спеціального виду.

Оцінка криптографічної стійкості до диференціального, лінійному, алгебраїчному, інтегральному й іншим методам аналізу (практичний критерій) показала, що шифр є стійким при 6 циклах для 128-бітового блоку, 7 циклах для 256-бітового й 9 циклах для 512-бітового (кожний додатковий цикл забезпечує експонентний ріст складності криптоаналізу). Таким чином, шифр, що містить 10, 14 і 18 циклів для блоку 128, 256 і 512 біт відповідно, забезпечує захист від розглянутих видів аналізу й має істотний запас стійкості.

Крім блокового шифру, ДСТУ 7624:2014 визначає режими роботи, що відповідають як ISO 10116:2006, так і додаткові, призначені для сучасних систем криптографічного захисту IP-трафіка, прозорого шифрування носіїв інформації й ін. У стандарті визначені обсяги повідомлень, після обробки яких потрібна

обов'язкова зміна ключа. Крім того, приводяться рекомендації розроблювачам, що обертають увагу на необхідність запобігання атак з використанням особливостей реалізації засобів шифрування. Зокрема, враховані особливості, що дозволяли організацію атак BEAST і CRIME/BREACH у протоколах SSL/TLS, повторне приймання повідомлення, відновлення конфіденційних параметрів на основі залежності часу шифрування від оброблюваних даних (через промахи кешу при табличній реалізації) і ін.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у бакалаврської дипломної роботі система забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF. 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; Верхнього меню; Розділу обрання групи; Розділу виведення результату роботи системи.

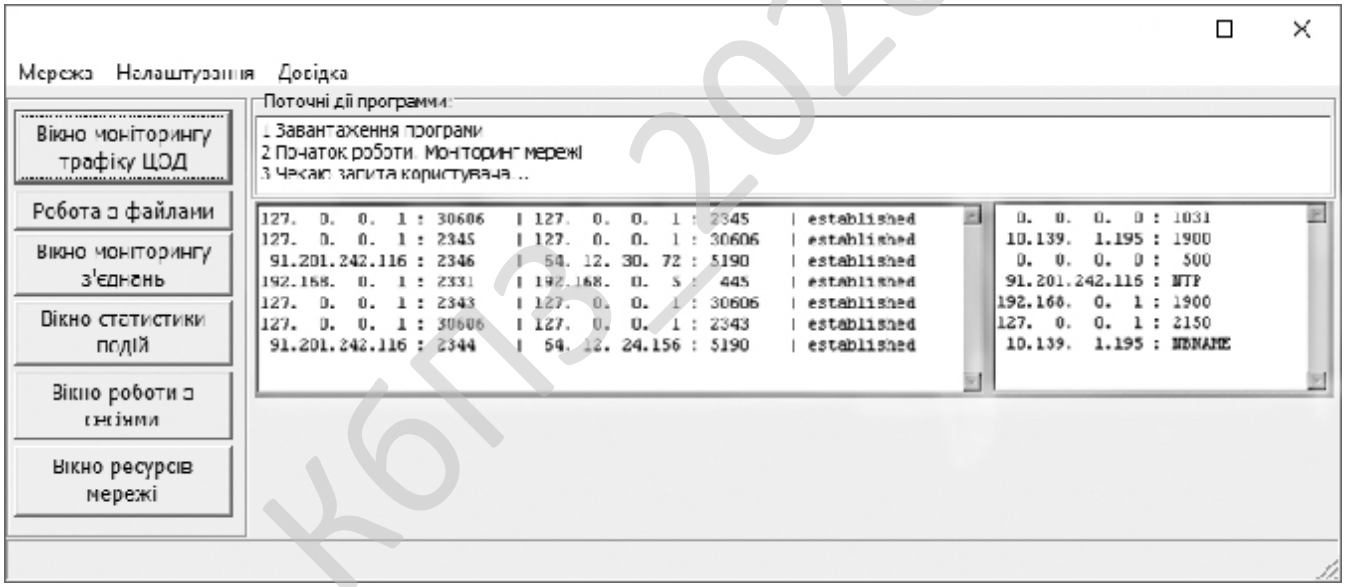

Рисунок 5.1 – Головне вікно ПЗ

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.

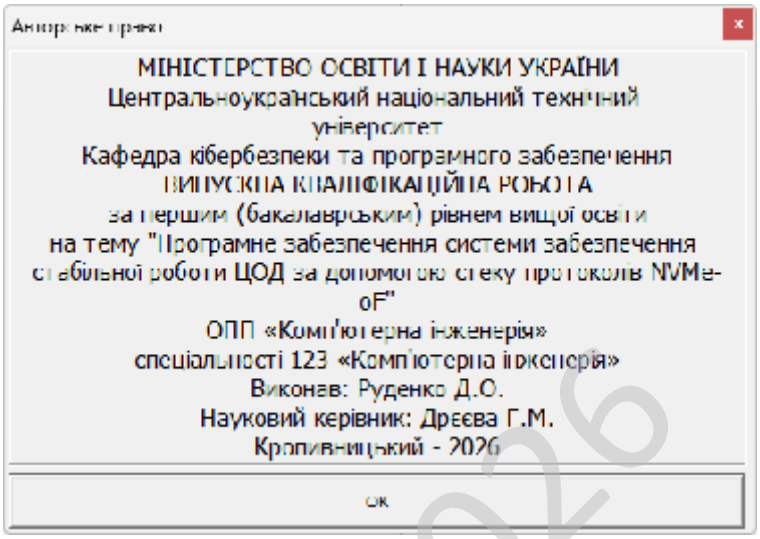

Рисунок 5.2 – Авторське право

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.

Проводилось тестування чорної скриньки. Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе. Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок. Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми. Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ІТс).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ІТс.

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.

– Помилок характеристик (необхідна ємність пам'яті і т.д.).

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми. В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання. Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості. Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму. В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF.

– Досліджена система забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF.

– На основі отриманих результатів досліджень створена програмна реалізація системи забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF.

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 забезпечення стабільної роботи ЦОД за допомогою стеку протоколів NVMe-oF. 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 ДСТУ 7624:2014.

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

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

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

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. 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.

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

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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).

44. 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

45. 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.

46. 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.

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

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

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

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

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

52. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових

праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. - 2016. - С. 121-127.

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

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

55. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

56. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

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