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 проектування структурованої кабельної мережі ЦОД”

КбПЗ_2026

Виконав здобувач вищої освіти
IV курсу, групи КІ-22-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Момонт В.А.
«____» _____ 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

Момонту Вадиму Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи проектування структурованої кабельної мережі ЦОД*

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 проектування структурованої кабельної мережі ЦОД*

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*

КБПЗ-2026

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.	

КбПЗ_2026

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
_____ Момонт В.А.
(прізвище та ініціали)

АНОТАЦІЯ

Момонт В.А. Програмне забезпечення системи проектування структурованої кабельної мережі ЦОД. 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 проектування структурованої кабельної мережі ЦОД.

Метою розробки є програмне забезпечення системи проектування структурованої кабельної мережі ЦОД.

Результат роботи – програмна реалізація системи проектування структурованої кабельної мережі ЦОД.

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, структурована кабельна мережа, ЦОД

КПЗ – 2026

ABSTRACT

Momont V.A. Software for the design system of a structured cabling network of a data center. 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 design system of a structured cabling network of a data center.

The purpose of the development is the software for the design system of a structured cabling network of a data center.

The result of the work is the software implementation of the design system of a structured cabling network of a data center.

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

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

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

The program is developed in the Python environment.

Keywords: computer engineering, structured cabling, data center

КБПЗ_2026

ЗМІСТ

КОПІЯ 2026

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ГК	–	горизонтальні кабелі
ІТ	–	інформаційні технології
ІТС	–	інформаційно-телекомунікаційні системи
МБ	–	магістраль ЦОД
МК	–	магістраль комплексу
ПЗ	–	програмне забезпечення
РП	–	розподільний пункт
СКС	–	структуровані кабельні системи
СКУД	–	системи контролю та управління доступом
ТП	–	точка переходу
ТР	–	телекомунікаційні роз’єми
УАТМ	–	управлінська автоматична телефонна мережа
ЦОД	–	центр обробки даних
QoS	–	механізми контролю й керування якістю

КбПЗ_2026

ВСТУП

Актуальність теми. Центри обробки даних бувають різних форм і розмірів, і з цим пов'язані величезні відмінності у використанні кабелів. У даній роботі ми зосередимося на традиційних корпоративних центрах обробки даних. Які їхні потреби та як вони впливають на кабельну інфраструктуру?

Для спрощення скажімо, що існує 3 типи центрів обробки даних: корпоративні, гіпермасштабні та хмарні. Всі вони підпадають під одні й ті ж стандарти кабельних систем. Міжнародний стандарт ISO/IEC розділяє їх на ISO/IEC 11801-5 (кабельні системи) та ISO/IEC 14763-2 (шляхи, простори). Американський стандарт TIA об'єднує всі вони під стандартом TIA-942-B.

Корпоративний центр обробки даних є одним із найрізноманітніших, а також найменшим із трьох типів, зазначених вище. У світі існують мільйони таких центрів, і вони зазвичай призначені для використання одним власником. Ми спостерігаємо тенденцію, коли корпоративні центри обробки даних переходять до гібридної хмарної моделі. Частина їхніх некритично важливих програм переноситься в приватну або публічну хмару, тоді як критичні програми залишаються в корпоративному центрі обробки даних. Прикладами є виділені сервери з власними протоколами або системами, які не можна перенести за межі їхнього приміщення з міркувань безпеки, обслуговування або з інших бізнес-причин.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи проектування структурованої кабельної мережі ЦОД.

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

– Огляд існуючих систем проектування структурованої кабельної мережі ЦОД.

КПЗ_2026

– Дослідження системи проектування структурованої кабельної мережі ЦОД.

– Програмна реалізація системи проектування структурованої кабельної мережі ЦОД.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі проектування структурованої кабельної мережі ЦОД.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи проектування структурованої кабельної мережі ЦОД, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КБПЗ_2026

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Далеко не всі нововведення в нормативні документи СКС масово впроваджуються. Причини можуть бути самими різними: неочевидність ідеї, що просувається, складність реалізації, зміна вихідних посилок, помилки в базових положеннях і т.д. Разом з тим частина нововведень не знаходить своєчасного відбиття в стандартах через досить тривалий цикл відновлення базових документів (приблизно 5-7 років).

Прикладом слабкої затребуваності системних пропозицій може служити досить активно обговорювана в середині 90-х років минулого століття централізована оптична архітектура. Так, при впровадженні нових рішень компонентного рівня досить довго залишалася «не при справах» техніка Категорії 6, що розроблялася із прицілом на підтримку функціонування двопарних мережевих інтерфейсів 1 Gigabit Ethernet, що не одержали помітного поширення. Тільки із часом потенційні можливості кабельних трактів цієї категорії виявилися затребуваними спочатку в промислових, а потім в інформаційно-телекомунікаційних системах (ІТС), де було потрібно забезпечити масове дистанційне живлення термінальних пристроїв за технологіями PoE й PoE+.

Прискореному впровадженню перспективних розробок сприяють інші технічні документи. До таких відносяться системні бюлетені TSB у США й звіти TR на міжнародному рівні. Всі доповнення, що довели свою фактичну затребуваність, уводяться в текст наступної редакції стандарту. Іноді сам звіт у переробленій формі стає стандартом (наприклад, ISO/IEC TR 147633 через чотири роки перетворився в повноцінний стандарт ISO/IEC 147633; у нього були включені нові положення, через що обсяг тексту збільшився приблизно в три

КПЗ-2026

рази). Відомо також видання «проміжних» редакцій стандарту ISO/IEC 11801 з одним або декількома доповненнями.

У США до розробки стандартів підключилася організація BICSI. Змістовна частина видаваних нею керівництв (Best Practices) потім включається ANSI у національні стандарти. Прикладом може служити документ BICSI 002-2011, значима частина якого присвячена СКС.

1.2 Область застосування

Промислові кабельні системи й інформаційна проводка ЦОД істотно відрізняються від свого прототипу – СКС офісного призначення. Це треба навіть із термінології, використовуваної розроблювачами.

Відмінності кабельних систем ЦОД обумовлені дуже високою швидкістю обміну інформацією, що, у свою чергу, визначається більшим обсягом переданих даних, масовим поширенням хмарних структур, а також відсутністю на об'єкті «білкових» споживачів інформації з їхніми обмеженими можливостями по її сприйняттю.

Потреба у високошвидкісних каналах зв'язку в сполученні з обмеженою швидкодією сучасної електроніки привела до того, що в ЦОД масово застосовується паралельна передача, а обмін інформацією здійснюється переважно за допомогою волоконої оптики.

Останній фактор обумовлений:

– істотно більше високою щільністю конструкції лінійних і шнурових кабелів, що полегшує організацію повітряного охолодження активного мережевого устаткування;

– значною часткою ліній ЦОД довжиною понад 30-40 м; при таких довжинах оптичні інтерфейси генерують менше тепла, що має критичне значення для даної області застосування;

КОПЗ_2026

– можливістю скористатися наробітками з області мереж зв'язку загального користування при переході на наступний рівень швидкості (на відміну від мідножильних ліній).

У міру наближення підтримуваних швидкостей до 400 Гбіт/с починає рости значення одномодової техніки. При паралельній передачі поряд із просторовим ущільненням починає активно використовуватися спектральне. Потенційна кількість несучих в одномодовій техніці значно вище в порівнянні із багатомодової (не більше чотирьох для SWDM). Крім того, розроблювачі можуть скористатися наробітками з мереж зв'язку загального користування. У цій ситуації економічна перевага багатомодової техніки на відстанях до 200-300 м стає не настільки очевидною.

Проте частка багатомодової техніки зменшується досить повільно. Це пояснюється тим, що потенційно вона здатна забезпечити швидкості аж до 1,6 Тбіт/с. Крім того, реальна потреба в надшвидкісних лініях поки не настільки висока, та й необхідність мінімізації витрат на їхню реалізацію не варто скидати з рахунків.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи проектування структурованої кабельної мережі ЦОД, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КОПЗ – 2026

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

Наш цифровий світ став залежним від бездротового з'єднання та хмарних додатків, і ми часто забуваємо, що передача даних та зв'язок залежать від оптоволоконних кабелів у центрах обробки даних.

Структурована кабельна система є основою сучасної інфраструктури центрів обробки даних, що забезпечує взаємозв'язок між серверами, зовнішні підключення та високопродуктивні обчислення.

Оскільки центри обробки даних продовжують розвиватися та розширюватися, щоб задовольнити зростаючі потреби штучного інтелекту (ШІ) та інших обчислювальних потреб, впровадження ефективної структурованої кабельної системи стало важливішим, ніж будь-коли.

Щоб задовольнити величезні потреби штучного інтелекту, розробники будують нові центри обробки даних – ті, що побудовані з урахуванням штучного інтелекту. Затримка в центрі обробки даних набагато менш важлива, ніж затримка всередині центру обробки даних. Тому кабельні з’єднання між кожною стійкою та кластером вимагає набагато більше уваги, щоб групи серверів працювали якомога швидше та злагодженіше, особливо зі спеціальними шаблонами зв’язку штучного інтелекту, такими як паралелізм моделей та розподілене навчання.

Що таке структурована кабельна система в центрах обробки даних?

Структурована кабельна система – це стандартизований підхід до організації та управління кабельною інфраструктурою мережі в центрі обробки даних.

КПЗ-2026

Це передбачає використання систематичного розташування кабелів, роз'ємів та шляхів для створення гнучкої, масштабованої та легкокерованої мережі.

На відміну від кабельної системи «точка-точка», структурована кабельна система використовує ієрархічну топологію, яка розділяє мережу на окремі секції, що спрощує усунення несправностей, обслуговування та модернізацію.

Уявіть собі свій центр обробки даних як галасливе місто. Будівлі – це сервери, а вулиці – кабелі, що їх з'єднують. Структурована кабельна система подібна до добре спланованої міської мережі, яка забезпечує ефективне та безхаосне підключення кожної будівлі (сервера) до головних доріг (мережі).

Переваги структурованої кабельної системи в центрах обробки даних

Впровадження структурованої кабельної системи в центрах обробки даних пропонує численні переваги:

– **Покращена організація**: структурована кабельна система забезпечує акуратну та організовану інфраструктуру, зменшуючи безладдя кабелів та покращуючи загальний естетичний вигляд.

– **Підвищена гнучкість**: Модульна конструкція дозволяє легко додавати, переміщувати та змінювати елементи без порушення роботи всієї мережі.

– **Підвищена ефективність**: Добре організовані кабелі покращують потік повітря, що призводить до кращої ефективності охолодження та зниження витрат на енергію.

– **Надійна продуктивність**: Дотримуючись певних стандартів, структурована кабельна система забезпечує безперебійний та швидкий потік даних між пристроями.

– **Спрощене усунення несправностей**: Стандартизоване маркування та документація спрощують виявлення та швидке вирішення проблем.

– **Забезпечення майбутнього**: Правильно спроектована структурована кабельна система може враховувати майбутні технології та вимоги до пропускної здатності.

КОПЗ–2026

– **Економічно ефективний**: Хоча початкові витрати на встановлення можуть бути вищими, структурована кабельна система зменшує довгострокові витрати на обслуговування та модернізацію.

Ефективна та неефективна структурована кабельна система

Що ми маємо на увазі під «ефективною структурованою кабельною системою»?

Можливо, найкраще повернутися до аналогії Ерінгера про центр обробки даних як місто. Якщо ви коли-небудь подорожували кудись у місті, ви знаєте, що дістатися з пункту А до пункту Б не завжди просто.

Ефективна поїздка включатиме чітко визначені дороги, синхронізоване світлофорне освітлення та добре сплановані смуги для повороту, тоді як неефективна поїздка може включати затори з частими зупинками, вузькі місця та розвороти, необхідні для досягнення пункту призначення.

Ефективна структурована кабельна система характеризується:

– Правильне управління та організація кабелів.

– Оптимальне використання простору та шляхів.

– Дотримання галузевих стандартів та найкращих практик.

– Масштабованість та гнучкість для врахування майбутнього зростання.

– Чітке маркування та документація.

Натомість, неефективна система може демонструвати:

– Заплутані або невпорядковані кабелі.

– Переповнені доріжки та стелажі.

– Недотримання стандартів.

– Обмежена масштабованість та гнучкість.

– Неякісне або непослідовне маркування.

Наслідки неефективної структурованої кабельної системи

Інвестування в ефективну структуровану кабельну мережу дозволяє центрам обробки даних працювати надійніше, ефективніше та економічно ефективніше.

КПЗ-2026

Організація кабельних розводок у вашому центрі обробки даних – це більше, ніж просто естетика, адже це дозволяє уникнути жахливих конструкцій «спагеті-монстрів», які виглядають так само привабливо, як гадюча яма.

Невпровадження ефективної структурованої кабельної системи може призвести до кількох негативних наслідків:

– **Збільшення часу простою**: Неорганізована кабельна система ускладнює усунення несправностей, що потенційно може призвести до триваліших періодів простою.

– **Вищі експлуатаційні витрати:** Неефективне кабельне прокладання може призвести до збільшення споживання енергії та витрат на обслуговування.

– **Зниження продуктивності**: Неправильне прокладання кабелів може призвести до перешкод сигналу та погіршення продуктивності мережі.

– **Обмежена масштабованість**: Неефективна система може мати труднощі з адаптацією до зростання та нових технологій.

– **Небезпека**: Заплутані кабелі можуть створювати небезпеку спотикання та перешкоджати належному потоку повітря, що потенційно може призвести до перегріву.

Найкращі практики проектування ефективної структурованої кабельної системи

Найкращі практики проектування ефективних структурованих кабельних систем починаються з ієрархічного підходу: розділіть кабельну систему на логічні рівні, такі як робоча зона, горизонтальне кабельне з'єднання, телекомунікаційна шафа, магістральне кабельне з'єднання та головний розподільний щит.

Така ієрархічна структура допоможе вам створити надійну та ефективну мережеву інфраструктуру, яка зможе задовольнити потреби вашої організації, що постійно змінюються.

Щоб забезпечити ефективну структуровану кабельну систему для вашого центру обробки даних, враховуйте такі рекомендації:

КПЗ_2026

– **Плануйте майбутнє:** Під час проектування структурованої кабельної системи враховуйте майбутнє зростання та технологічний прогрес. Впроваджуйте масштабовану конструкцію, яка може враховувати підвищені вимоги до пропускної здатності та нове обладнання без капітального ремонту.

– **Дотримуйтесь галузевих стандартів:** дотримуйтесь встановлених галузевих стандартів, таких як ANSI/TIA-942, для кабельних систем центрів обробки даних. Ці стандарти містять рекомендації щодо типів кабелів, методів монтажу та вимог до продуктивності.

– **Впроваджуйте належне керування кабелями:** використовуйте рішення для керування кабелями, такі як кабельні лотки, сходові стійки та вертикальні органайзери для організації та легкого доступу до кабелів. Це покращує потік повітря та спрощує обслуговування.

– **Виберіть правильний тип кабелю:** Виберіть відповідні типи кабелів залежно від потреб вашого центру обробки даних. Вибираючи між мідним та оптоволоконним кабелем, враховуйте такі фактори, як вимоги до пропускної здатності, обмеження відстані та умови навколишнього середовища.

– **Оптимізація проектування кабельних траєкторій:** проектування ефективних кабельних траєкторій, що мінімізують довжину кабелю та уникають потенційних джерел перешкод. Розділіть кабелі живлення та передачі даних, щоб зменшити електромагнітні перешкоди.

– **Впровадження узгодженої системи маркування:** Розробіть та впровадьте чітку, узгоджену систему маркування для всіх кабелів, роз'ємів та обладнання. Це спрощує завдання з усунення несправностей та обслуговування.

– **Документуйте все:** Ведіть детальну документацію вашої кабельної інфраструктури, включаючи кабельні маршрути, з'єднання та специфікації. Оновлюйте цю інформацію в міру внесення змін.

– **Використовуйте високоякісні компоненти**: Інвестуйте у високоякісні кабелі, роз'єми та інші компоненти, такі як ті, що виробляються DCS на їхньому американському заводі, щоб забезпечити оптимальну продуктивність та

КПЗ - 2026

довговічність вашої структурованої кабельної системи.

– **Розгляньте модульні рішення**: впроваджуйте модульні кабельні рішення, які дозволяють легко розширювати та реконфігурувати ваш центр обробки даних у міру розвитку.

– **Проводьте регулярні аудити та технічне обслуговування**: Регулярно проводите аудити вашої кабельної інфраструктури, щоб виявити та усунути потенційні проблеми, перш ніж вони перетворяться на серйозні. Впроваджуйте проактивний графік технічного обслуговування, щоб забезпечити оптимальну продуктивність.

Дотримуючись цих найкращих практик, ви можете спроектувати та впровадити ефективну структуровану кабельну систему, яка задовольнятиме потреби вашого центру обробки даних зараз і в майбутньому.

Світ IT-інфраструктури постійно розвивається, і мережеве кабельне забезпечення не є винятком. Зі зростанням бізнесу, зростанням потреб у даних та розвитком технологій важливо бути на крок попереду нових тенденцій, щоб забезпечити швидке, надійне та безпечне підключення. Від розумніших центрів обробки даних до передових кабельних матеріалів, 2026 рік обіцяє захопливі розробки в галузі мережевого кабельного забезпечення, які вплинуть на те, як організації планують, встановлюють та обслуговують свої мережі.

Збільшення використання волоконно-оптичних кабелів

Оптоволоконні рішення продовжують набирати обертів, оскільки організації прагнуть до більшої пропускної здатності та підключення на більші відстані. У 2026 році мережеві кабельні системи все частіше використовуватимуть одномодове та багатомодове оптоволокно для корпоративних мереж, центрів обробки даних та телекомунікаційних застосувань. Оптоволокно пропонує вищу швидкість, зменшення перешкод та готовність до зростаючих потреб у даних, що робить його критично важливим компонентом сучасної інфраструктури.

КПЗ_2026

Структурована кабельна система для розумних будівель

Розумні будівлі спираються на інтегровані системи, такі як пристрої Інтернету речей, камери безпеки, системи контролю доступу та засоби контролю навколишнього середовища. Мережева кабельна система є основою цих систем, забезпечуючи надійне з'єднання для інтелектуальних датчиків та пристроїв. У 2026 році структурована кабельна система буде розроблена з урахуванням масштабованості, що дозволить об'єктам легко модернізуватися та розширюватися в міру появи нових технологій.

Стандарти високошвидкісного Ethernet

Зі зростанням популярності хмарних обчислень, застосувань штучного інтелекту та потокового відео 4K/8K, підприємствам потрібні вищі швидкості мережі. Очікується, що мережеві кабелі підтримуватимуть передові стандарти Ethernet, такі як 25G, 40G та 100G, особливо в центрах обробки даних та корпоративних магістральних мережах. Ці високошвидкісні кабельні рішення зменшать затримку та покращать загальну продуктивність мережі, забезпечуючи безперебійний зв'язок між пристроями та місцями розташування.

Акцент на сталому розвитку

Сталий розвиток стає пріоритетом в IT-інфраструктурі. У 2026 році тенденції мережевих кабелів включатимуть екологічно чисті матеріали, компоненти, що підлягають переробці, та енергоефективні установки. Виробники виробляють кабелі, які мінімізують вплив на навколишнє середовище, зберігаючи при цьому високу продуктивність, що відповідає цілям корпоративного сталого розвитку та допомагає підприємствам зменшити свій вуглецевий слід.

Розширення PoE (живлення через Ethernet)

Технологія живлення через Ethernet (PoE) продовжує революціонізувати способи отримання живлення пристроями. У 2026 році все більше організацій розгортатимуть мережеві кабелі, здатні підтримувати вищі стандарти PoE, що дозволить таким пристроям, як камери безпеки, VoIP-телефони та системи контролю доступу, отримувати живлення та дані по одному кабелю. Це спрощує

КОПЗ_2026

встановлення, зменшує витрати на інфраструктуру та підвищує енергоефективність.

Покращені рішення для управління кабелями

Оскільки мережі стають щільнішими та складнішими, ефективне управління кабелями є надзвичайно важливим. Передові тенденції мережевої кабельної системи включають кабелі з кольоровим кодуванням, модульні патч-панелі та структуровані траєкторії, що зменшують перевантаження та спрощують обслуговування. Розумне управління кабелями не лише покращує продуктивність, але й допомагає запобігти випадковим перебоям, простоям та дорогому усуненню несправностей у великомасштабних установках.

Інтеграція з бездротовими мережами

Хоча бездротові мережі розширюються, вони все ще залежать від надійної дротової магістралі. Тенденції мережевих кабелів у 2026 році будуть зосереджені на інтеграції з Wi-Fi 6/7 та майбутніми бездротовими технологіями. Поєднання високошвидкісного кабелю з передовими бездротовими точками доступу забезпечує безперебійне покриття, вищу пропускну здатність даних та кращу загальну продуктивність мережі як для офісних, так і для промислових середовищ.

Моніторинг мережі на основі штучного інтелекту

Штучний інтелект змінює способи моніторингу та обслуговування мереж. Передові рішення для мережевих кабельних систем у 2026 році включатимуть датчики та системи моніторингу, які використовуватимуть штучний інтелект для виявлення потенційних проблем, прогнозування збоїв та оптимізації продуктивності. Такий проактивний підхід зменшує час простою, підвищує надійність та дозволяє IT-командам зосередитися на стратегічних ініціативах, а не на реактивному обслуговуванні.

Кабельні системи, орієнтовані на безпеку

Кібербезпека залишається критично важливою проблемою. Тенденції мережевих кабелів включають підвищену увагу до фізичної безпеки, такої як

КПЗ - 2026

захищені від несанкціонованого доступу корпуси та безпечні методи маршрутизації. Захищаючи фізичний рівень, підприємства можуть зменшити ризик несанкціонованого доступу або випадкового пошкодження, доповнюючи заходи безпеки на основі програмного забезпечення.

Забезпечення майбутнього та масштабованість

Мабуть, найважливішою тенденцією 2026 року є мережі, орієнтовані на майбутнє. Прогресивні організації інвестують у мережеві кабельні системи, які можуть впоратися зі зростаючими потребами в даних, новими технологіями та розширенням без капітального ремонту. Масштабована, високопродуктивна кабельна система гарантує, що бізнес залишатиметься конкурентоспроможним та готовим до нових інновацій у мережах, хмарних обчисленнях та інтеграції Інтернету речей.

Тенденції, що формують мережеві кабельні системи у 2026 році, відображають поєднання швидкості, ефективності, сталого розвитку та безпеки. Волоконна оптика, високошвидкісний Ethernet, розширення PoE, моніторинг за допомогою штучного інтелекту та інтеграція розумних будівель – це лише деякі з розробок, які необхідно враховувати бізнесу. Випереджаючи ці тенденції, організації можуть створювати надійні, масштабовані мережі, які відповідають сучасним вимогам і готові до майбутнього зростання.

Інвестування в сучасні рішення для мережевих кабельних систем забезпечує високу продуктивність, скорочення часу простою та довгострокову адаптивність. З розвитком технологій підтримка актуальності вашої мережевої інфраструктури є ключовим фактором для забезпечення безперервності бізнесу, підвищення операційної ефективності та забезпечення безперебійного підключення на всіх пристроях та в усіх місцях розташування.

VC4 Service2Create

Єдине джерело достовірної інформації для вашої мережі.

VC4 Service2Create (S2C) керує даними для **будь-якої мережевої інфраструктури та топології** й забезпечує **повний огляд** ваших **мереж від**

КОПЗ_2026

кількох постачальників.

Він надає точні дані для підтримки вашого бізнесу та операційних підрозділів, пришвидшуючи технічне обслуговування та підтримку, а також допомагаючи вам ефективніше планувати.

Точно та послідовно консолідуйте дані про мережеву інвентаризацію та активи, використовуючи 10 модулів та розширень, для побудови **оптимальних операцій** та **надання послуг** на вашому ринку.

Service2Create Modules

Модульний, масштабований бізнес-інструмент OSS

Рисунок 2.1 – Інтерфейс користувача Service2Create

Модуль інвентаризації

Модуль інвентаризації VC4 S2C фіксує всі активи – фізичні, логічні, віртуальні та сервісні, а потім зіставляє їх з фізичною мережею, забезпечуючи насичене інтерактивне представлення:

– Фізична інвентаризація.

– Логічна та віртуальна інвентаризація.

– Інвентаризація послуг.

– Планування.

– Місткість.

ГІС-модуль

Повний географічний огляд ваших мереж, що дозволяє точно планувати та здійснювати операції:

– Чітке географічне уявлення про всі мережеві активи.

– Функції ГІС для планування та експлуатації оптоволоконних та мідних мереж.

– Прямий зв'язок фізичного з логічними з'єднаннями та послугами.

Модуль виділеної лінії

Об'єднання всіх орендованих з'єднань, включаючи орендовану потужність від інших операторів. Це підтверджується фінансовою звітністю та аналізом:

– Повний огляд підключень орендованих ліній

– Пов'язані орендовані лінії та власні об'єкти мережевого інвентарю

– Зведений реєстр комерційної інформації

– Позбавтеся зайвих орендованих ліній

– Автоматизовані перевірки рахунків-фактур

Модуль керування IP-адресами

Керування та пошук усіх IP-адрес (IPv4/IPv6) для реєстрації та призначення:

– Діапазони IP-адрес.

– Бронювання.

– Автоматичний розрахунок IP-підмережі.

– Автоматичне виявлення IP-адрес з активних мереж.

Модуль телефонного номера

Керуйте всіма географічними та негеографічними номерами відповідно до

КПЗ–2026

об'єктів інвентаризації:

– Підтримує всі системи нумерації телефонів країн.

– Підтримує перенесення телефонних номерів.

– Зрозуміла функціональність звітності та налаштування для місцевих органів влади.

– Автоматичне виявлення номерів телефонів з активних мереж.

Модуль інтеграції

Уніфікований набір процесів для забезпечення оновлення даних з інтеграцією з усіма пов'язаними системами з OSS та BSS:

– Інтеграція з елементами різних постачальників.

– Інтерфейси NMS/EMS для автоматичного виявлення та узгодження мережі.

– BSS / OSS RESTful API.

– Активація та відстеження змін/статусу.

– Інтерфейси управління продуктивністю.

Модуль звітності та панелі інструментів

Повний набір інструментів для створення запланованих та налаштованих звітів, заснований на чіткій, інтерактивній візуалізації та з автоматичними опціями планування:

– Звіти.

– Панелі інструментів.

– Користувацькі звіти та шаблони.

– Планування та розповсюдження.

Модуль замовлень та заявок на несправності

Метою функціональності управління замовленнями S2C є автоматизація, перевірка та захист процесів протягом життєвого циклу мережі:

– Управління замовленнями та доставкою послуг.

– Управління заявками на проблеми.

– Процес налаштовується, що дозволяє налаштувати робочий процес.

КОПІЯ 2026

– Користувацькі звіти та шаблони.

Модуль аналізу впливу

Виявляти, відстежувати та інформувати клієнтів або мережеві оператори про вплив планових та незапланованих відключень:

– Планова робота.

– Аналіз впливу несправностей.

– Одиночні точки відмови.

– Інформуйте клієнтів послідовно.

Модуль складу та запасних частин

Ефективно обробляйте всі транзакції та досягайте цільових показників доставки, з повним контролем витрат для пришвидшення розгортання послуг та усунення несправностей.

– Склад та запаси.

– Управління запасними частинами.

– Управління замовленнями на придбання.

– Фінанси та управління активами.

– Контроль рахунків-фактур та платежів.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – динамічна інтерпретована об’єктно-орієнтована скриптова мова програмування із строгою динамічною типізацією. Офіційний сайт мови програмування Python https://www.python.org/. Python – багатоцільова мова програмування, яка дозволяє писати код, що добре читається. Відносний лаконізм мови Python дозволяє створити програму, яка буде набагато коротше свого аналога, написаного на іншій мові. Python – багатоплатформова мова програмування. Це означає, що програми на Python можна запускати в різних операційних системах без будь-яких змін.

КбПЗ – 2026

Ще однією перевагою Python є його стандартна бібліотека, яка встановлюється разом з Python і містить готові інструменти для роботи з операційною системою, веб-сторінками, базами даних, різними форматами даних, для побудови графічного інтерфейсу програм тощо. Програми, написані на мові програмування Python, можуть бути як невеликими скриптами, так і складними системами. Python абсолютно безкоштовний.

Швидкість виконання коду Python

Один з можливих недоліків Python – швидкість виконання коду. Python не є компільованою мовою. Код на Python спочатку компілюється у внутрішній байт-код, який потім виконується інтерпретатором Python. У більшості випадків при використанні Python виходять програми повільніші в порівнянні з такими мовами, як C.

Втім, сучасні комп'ютери мають таку обчислювальну потужність, що для більшості застосунків швидкість розробки важливіша швидкості виконання, а програми на Python зазвичай пишуться набагато швидше.

Окрім того, Python легко розширюється модулями, написаними на C або C++. Такі модулі можуть використовуватися для виконання частин програми, що створюють інтенсивне навантаження на процесор.

Використання Python

Python використовується для різних цілей: для створення ігор і веб-застосунків, розробки внутрішніх інструментів для різноманітніх проектів. Мова також широко застосовується в науковій області для досліджень і розв'язування прикладних завдань.

Застосування мови програмування Python:

1. BitTorrent – протокол для обміну даними.

2. Ubuntu Software Center – вільне програмне забезпечення для пошуку, установки і видалення пакунків в системі Ubuntu Linux.

КПЗ-2026

3. Blender – програма для створення тривимірної комп'ютерної графіки, що включає засоби моделювання, анімації, вимальовування, пост-обробки відео, а також створення відеоігор.

4. GIMP – растровий графічний редактор, із підтримкою векторної графіки.

5. World of Tanks.

6. Вільна енциклопедія Вікіпедія.

7. Пошукова система Google.

8. DropBox – файловий хостинг, що включає персональне хмарне сховище, синхронізацію файлів і програму-клієнт.

9. YouTube – популярне відеосховище.

Версії Python

Мови програмування з часом змінюються – розробники додають в них нові можливості, а також виправляють помилки. Так з'являються різні версії мови. Наприклад, код написаний на Python 2 у більшості випадків не буде працювати у версії Python 3 без внесення додаткових змін.

Процесор є найважливішим компонентом в комп'ютері. Одна з основних функцій процесора – це обробка даних згідно комп'ютерної програми, яка є списком інструкцій, шляхом виконання арифметичних і логічних операцій над фрагментами даних.

Кожна інструкція в програмі – це команда, яка «повідомляє» процесору, яку операцію він повинен виконати. Процесор комп'ютера може розуміти лише ті інструкції, які написані на машинній мові. Машинна мова – це штучна мова, створена для передачі команд комп'ютеру. За допомогою машинної мови створюються ефективні програми, оскільки розробник отримує доступ до всіх можливостей процесора. Машинна мова – мова низького рівня.

Інструкція машинної мови існує для кожної операції, яку процесор здатний виконати – є інструкція для додавання чисел, є інструкція для віднімання

КОПЗ-2026

чисел і т.д. Увесь набір інструкцій, який центральний процесор може виконати, відомий як набір інструкцій процесора.

Наприклад, у вас є певна програма, яка зберігається на диску вашого комп'ютера. Для виконання програми, ви здійснюєте подвійний клік на значку програми. Це змушує програму копіюватися з диска в оперативну пам'ять, після чого процесор комп'ютера виконує копію програми, яка знаходиться в оперативній пам'яті.

Коли процесор виконує інструкції програми, він бере участь у процесі, який є відомим як цикл fetch – decode – execute (отримати – декодувати – виконати). Цей цикл виконується для кожної інструкції у програмі і складається з трьох кроків:

Отримати

Програма – це послідовність інструкцій на машинній мові. Першим кроком циклу є завантаження (отримання) наступної інструкції з пам'яті в процесор.

Декодувати

Інструкція машинної мови – це двійкове число, яке представляє команду, що повідомляє процесору виконати певну операцію. На цьому кроці процесор декодує інструкцію, яку було «витягнуто» з пам'яті, для визначення того, яка операція повинна виконуватись.

Виконати

Останній крок циклу – виконати операцію.

Хоча процесор комп'ютера розуміє тільки машинну мову, людині непрактично писати програми на машинній мові. Така програма може мати тисячі або навіть мільйони бінарних інструкцій, і написання такої програми буде дуже обтяжливим процесом.

З цієї причини була створена мова асемблера як альтернатива машинній мові. Замість використання двійкових чисел для написання інструкцій, мова асемблера використовує короткі слова, відомі як мнемокоди.

КОПЗ – 2026

Незважаючи на те, що мова асемблера не вимагає двійкових інструкцій, як у випадку машинної мови, проте вона вимагає високих знань про процесор. Використовуючи мову асемблера, навіть для найпростішої програми, необхідно написати велику кількість інструкцій.

Мова програмування високого рівня дозволяє створювати складні програми, не знаючи, як працює процесор, і не записуючи великої кількості інструкцій низького рівня. Крім того, більшість мов програмування високого рівня використовують слова, які легко зрозуміти.

Python – одна із популярних сучасних мов програмування високого рівня. Python – інтерпретована мова програмування. Python – це високорівнева інтерпретована мова програмування, на відміну від C++, яка є прикладом компільованої мови програмування. Назва Python відноситься як до мови програмування, так і до інтерпретатора – комп'ютерної програми, яка зчитує початковий код (написаний на Python) і виконує інструкції (команди).

Для перекладу мови високого рівня на машинну мову доступні два типи програм:

1. Компілятор.

2. Інтерпретатор.

Завантаження Python

Версії інтерпретатора Python для різних операційних систем доступні для безкоштовного завантаження за адресою https://www.python.org/downloads.

Середовище програмування для Python

Для написання програм використовують текстові редактори або інтегровані середовища розробки, які включають в себе різні інструменти для роботи з кодом: засіб для написання коду (текстовий редактор), інтерактивний інтерпретатор, відлагоджувач тощо.

Текстові редактори та інтегровані середовища програмування для Python:

– IDLE – стандартний редактор Python. Встановлюється разом з Python для користувачів Windows, окремим пакунком для користувачів Linux.

КПЗ_2026

– Notepad++ – безкоштовний текстовий редактор початкового коду, який підтримує велику кількість мов, в тому числі і Python. Лише для користувачів Windows.

– Visual Studio Code – це легкий, але потужний редактор початкового коду, який розповсюджується безкоштовно і доступний у версіях для платформ Linux, Windows і macOS.

– PyScripter – інтегроване середовище розробки для мови програмування Python. Для користувачів Windows. Поширюється безкоштовно.

– Wing IDE 101 – вільне інтегроване середовище для Python, розроблене для навчання програмістів-початківців. Для користувачів Linux, Windows і macOS. Поширюється безкоштовно.

– Geany – вільний текстовий редактор з базовими елементами інтегрованого середовища розробки, доступний для операційних систем Linux, Windows і macOS.

– PyCharm – інтегроване середовище розробки для мови програмування Python. PyCharm є власницьким програмним забезпеченням. Наявна безкоштовна версія Community з усіченим набором можливостей. Для користувачів Linux, Windows і macOS.

– Thonny – IDE для вивчення програмування мовою Python. Для користувачів Linux, Windows і macOS.

– Mu – редактор коду Python для програмістів-початківців. Для користувачів Linux, Windows і macOS.

КОПЗ–2026

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 проектування структурованої кабельної мережі ЦОД.

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.

КБПЗ – 2026

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Якщо ваш офісний Wi-Fi щоразу, коли кілька людей підключаються до відеодзвінка, здається, що він тягнеться крізь багнюку, проблема, ймовірно, не в маршрутизаторі. Проблема в тому, що знаходиться за стінами. Більшість компаній намагаються використовувати технології майбутнього на вчорашніх проводах. Це вузьке місце, яке вбиває продуктивність. Випереджаючи інших

Тенденції структурованої кабельної системи – це не гонитва за блискучими новинками, а забезпечення того, щоб ваша команда могла працювати без колеса смерті, що з'являється на їхніх екранах.

Ми переходимо в епоху, коли «достатньо хороший» зв’язок є перешкодою.

Чому всі переходять на Cat6A та оптоволоконний кабель?

Коротка відповідь? Пропускна здатність та нагрівання. Старі кабелі просто не справляються з високим навантаженням на дані у 2026 році. Чи то конференції 4K, чи масивні хмарні резервні копії, вашій мережі потрібна потужніша магістраль.

Категорія 6A стала базовою. Вона забезпечує швидкість 10 Гбіт/с на повну відстань без зайвих зусиль. Але справжнім зрушенням є те, що оптоволоконне з'єднання переміщується ближче до робочого столу. Оптоволоконне з'єднання більше не призначене лише для магістралі; воно стає основним варіантом для зон з високим навантаженням, оскільки воно стійке до електричних перешкод, що виникають у мідних проводах у переповнених офісах.

Живлення через Ethernet (PoE) більше не є опцією

Уявіть собі, що ви живите свої камери безпеки, світлодіодне освітлення та навіть деякі настільні монітори через один мережевий кабель. Ніяких додаткових

КОПЗ-2026

розеток. Ніяких громіздких адаптерів. Така реальність сучасних стандартів PoE, таких як 802.3bt.

Тенденція рухається в бік «розумних будівель», де кабелі виконують подвійну функцію – передаючи як дані, так і потужне живлення. Це спрощує вашу інфраструктуру та значно скорочує витрати на встановлення. Якщо ви не плануєте PoE++ під час наступного оновлення, ви будуєте застарілу систему з першого дня.

Як тенденції структурованої кабельної системи впливають на дизайн розумного офісу?

Сучасне планування офісів є гнучким. Люди рухаються, відділи переміщуються, а датчики Інтернету речей розташовані всюди, від термостата до кавоварки. Жорстка, старомодна електропроводка з цим не впорається.

Поточний крок – у бік «зональної кабельної мережі». Замість того, щоб прокладати кожен дріт до однієї центральної шафи, ми бачимо консолідовані точки по всій підлозі. Це неймовірно спрощує додавання або переміщення пристроїв, не зносячи стелю кожні шість місяців. Йдеться про гнучкість. Якщо ваш офіс зростає, ваша мережа повинна зростати разом з ним, а не проти нього.

Штучний інтелект та автоматизоване управління інфраструктурою

Раніше кабелі були «дурними». Ви підключали їх і сподівалися, що їх не перетиснуть. Зараз ми спостерігаємо зростання популярності інтелектуальних патч-систем. Ці системи використовують датчики для моніторингу з’єднань у режимі реального часу.

Якщо кабель відключено або порт виходить з ладу, система негайно сповіщає ІТ-відділ. Деякі навіть використовують штучний інтелект для прогнозування можливого збою з’єднання на основі даних про продуктивність. У світі, де п’ять хвилин простою можуть коштувати тисячі, така проактивна технологія є справжнім порятунком.

Оскільки центр обробки даних підприємства знаходиться лише на одному етапі від традиційної структурованої кабельної системи, його потреби дещо

КПЗ-2026

схожі. Ключовими проблемами для більшості цих клієнтів є:

– Легкість встановлення, що дозволяє використовувати їх різноманітним технікам.

– Легкість обслуговування для полегшення регулярних переміщень, додавань та змін (MAC).

– Плани міграції, оскільки центр обробки даних зазвичай є центром витрат, який потребує високої рентабельності інвестицій.

– Мінімізація витрат, щоб уникнути необхідності переходу до хмари.

Кабельні системи для корпоративних центрів обробки даних

Охоплення всередині корпоративних центрів обробки даних, як правило, коротше порівняно з хмарними або гіпермасштабованими, що сприяє використанню мідних та/або багатомодових оптоволоконних з'єднань. Швидкість передачі даних також найнижча, оскільки зазвичай використовується для підключення до виділених серверів та комутаторів.

У великих корпоративних центрах обробки даних обмежений простір може бути значною проблемою, оскільки робота MAC вважається нормою. Тому патч-панелі надвисокої щільності (UHD) є поширеними для забезпечення періодичного обслуговування та розвитку кабельних систем. UHD-панель зазвичай визначається як панель, що містить в середньому 144 LC-роз'єми в 1U. Ці споруди повинні бути спроектовані з урахуванням терміну служби кількох поколінь обладнання, і це одна з основних причин, чому структуровані кабельні системи були розроблені в першу чергу. Також важливо враховувати поточну роботу, яку виконуватимуть оператори (зазвичай у передній частині стійки), порівняно з тією, яку виконуватимуть установники (зазвичай у задній частині стійки). Кожна з них має свій власний набір вимог до доступності.

КОПЗ-2026

3.2 Розробка структурної схеми

Структурні елементи СКС:

– Розподільний пункт комплексу (ЦОД) (РП комплексу).

– Магістраль комплексу (МК).

– Розподільний пункт ЦОД (РП ЦОД).

– Магістраль ЦОД (МБ).

– Розподільний пункт поверху (РП поверху).

– Горизонтальні кабелі (ГК).

– Точка переходу (ТП).

– Телекомунікаційні роз’єми (ТР).

– Робоча область.

– Телекомунікаційні приміщення.

– Апаратні.

– Вводи в ЦОД.

– Адміністрування.

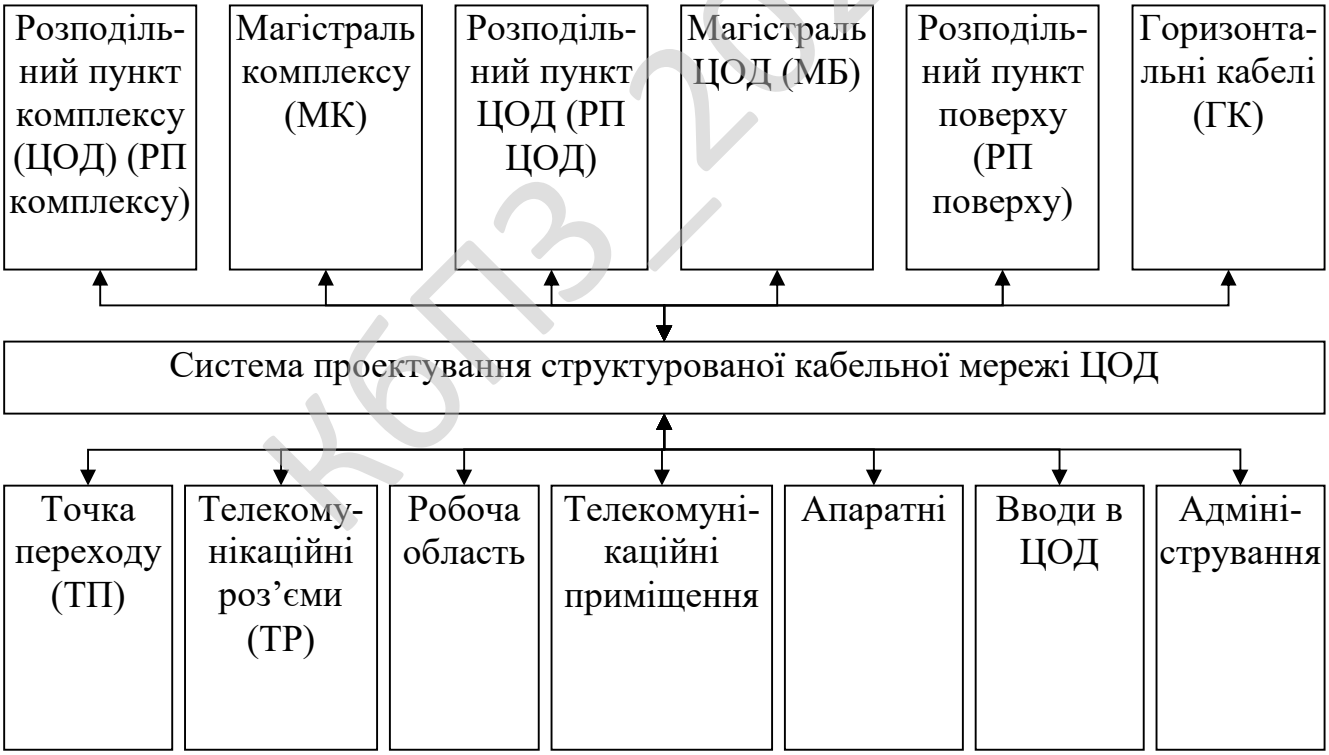


Рисунок 3.1 – Структурна схема системи

Робочі навантаження штучного інтелекту стрімко змушують центри обробки даних підтримувати дедалі вищі швидкості передачі даних по мережі,

КПЗ 2026

збільшуючись з 400 Гбіт/с до 800 Гбіт/с і навіть 1,6 Тбіт/с. Щоб задовольнити ці потреби, структурована кабельна система стала основою для масштабованих, ефективних та високопродуктивних інфраструктур. Зростання вимог до передачі даних спонукає до переходу від традиційних з'єднань низької щільності до високощільних, гнучких кабельних систем, які можуть підтримувати пропускну здатність та надійність, необхідні для операцій штучного інтелекту. Як основа продуктивності та масштабованості, структурована кабельна система тепер повинна обробляти значно збільшені робочі навантаження, забезпечуючи при цьому максимальну рентабельність інвестицій.

Для досягнення цієї мети мережі на базі штучного інтелекту переходять від традиційних дуплексних LC-з'єднань до з'єднань на основі MPO. Сервери на базі GPU зазвичай використовують 8-волоконні роз'єми MPO-08, що підтримують агреговані швидкості передачі даних 800G, як визначено в IEEE 802.3df, по 8 дуплексних лініях (16 волокон) як для багатомодових (800GBASE-SR8), так і для одномодових (800GBASE-DR8) каналів. Така конструкція ефективно збільшує кількість волокон щонайменше в чотири рази порівняно з традиційними LC-конфігураціями. При розгортанні в AI-подах, які розміщують величезну кількість серверів, загальна щільність волокон може бути до восьми разів більшою, ніж у стандартних центрах обробки даних, що підкреслює вирішальну роль передових кабельних систем у забезпеченні продуктивності та масштабованості ШІ.

Цей перехід також сприяє вже поширеному переходу до паралельних оптичних інфраструктур, а також масовому збільшенню використання оптоволоконного зв'язку, як зазначалося, у вісім разів. Цей зсув необхідний для забезпечення зростання пропускної здатності, необхідної для штучного інтелекту, але він також створює нові проблеми, пов'язані з щільністю та управлінням оптоволоконним зв'язком.

Швидке впровадження мережевих швидкостей 400 Гбіт/с та 800 Гбіт/с вимагає значно більшої кількості оптоволоконних з'єднань. Кластери штучного

КОПЗ-2026

інтелекту покладаються на багатоволоконні MPO-з'єднання APC для з'єднань між сервером та кінцевою мережею, а також на більш традиційні одномодові MPO-з'єднання для з'єднань між кінцевою мережею та магістраллю, що означає експоненціальне збільшення обсягів оптоволоконних з'єднань. Без структурованого підходу центри обробки даних ризикують надмірним перевантаженням кабелів, що збільшує труднощі в обслуговуванні та зменшує оптимізацію повітряного потоку.

Робочі навантаження штучного інтелекту працюють у кластерних архітектурах, часто вимагаючи коротших кабельних ліній, причому великий відсоток мереж штучного інтелекту побудовано в SuperPods довжиною <50 м. Це означає, що занепокоєння щодо затримки поширення через додаткові точки підключення, які вводять структуровані кабелі, не є проблемою, оскільки затримка поширення світла на таких відстанях (<50 м на SuperPod) залишається менше 250 наносекунд, що є незначним порівняно із затримками комутації та обробки сигналів.

Помилкове уявлення про те, що структурована кабельна система створює надмірну затримку порівняно з прямим двоточковим з'єднанням, спростовується вагомими доказами того, що затримка від структурованої кабельної системи мінімальна. Більше того, більшість затримок у мережах штучного інтелекту виникають через пряму корекцію помилок (FEC) та буферизацію на рівні комутатора, а не через додаткові волоконно-оптичні роз'єми, які призводять лише до оптичних втрат. Однією з проблем, пов'язаних із структурованою кабельною системою в мережах штучного інтелекту, є додаткові втрати через роз'єми, які можуть спричинити ризик для продуктивності каналу. Цей аргумент слід обговорити, зазначивши, що під час роботи з трансиверами, повністю сумісними зі специфікаціями каналів Ethernet, які виділяють втрати на з'єднання 1,5 дБ для каналів MMF та близько 2,5 дБ для каналів SMF. Цю проблему можна вирішити навіть при розгляді власних конструкцій, оскільки існують ретельні випробування, які показують, що при дотриманні втрат на роз'єми в зазначених

КПЗ-2026

межах та дотриманні належних практик встановлення та очищення з'єднання відповідатимуть стандарту IEEE 802.3df. Таким чином, добре спроектована структурована кабельна інфраструктура не впливає негативно на робочі навантаження штучного інтелекту, чутливі до затримки.

Впровадження структурованої кабельної системи в робочих навантаженнях зі штучним інтелектом: для максимізації рентабельності інвестицій, оптимізації довговічності та забезпечення безперебійної роботи структуровані кабельні системи повинні бути розроблені з урахуванням наступних критеріїв:

Масштабованість та модульність: Центри обробки даних повинні впроваджувати модульні патч-панелі та високощільну кабельну систему MPO, щоб забезпечити безперебійне оновлення зі збільшенням швидкості мережі. Структурований підхід дозволяє краще керувати розширенням оптоволоконного зв'язку без необхідності частого капітального ремонту.

Оптимізовані кабельні шляхи та управління ними: Високощільне прокладання кабелів може призвести до перевантаження шляхів, що негативно впливає на потік повітря та зручність обслуговування. Структуроване прокладання кабелів зменшує ці ризики, об'єднуючи кілька волоконно-оптичних трас у великі магістралі. Такий підхід значно зменшує фізичну площу волоконно-оптичних трас, при цьому оцінки показують скорочення використання трас до 70% при розгортанні структурованого прокладання кабелів.

Надійність та скорочення часу простою мережі: структурована кабельна система покращує ремонтопридатність та мінімізує ризики, пов'язані з надмірним провисанням кабелю, неправильними радіусами вигину та неорганізованим управлінням кабелями. Впровадження структурованих шляхів забезпечує належну документацію, маркування та доступність з'єднань, що спрощує усунення несправностей та скорочує середній час ремонту (MTTR).

Перспективність на майбутнє завдяки високощільному з’єднанню: Перехід до високошвидкісних мереж вимагає інфраструктури, яка підтримує

КОПІЯ 2026

стандарти, що постійно розвиваються. Перехід до 16-волоконних MPO-роз'ємів для розгортання зі швидкістю 800 Гбіт/с забезпечує масштабованість мережі, зберігаючи при цьому сумісність з існуючими системами зі швидкістю 400 Гбіт/с. Інвестування в структуровану кабельну систему, яка враховує майбутні роз'єми вищої щільності, забезпечує безперешкодний шлях міграції для зростаючих потреб у пропускній здатності.

Енергоефективність та сталий розвиток: Оскільки робочі навантаження штучного інтелекту споживають значну кількість енергії, структурована кабельна система може сприяти підвищенню енергоефективності завдяки використанню багатомодового оптоволокна. Багатомодові приймачі споживають до 15% менше енергії, ніж їхні одномодові аналоги, що робить їх привабливим варіантом для робочих навантажень штучного інтелекту, що працюють на менших відстанях досяжності.

Структурована кабельна система проти кабельної системи "точка-точка" в мережах штучного інтелекту

Хоча деякі організації досі покладаються на пряме кабельне з'єднання «точка-точка» для своїх високошвидкісних мереж штучного інтелекту, такий підхід створює кілька проблем:

– Підвищена складність управління оптоволоконним кабелем: кабельні системи типу «точка-точка» можуть створювати хаотичну інфраструктуру з надмірним запасом ресурсів, що ускладнює переміщення, додавання та зміни.

– Обмежена масштабованість: розширення мереж «точка-точка» вимагає додаткових оптоволоконних ліній, що призводить до перевантаження та неефективного використання простору.

– Вищі експлуатаційні витрати: Складність обслуговування та усунення несправностей систем «точка-точка» з часом збільшує експлуатаційні витрати.

На противагу цьому, структурована кабельна система забезпечує добре організовану, масштабовану та зручну в обслуговуванні мережеву інфраструктуру, яка краще підходить для середовищ на базі штучного інтелекту.

КПЗ-2026

Належні методи встановлення та дотримання галузевих стандартів, таких як IEEE802.3df для 800 Гбіт/с по багатомодовому оптоволокну, забезпечують рекомендації, які допомагають підтримувати цілісність мережі та полегшують майбутні оновлення, а також забезпечують сумісність та стабільність продуктивності, а також довговічність та надійність кабельної інфраструктури. Такі фактори, як картографування та документування фізичної інфраструктури, є критично важливими, особливо враховуючи, що швидкість мережі поступово зростає. Повні та точні записи необхідно вести з першого дня, щоб створити базу даних, яку можна використовувати для аудиту поточних установок, і яка є безцінною, коли потрібно внести зміни.

Дорожня карта до 1,6 Тбіт/с і вище

Майбутнє структурованої кабельної мережі розвивається відповідно до галузевих тенденцій. Оскільки мережеві стандарти розвиваються до 1,6 Тбіт/с і вище, рішення для структурованої кабельної мережі повинні адаптуватися. Очікується, що новітні технології, такі як ко-пакована оптика (CPO) та роз'єми малого форм-фактора наступного покоління, такі як MDC, ще більше оптимізують управління оптоволоконними мережами в робочих навантаженнях штучного інтелекту. Крім того, вдосконалені методи модуляції, такі як PAM-4, дозволять підвищити швидкість передачі даних через існуючу оптоволоконну інфраструктуру, зменшуючи потребу в частій заміні кабелів.

Робочі навантаження, керовані штучним інтелектом, вимагають високошвидкісних мережевих інфраструктур високої щільності, що робить структуровану кабельну інфраструктуру необхідною для сучасних центрів обробки даних. Перехід від LC до MPO, збільшення щільності оптоволоконного зв'язку та потреба в масштабованих, ефективних мережевих архітектурах підкреслюють важливість структурованого підходу. Впроваджуючи найкращі практики в структурованій кабельній інфраструктурі, організації можуть оптимізувати продуктивність мережі, мінімізувати проблеми із затримкою,

КОПЗ_2026

зменшити перевантаження шляхів зв'язку та забезпечити інфраструктуру, орієнтовану на майбутнє.

З огляду на те, що навантаження штучного інтелекту продовжують розширювати свій вплив на наше повсякденне життя, підхід до структурованої кабельної системи не лише корисний, а й необхідний для забезпечення безперебійної роботи, масштабованості та довгострокової надійності у високопродуктивних центрах обробки даних.

3.3 Розробка функціональної схеми

Перейдемо до розгляду функціональної схеми. Сформулюємо й вирішимо завдання аналізу функціональної структури системи проектування структурованої кабельної мережі ЦОД. Для цього:

– розроблена концепція проведення аналізу;

– вирішені завдання формального опису структури;

– вирішені завдання обчислення характеристик структури: навантаження на канали зв'язку, вузли й структуроутворююче устаткування системи проектування структурованої кабельної мережі ЦОД.

На рисунку 3.2 представлена функціональна схема системи, які відповідає запропонованому підходу.

Основною метою аналізу структури є визначення параметрів потоків даних, що проходять по каналах зв'язку системи проектування структурованої кабельної мережі ЦОД й вступні на вузли системи проектування структурованої кабельної мережі ЦОД. Однак, завдання структури системи проектування структурованої кабельної мережі ЦОД тільки як сукупності вузлів і зв'язків між ними, не дозволяє досліджувати потоки даних, оскільки потоки даних формуються розв'язуваними на системи проектування структурованої кабельної мережі ЦОД завданнями, точніше застосунками, які запускаються на вузлах системи проектування структурованої кабельної мережі ЦОД й обмінюються між собою даними. Отже, для аналізу системи проектування структурованої

КБПЗ-2026

кабельної мережі ЦОД необхідно відомості про функціональну структуру доповнити відомостями про застосунки і їхнє розміщення в системи проектування структурованої кабельної мережі ЦОД.

Результатами аналізу повинні стати математичні залежності, що дозволяють обчислювати чисельні значення характеристик системи проектування структурованої кабельної мережі ЦОД з урахуванням особливостей конкретної структури системи проектування структурованої кабельної мережі ЦОД.

Запропоновано концептуальний підхід до аналізу функціональної структури системи проектування структурованої кабельної мережі ЦОД, заснований на дослідженні взаємодії застосунків (завдань) як незалежних джерел і приймачів даних. У цьому випадку спочатку визначаються параметри потоків даних між застосунками при виконанні всього комплексу завдань (будується інформаційна модель), а потім, залежно від розміщення застосунків по вузлах системи проектування структурованої кабельної мережі ЦОД й використовуваного устаткування, визначаються параметри потоків даних між вузлами системи проектування структурованої кабельної мережі ЦОД (будується технічна модель).

При цьому не тільки повністю враховуються всі взаємодії між застосунками, але й з'являється можливість проведення аналізу складних ієрархічних мережних структур, шляхом декомпозиції на підмережі системи проектування структурованої кабельної мережі ЦОД, що часто застосовується в технологіях VLAN і VPN. Даний підхід розвивається й застосовується в даній роботі.

Відзначимо, що отримані результати аналізу потоків даних, можна надалі використовувати для проведення більше глибоких досліджень із застосуванням відомих методів і моделей, наприклад, CeMO.

Для аналізу функціональної структури й розрахунку характеристик системи проектування структурованої кабельної мережі ЦОД визначені правила її формального опису, що однозначно задають властивості застосунків і структуру системи проектування структурованої кабельної мережі ЦОД, що дозволяють будувати моделі для проведення розрахунків.

КОПЗ-2026

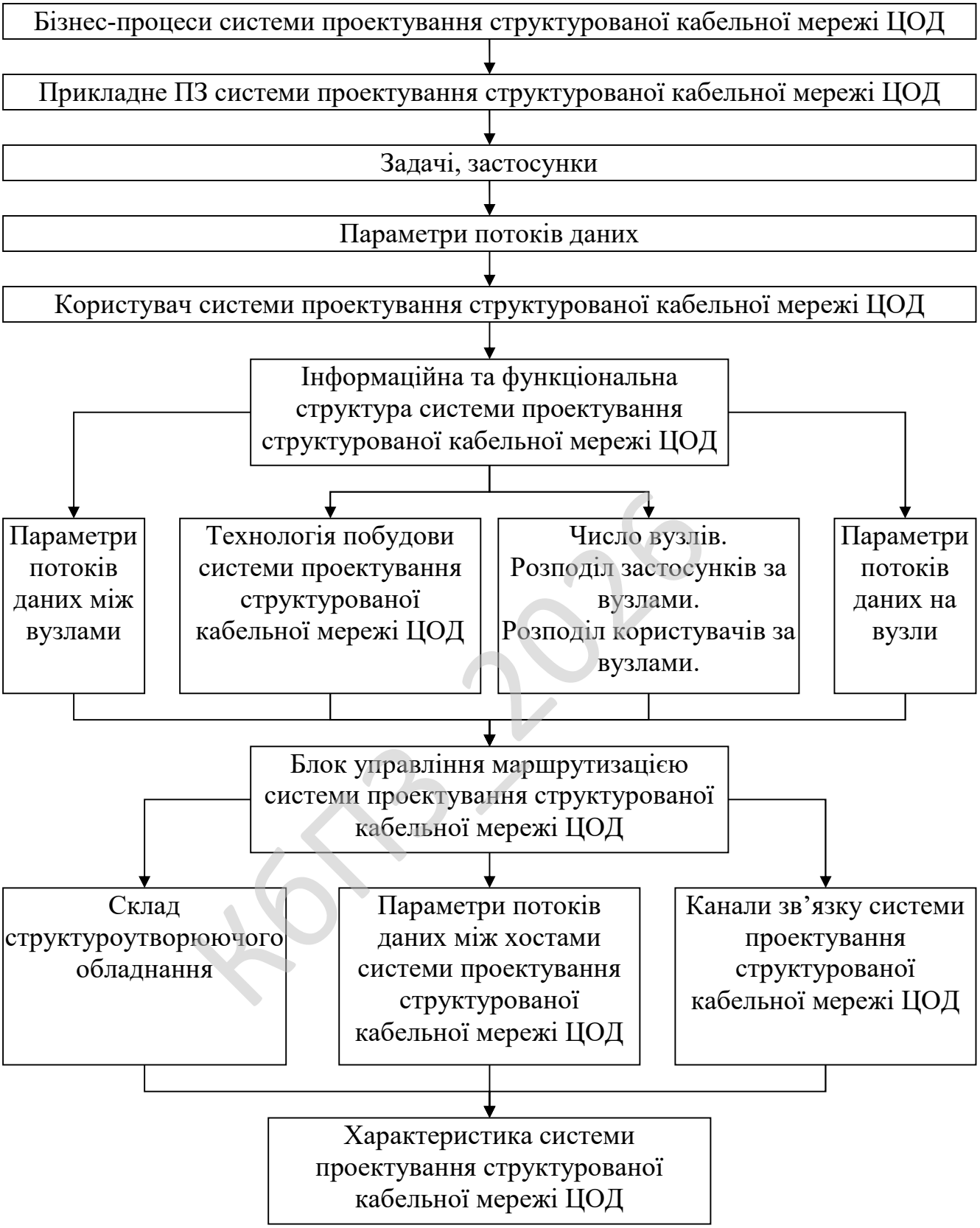


Рисунок 3.2 – Функціональна схема системи

КПІ – 2026

Відповідно до концептуального підходу до проведення аналізу функціональної структури системи проектування структурованої кабельної мережі ЦОД виділені дві складові її структури інформаційна й технічна:

– Інформаційна структура визначає інформаційні потоки між інформаційними вузлами, на яких установлене програмне забезпечення й представляє сукупність вузлів і інформаційних ресурсів, розміщених на цих вузлах. Маючи у своєму розпорядженні дані про інформаційну структуру системи проектування структурованої кабельної мережі ЦОД можна приймати рішення про організацію каналів зв'язку між вузлами системи проектування структурованої кабельної мережі ЦОД, визначати необхідні параметри телекомунікаційної системи, формувати структуру системи проектування структурованої кабельної мережі ЦОД.

– Технічна структура – сукупність структуроутворюючого устаткування, технічних вузлів системи проектування структурованої кабельної мережі ЦОД й каналів зв'язку. Технічному вузлу можуть відповідати декілька інформаційних.

Таким чином, для повноцінного аналізу функціональної структури реальної системи проектування структурованої кабельної мережі ЦОД необхідно провести аналіз складових її інформаційної й технічної структур і зв'язати результати аналізу.

Зв'язування результатів аналізу інформаційної й технічної структур має на увазі відображення характеристик інформаційної структури в характеристики технічної структури й визначення параметрів технічної структури на основі параметрів і характеристик інформаційної структури.

Інформаційна структура системи проектування структурованої кабельної мережі ЦОД задається набором параметрів:

$$\mathbf{SI}=\{N,M,D,L,R,\mathbf{S}_k(k=1,2,\ldots,L),\mathbf{A}_{km}(k=1,2,\ldots,L;m=1,2,\ldots,D),\mathbf{G},\mathbf{H},\mathbf{S}\},$$

де:

– N – число працюючих користувачів;

– M – число задіяних вузлів;

– D – число використовуваних застосунків;

КПЗ-2026

– L – число розв'язуваних завдань;

– R – число баз даних;

– $\mathbf{S}_k = \{\mathbf{p}_k, \mathbf{d}_k, \mathbf{u}_k, \mathbf{W}_k\}$, ($k = 1,2,...L$) – набір даних, що описують розв'язувані завдання, де:

– $\mathbf{p}_k = (p_{k1}, p_{k2}, ..., p_{kD})$, – вектор-рядок, що визначає завданням k додатка, що запускаються;

– $\mathbf{d}_k = (d_{k1}, d_{k2}, ..., d_{kR})$ – вектор-рядок, що визначає використовувані завданням k бази даних (сховища даних);

– $\mathbf{u}_k = (u_{k1}, u_{k2}, ..., u_{kN})$, – вектор-рядок, що визначає завдання k користувачів, що запускають $\mathbf{W}_k = \left\| w_{kij} \right\|$ ($i = 1,2,..., D$; $j = 1,2,..., D$) – матриця, що встановлює послідовність запуску застосунків завданням k;

– $\mathbf{A}_{km} = \{\mathbf{v}_{km}, \mathbf{b}_{km}\}$, ($k = 1,2,...,L$; $m = 1,2,...,D$) – набір даних, що описують застосунки, використовувані завданнями, де:

– $\mathbf{v}_{km} = (v_{km1}, v_{km2}, ..., v_{kmR})$, – вектор-рядок, що задає обсяги даних, якими обмінюється додаток m з базами даних, при рішенні завдання k;

– $\mathbf{b}_{km} = (b_{km1}, b_{km2}, ..., b_{kmD})$ – вектор-рядок, що задає обсяги даних, якими обмінюється додаток m з іншими застосунками, при рішенні завдання k;

– **G** матриця розміщення застосунків по вузлах;

– **H** – матриця підключення користувачів до вузлів;

– **S** – матриця розміщення баз даних по вузлах.

Набір однозначно визначає інформаційну структуру системи проектування структурованої кабельної мережі ЦОД.

Для заданого набору **SI** отримані результати, що дозволяють визначити параметри потоків даних між вузлами системи проектування структурованої кабельної мережі ЦОД.

При цьому використовувалася матриця інтенсивностей потоків запитів користувачів на запуск завдань $\mathbf{\Lambda} = \left\| \lambda_{ij} \right\|$, ($i = 1,2,..., N$; $j = 1,2,..., L$).

КПЗ – 2026

Показано, що матриця інтенсивностей потоків даних між вузлами системи проектування структурованої кабельної мережі ЦОД при рішенні завдання k обчислюється за формулою: $\mathbf{A}_k = \lambda_k \mathbf{Z}_k$, $(k = 1,2,...,L)$. , де $\mathbf{Z}_k$ – матриця обсягів даних, переданих між вузлами, при рішенні завдання. Обчислені також матриця інтенсивностей потоку запитів на запуск додатка номер j, на вузлі номер i – $\mathbf{B}^*$ і матриця сумарної інтенсивності потоку запитів до бази даних номер j, на вузлі номер i – $\mathbf{\Phi}$.

Таким чином, визначена множина параметрів потоків даних інформаційної структури системи проектування структурованої кабельної мережі ЦОД: $\mathbf{PSI(SI)} = \{\mathbf{\Lambda}, \mathbf{Z}_k, \mathbf{A}_k(k = 1,2,...,L), \mathbf{A}, \mathbf{B}^*, \mathbf{\Phi}\}$.

Розроблено моделі для аналізу ієрархічної трьохрівневої інформаційної структури системи проектування структурованої кабельної мережі ЦОД, що дозволяють визначити завантаження каналів зв'язку й мережного устаткування в системи проектування структурованої кабельної мережі ЦОД, що складається із сукупності підмереж і корпоративних серверів.

Отримано наступні результати:

– матриці інтенсивностей потоків даних між групами й усередині груп кожного рівня $\mathbf{A}_1(\mathbf{C}_1) = \mathbf{C}_1\mathbf{A}(\mathbf{C}_1)^T$, $\mathbf{A}_2(\mathbf{C}_2) = \mathbf{C}_2[\mathbf{A}_1(\mathbf{C}_1) - dg(\mathbf{A}_1(\mathbf{C}_1))](\mathbf{C}_2)^T$;

– матриці інтенсивностей потоків даних утворених кожним завданням $\mathbf{A}_{1k}(\mathbf{C}_1)$ і $\mathbf{A}_{2k}(\mathbf{C}_1)$, ($\mathbf{A} = \sum_{k=1}^{L} \mathbf{A}_k$).

Досліджено властивості матриць, що відбивають взаємозв'язки потоків даних окремих завдань, сформульовані у вигляді доведених тверджень.

Як міра ефективності ієрархічної структури запропоновані коефіцієнти поглинання інтенсивностей потоків даних на кожному рівні. Коефіцієнт поглинання на кожному рівні відбиває частку сумарної інтенсивності потоків, що локалізується усередині рівня.

КПЗ_2026

Результати аналізу інформаційної структури дозволяють визначати параметри потоків даних між логічними об'єднаннями вузлів системи проектування структурованої кабельної мережі ЦОД. Інформаційна структура реалізується конкретними технічними засобами й втілюється у вигляді технічної структури. При цьому можливо невідповідність інформаційної й технічної структур, пов'язане з технічними характеристиками й можливостями апаратури. Наприклад, на одному технічному вузлі можуть бути встановлені кілька інформаційних вузлів. У зв'язку із цим розроблені методи й засоби аналізу технічної структури системи проектування структурованої кабельної мережі ЦОД, створюваної на базі інформаційної структури.

Для аналізу роботи реальної системи проектування структурованої кабельної мережі ЦОД розроблений метод формального опису її технічної структури. Структура реальної системи проектування структурованої кабельної мережі ЦОД формується із застосуванням структуроутворюючого устаткування (комутатори, маршрутизатори), до якого підключаються вузли системи проектування структурованої кабельної мережі ЦОД, при цьому створюється багаторівнева (часто трьохрівнева) мережа. Такий підхід застосовується, як правило, при створенні системи проектування структурованої кабельної мережі ЦОД на базі технології VLAN. При цьому групи першого рівня інформаційної структури становлять віртуальні локальні системи проектування структурованої кабельної мережі ЦОД. Другий і третій рівні інформаційної структури призначені для з'єднання цих мереж між собою.

Число комутаторів, використовуваних для з'єднання вузлів технічної структури системи проектування структурованої кабельної мережі ЦОД при створенні груп першого рівня (комутатори першого рівня), визначається особливостями реальної системи проектування структурованої кабельної мережі ЦОД, технічними можливостями комутаторів. Позначимо це число K_1^*, ($K_1^* \geq K_1 \geq 1$). Число комутаторів, для з'єднання комутаторів першого рівня й створення груп другого рівня (комутатори другого рівня) – K_2^*, ($K_2^* \geq K_2 \geq 1$).

КОПЗ-2026

Число комутаторів третього рівня для з'єднання комутаторів другого рівня – $K_3^* \geq 0$.

Технічна структура системи проектування структурованої кабельної мережі ЦОД задається множиною **ST**, елементами якого є:

– $\mathbf{Y}_1^* = \left\| y_{1ij}^* \right\|$, ($i = 1,2,...,M$; $j = 1,2,...,K_1^*$) матриця з'єднань технічних вузлів системи проектування структурованої кабельної мережі ЦОД (робітники станції, сервери) з комутаторами першого рівня;

– $\mathbf{Y}_2^* = \left\| y_{2ij}^* \right\|$, ($i = 1,2,...,K_1^*$; $j = 1,2,...,K_2^*$) матриця з'єднань комутаторів першого рівня з комутаторами другого рівня;

– $\mathbf{Y}_3^* = \left\| y_{3ij}^* \right\|$, ($i = 1,2,...,K_2^*$; $j = 1,2,...,K_3^*$) матриця з'єднань комутаторів третього рівня з комутаторами другого рівня;

– $\mathbf{X}_1^* = \left\| x_{1ij}^* \right\|$, ($i, j = 1,2,..., K_1^*$) матриця з'єднань комутаторів першого рівня;

– $\mathbf{X}_2^* = \left\| x_{2ij}^* \right\|$, ($i, j = 1,2,..., K_2^*$) матриця з'єднань комутаторів другого рівня;

– $\mathbf{X}_3^* = \left\| x_{3ij}^* \right\|$, ($i, j = 1,2,..., K_3^*$) матриця з'єднань комутаторів третього рівня.

Для кожного рівня задані матриці, що задають пропускні здатності каналів зв'язку, використовуваних при побудові мереж:

$$\mathbf{C}_1^*(\mathbf{Y}_1^*), \mathbf{C}_2^*(\mathbf{Y}_2^*), \mathbf{C}_3^*(\mathbf{Y}_3^*), \mathbf{C}_1^*(\mathbf{X}_1^*), \mathbf{C}_2^*(\mathbf{X}_2^*), \mathbf{C}_3^*(\mathbf{X}_3^*).$$

Для заданої множини **ST** отримані наступні результати:

– матриця інтенсивностей інформаційних потоків, між комутаторами першого рівня й усередині груп технічних вузлів, підключених до комутаторів першого рівня: $\mathbf{A}_1^*(\mathbf{Y}_1^*) = (\mathbf{Y}_1^*)^T \mathbf{A} \mathbf{Y}_1^*$;

– матриця сумарних інтенсивностей інформаційних потоків для комутаторів другого рівня: $\mathbf{A}_2^*(\mathbf{Y}_2^*) = (\mathbf{Y}_2^*)^T [\mathbf{A}_1^*(\mathbf{Y}_1^*) - dg(\mathbf{A}_1^*(\mathbf{Y}_1^*))]\mathbf{Y}_2^*$;

КбПЗ_2026

– матриця інтенсивностей інформаційних потоків, між і усередині комутаторів третього рівня: $\mathbf{A}_3^*(\mathbf{Y}_3^*) = (\mathbf{Y}_3^*)^T[\mathbf{A}_2^*(\mathbf{Y}_2^*) - dg(\mathbf{A}_2^*(\mathbf{Y}_2^*))]\mathbf{Y}_3^*$;

– вектори інтенсивностей інформаційних потоків, що надходять на комутатори всіх рівнів: $\boldsymbol{\lambda}_1^* = (\lambda_{11}^*, \lambda_{12}^*, ..., \lambda_{1K_1^*}^*)$, $\boldsymbol{\lambda}_2^* = (\lambda_{21}^*, \lambda_{22}^*, ..., \lambda_{2K_2}^*)$, $\boldsymbol{\lambda}_3^* = (\lambda_{31}^*, \lambda_{32}^*, ..., \lambda_{3K_3}^*)$;

– вектор сумарних інтенсивностей потоків даних, переданих по каналах зв'язку $\boldsymbol{\gamma}_1^* = (\gamma_{11}^*, \gamma_{12}^*, ..., \gamma_{1M}^*)$, $(i = 1,2,...,M)$.

Отримані також формули для обчислення параметрів потоків даних кожного завдання.

Оскільки в корпоративних мережах великої розмірності використовується маршрутизація, то для обчислення параметрів потоків необхідно застосовувати відповідні алгоритми маршрутизації на графах. Для цього побудована матриця графа зв'язків між комутаторами $\boldsymbol{\Omega}$. Показано, що матриця $\boldsymbol{\Omega}$ може бути представлена у вигляді:

$$\boldsymbol{\Omega} = \begin{pmatrix} \mathbf{X}_1^* & \mathbf{Y}_2^* & \mathbf{0} \\ (\mathbf{Y}_2^*)^T & \mathbf{X}_2^* & \mathbf{Y}_3^* \\ \mathbf{0} & (\mathbf{Y}_3^*)^T & \mathbf{X}_3^* \end{pmatrix}.$$

Таким чином, параметри потоків даних у системи проектування структурованої кабельної мережі ЦОД для заданої технічної структури визначаються множиною:

$$\mathbf{PST(ST)} = \{\mathbf{A}_1^*(\mathbf{Y}_1^*), \mathbf{A}_2^*(\mathbf{Y}_2^*), \mathbf{A}_3^*(\mathbf{Y}_3^*), \boldsymbol{\lambda}_1^*, \boldsymbol{\lambda}_2^*, \boldsymbol{\lambda}_3^*, \boldsymbol{\gamma}_1^*, \boldsymbol{\Omega}\}.$$

Множина **PST(ST)** визначає параметри, як сумарних, так і часток потоків даних для окремих завдань (застосунків), переданих по каналах зв'язку.

Якість рішення завдань залежить від того, яка частина пропускної здатності каналу (смуга) виділена для кожного завдання, що звичайно досягається шляхом застосування режиму гарантованої якості обслуговування (QoS). Тому при рішенні завдань розрахунку потоків із застосуванням систем з

КПЗ – 2026

гарантованою якістю обслуговування проводиться розширення множини **PST**(**ST**) шляхом додавання множини коефіцієнтів поділу каналів – **PKST** . Це дозволяє визначити залежність характеристик системи проектування структурованої кабельної мережі ЦОД від смуги пропущення в каналі (комутаторі), що відводиться для завдання.

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. 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.

КПІ – 2026

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

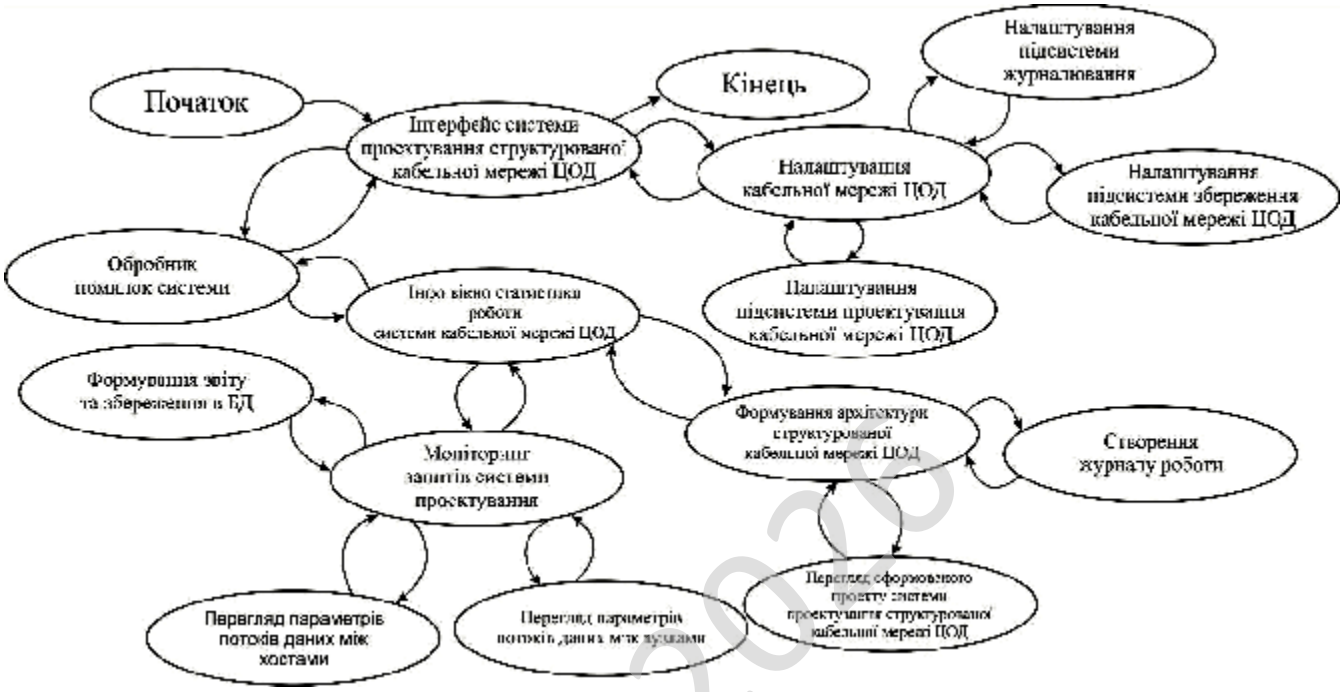


Рисунок 3.3 – Діаграма взаємодії процесів

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

КПЗ – 2026

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

Було створено блок-схеми роботи системи. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується. Блок-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.

КПЗ – 2026

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

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

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

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

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

Всередині символу записується назва процесу і передані в нього дані. Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення).

Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи). З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми.

КПЗ-2026

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

Блок-схеми показують весь процес роботи системи з підсистемами та частково доказують правильність вибраних проектних рішень. Тому від точності і детальної блок-схеми залежить результат всієї програми.

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

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

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підсистеми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю.

UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

КПЗ-2026

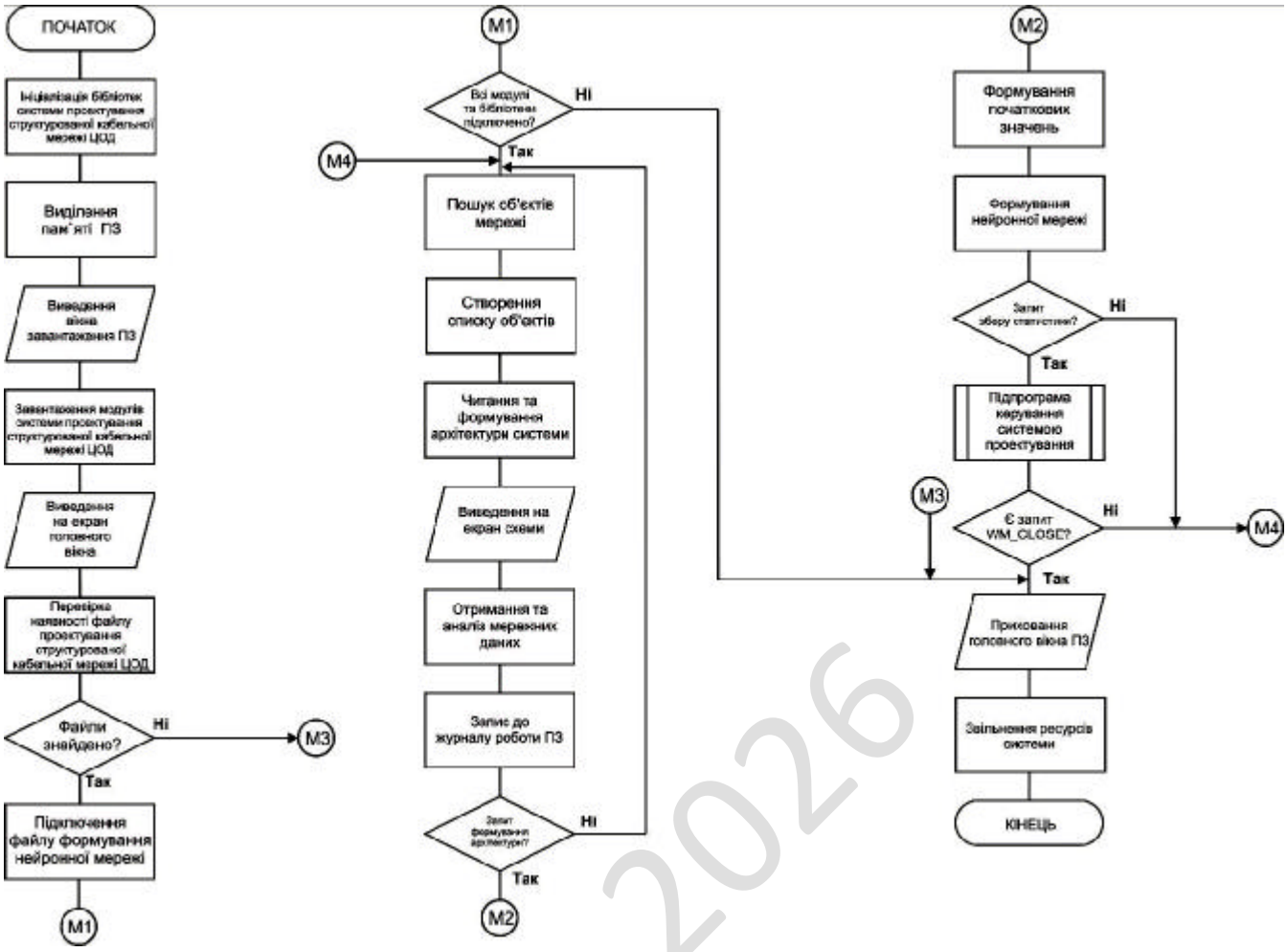


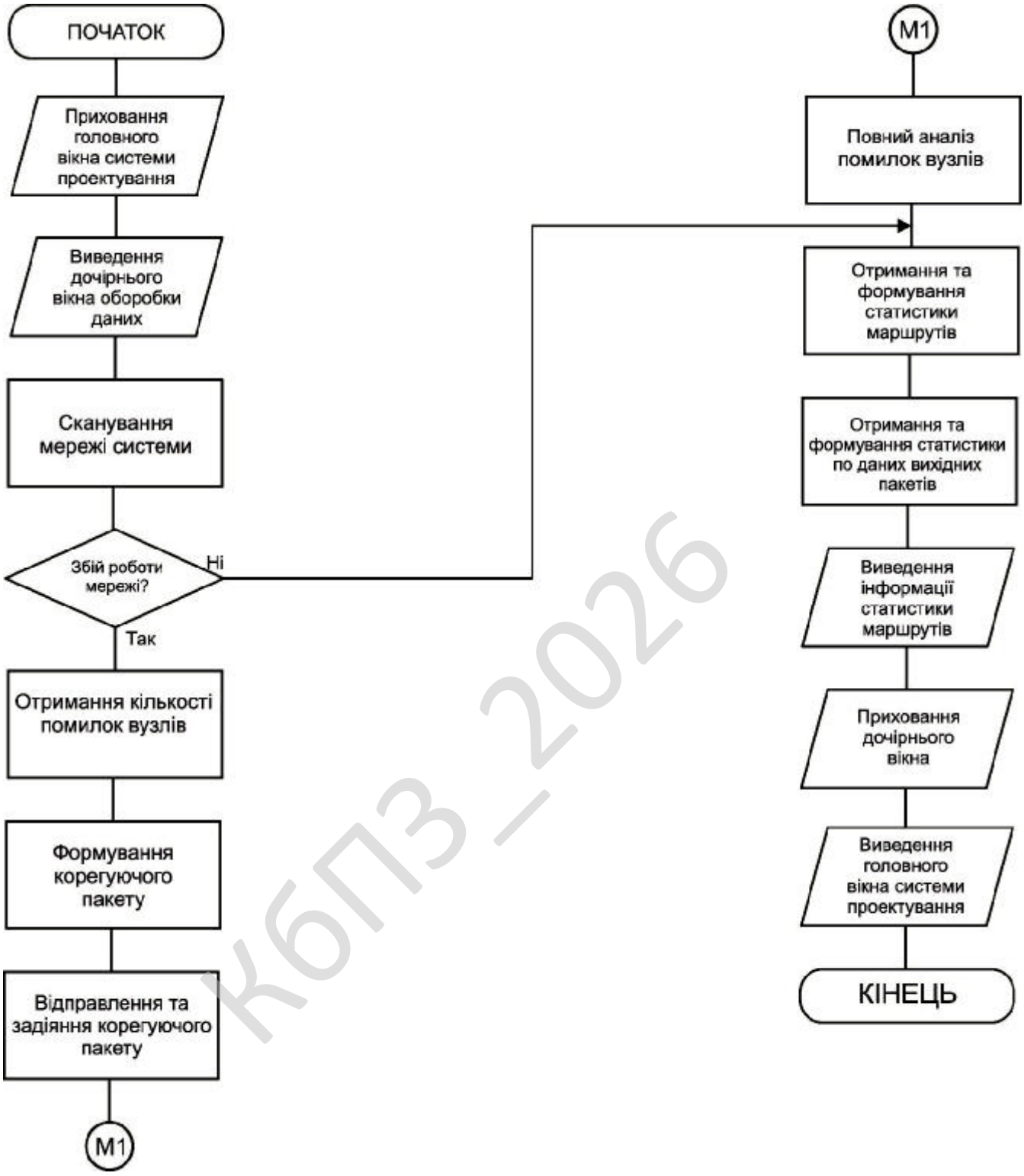
Рисунок 4.1 – Блок-схема основної програми

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

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

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії.

КОПЗ_2026



КбПЗ_2026

Рисунок 4.2 – Блок-схема роботи підпрограми

Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в

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

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

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

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

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

КОПЗ-2026

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

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

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

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

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

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

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

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

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі

КОПЗ_2026

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

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

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

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

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з

КПЗ-2026

класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

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

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

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

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

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

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

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

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

КПЗ_2026

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

Діаграма об'єктів в UML це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.

Діаграма об'єктів відображає об'єкти та зв'язки в певний момент роботи програми. Об'єкти можуть містити інформацію про власні значення а не про описання. Для відображення загальних шаблонів об'єктів та зв'язків, що можуть багаторазово створюватись під час роботи програми, слід використовувати діаграму взаємодії, яка може відображати характеристики об'єктів та зв'язків. Екземпляр діаграми взаємодії створює діаграму об'єктів.

Діаграма об'єктів не відображає еволюцію системи під час роботи. Натомість, слід використовувати діаграми взаємодії з повідомленнями, або діаграми послідовності.

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

Peer-to-peer (рівний до рівного) – варіант архітектури системи, в основі якої стоїть мережа рівноправних вузлів.

КПЗ_2026

Комп'ютерні мережі типу peer-to-peer (або P2P) засновані на принципі рівноправності учасників і характеризуються тим, що їх елементи можуть зв'язуватися між собою, на відміну від традиційної архітектури, коли лише окрема категорія учасників, яка називається серверами може надавати певні сервіси іншим.

Фраза «peer-to-peer» була вперше використана у 1984 році Парбауелом (Parbawell Yohnuhuitsman) при розробці архітектури Advanced Peer to Peer Networking фірми IBM.

В чистій «peer-to-peer» мережі не існує поняття клієнтів або серверів, лише рівні вузли, які одночасно функціонують як клієнти та сервери по відношенню до інших вузлів мережі. Ця модель мережевої взаємодії відрізняється від клієнт-серверної архітектури, в якій зв'язок відбувається лише між клієнтами та центральним сервером.

Така організація дозволяє зберігати працездатність мережі при будь-якій конфігурації доступних її учасників. Проте практикується використання P2P мереж які все ж таки мають сервери, але їх роль полягає вже не у наданні сервісів, а у підтримці інформації з приводу сервісів, що надаються клієнтами мережі.

В P2P системі автономні вузли взаємодіють з іншими автономними вузлами. Вузли є автономними в тому сенсі, що не існує загальної влади, яка може контролювати їх. В результаті автономії вузлів, вони не можуть довіряти один одному та покладатися на поведінку інших вузлів, тому проблеми масштабування та надмірності стають важливішими ніж у випадку традиційної архітектури.

Сучасні P2P-мережі набули розвитку завдяки ідеям, пов'язаними з обміном інформацією, які формувалися у руслі того, кожен вузол може надавати та отримувати ресурси які надаються будь-якими іншими учасниками. У випадку мережі Napster, це був обмін музикою, в інших випадках це може бути надання процесорного часу для пошуку інопланетних цивілізацій (SETI@home) або ліків від раку (Folding@home).

КПЗ-2026

Переваги P2P

Розподіл/зменшення вартості. Централізовані системи, які обслуговують багато клієнтів, зазвичай складають більшість вартості системи. Коли, ця вартість стає дуже великою, архітектура P2P може допомогти розподілити вартість серед користувачів. Наприклад, серед систем файлообміну Napster дозволив розподілити вартість зберігання файлів і міг підтримувати індекс, потрібний для сумісного використання. Економія коштів, здійснюється за допомогою використання та об'єднання ресурсів, які в іншому випадку не використовуються (наприклад SETI@home). Оскільки вузли зазвичай є автономними, важливо розподіляти витрати справедливо.

Об'єднання ресурсів. Децентралізований підхід веде до об'єднання ресурсів. Кожен вузол в системі P2P приносить певні ресурси як наприклад обчислювальна потужність або пам'ять. У програмах, які потребують величезну кількість цих ресурсів, як наприклад intensive моделювання або розподілені файлові системи, природно використовувати P2P, щоб залучити ці ресурси. Розподілені обчислювальні системи, як наприклад SETI@Home, distributed.net, і Endeavours – очевидні приклади цього підходу. Об'єднуючи ресурси тисяч вузлів, вони можуть виконувати важкі з точки зору кількості обчислень функції. Файлобмінні системи, як наприклад Napster, Gnutella, і так далі, також об'єднують ресурси. У цих випадках, це дисковий простір, щоб зберігати дані, та пропускна спроможність, щоб їх передавати.

Вдосконалена масштабованість/надійність. З відсутністю сильної центральної влади по відношенню до автономних вузлів, важливою метою є покращення масштабованості і надійності. Масштабованість і надійність визначаються в традиційному для розподілених систем сенсі, як наприклад використання пропускної спроможності – скільки вузлів можуть бути досягнуті від одного вузла, скільки вузлів може підтримуватися, скільки користувачів може підтримуватися. Розподілена природа peer-to-peer мереж також збільшує помилкостійкість у разі невдач, шляхом дублювання даних поміж багатьох вузлів,

КПЗ-2026

і – в чистих системах P2P – надаючи можливість вузлу знайти дані без залежності від єдиного централізованого індексного сервера. У останньому випадку, немає ніякої єдиної критичної точки в системі.

Збільшена автономія. У багатьох випадках, користувачі розподіленої системи не бажають залежати від будь-якого централізованого постачальника послуг. Натомість, вони воліють, щоб всі дані та призначена для них робота виконувалась локально. Системи P2P підтримують цей рівень автономії, тому що вони вимагають, щоб кожен вузол робив необхідну для нього частину праці.

Анонімність/конфіденційність. Пов'язаним із автономією є поняття анонімності і конфіденційності. Користувач, можливо, не хоче, щоб кого-небудь або будь-який постачальник послуг знав про нього або про його роль у системі. З центральним сервером, гарантувати анонімність важко, тому що сервер зазвичай зможе ідентифікувати клієнта, як мінімум через його адресу в Інтернет. Використовуючи структуру P2P, в якій дії виконуються локально, користувачі можуть уникати необхідності передавати будь-яку інформацію про себе до кого-небудь іншого. FreeNet – яскравий приклад того, як анонімність може вбудуватися в додаток P2P. Він пересилає повідомлення через інші вузли, щоб забезпечити неможливість вистежування початкового автора. Це збільшує анонімність, використовуючи ймовірнісні алгоритми таким чином, щоб походження не можливо було легко відстежити аналізуючи трафік у мережі.

Динамічність. Системи P2P припускають, що оточення надзвичайно динамічне. Тобто, ресурси, як наприклад вузли, з'являються та зникають із системи безперервно. У випадках комунікації, як наприклад мережі для обміну повідомленнями, використовуються так званий «список контактів», щоб інформувати користувачів, коли їхні друзі стають доступними. Без цього, потрібно було би, щоб користувачі «опитували» партнерів, посилаючи періодичні повідомлення. У випадку розподілених обчислень, як наприклад distributed.net і SETI@home, система повинна пристосуватись до заміни учасників. Тому вони повинні повторно видавати завдання для обчислення іншим учасникам, щоб

КПЗ–2026

гарантувати, що робота не втрачена, якщо попередні учасники відпадають від мережі, поки вони виконували крок обчислення.

Класифікація P2P систем

За функціями:

1. Розподілені обчислення. Обчислювальна проблема розподіляються на невеликі незалежні частини. Обробка кожної з частин робиться на індивідуальному ПК і результати збираються на центральному сервері. Цей центральний сервер відповідальний за розподілення елементів роботи серед окремих комп'ютерів в Інтернеті. Кожен із зареєстрованих користувачів має клієнтське програмне забезпечення. Воно користується періодами бездіяльності в ПК (часто це характеризується часами активації скрінсейверів), щоб виконувати деяке обчислення, надане сервером. Після того, як обчислення закінчене, результат посилається назад до сервера, і нова робота передається для клієнта.

2. Файлообмін. Зберігання та обмін даними – це одна з областей, де технологія P2P була найуспішнішою. Мультимедійні дані, наприклад, вимагають великих файлів. Napster і Gnutella використовувались користувачами, щоб обійти обмеження пропускної спроможності, які роблять передачу великих файлів неприйнятними.

3. Співпраця. Природа технології P2P робить її добре придатною для забезпечення співпраці між користувачами. Це може бути обмін повідомленнями, онлайн ігри, сумісна робота над документами в бізнесі, освіті та дома.

За ступенем централізації:

1. Чисті peer-to-peer системи. Вузли є рівними, поєднуючи ролі серверу та клієнту. Не існує центрального сервера, що керує мережею. Прикладами таких систем є Gnutella та Freenet

2. Гібридні peer-to-peer системи. Мають центральний сервер, що зберігає інформацію про вузли та відповідає на запити відносно цієї інформації. Вузли займаються забезпеченням ресурсами (тому що центральний сервер їх не має),

КОПЗ – 2026

повідомленням сервера про наявність цих ресурсів надання ресурсів іншим вузлам, які бажають ними скористатися.

В залежності від того, як вузли з'єднуються один з одним можна поділити мережі на структуровані та неструктуровані:

1. Неструктурована мережа P2P формується, коли з'єднання встановлюються довільно. Такі мережі можуть бути легко сконструйовані, оскільки новий вузол, який хоче приєднатися до мережі, може скопіювати існуючі з'єднання іншого вузла, а вже потім почати формувати свої власні. У неструктурованій мережі P2P, якщо вузол бажає знайти певні дані в мережі, запит доведеться передати майже через всю мережу, щоб охопити так багато вузлів, як можливо. Головним недоліком таких мереж є те, що запити, можливо, не завжди вирішуються. Скоріш за все популярні дані будуть доступні в багатьох вузлів та пошук швидко знайде потрібне, але якщо вузол шукає рідкісні дані, наявні лише в декількох інших вузлів, то надзвичайно малоймовірно, що пошук буде успішним. Оскільки немає ніякої кореляції між вузлами та даними, що вони зберігають, немає ніякої гарантії, що запит знайде вузол, який має бажані дані.

2. Структурована мережа P2P використовує єдиний алгоритм, щоб гарантувати, що будь-який вузол може ефективно передати запит іншому вузлу, який має бажаний файл, навіть якщо файл надзвичайно рідкісний. Така гарантія потребує структуровану систему з'єднань. У наш час найпопулярнішим типом структурованої мережі P2P є розподілені хеш-таблиці, в яких хешування використовується для встановлення зв'язку між даними та конкретним вузлом, який за них відповідає.

Розглянемо визначення API. Це прикладни́й програ́мний інтерфе́йс (Application Programming Interface, API) – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

Спрощено – це набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного

КПЗ-2026

забезпечення. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

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

Високорівневі API часто програють у гнучкості. Виконання деяких функцій нижчого рівня стає набагато складнішим, або навіть неможливим.

В об'єктно-орієнтованих мовах, прикладний програмний інтерфейс зазвичай включає в себе опис набору визначень класу, з набором форм поведінки, пов'язаних з цими класами. Це абстрактне поняття пов'язане з реальними функціями, які надані або надаватимуться, класами, які реалізуються в методах класу.

Прикладний програмний інтерфейс в даному випадку можна розглядати як сукупність всіх методів, які публічно доступні в класах (зазвичай званий інтерфейс класу). Це означає, що прикладний програмний інтерфейс вказує методи, за допомогою яких взаємодіє з об'єктами, отриманими з визначень класів і обробляє їх.

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

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

У цьому випадку Прикладний Програмний Інтерфейс може бути інтерпретованим як два методи pop() і push(), або, більш широко,

КбПЗ – 2026

використовується варіант, коли можна використовувати елемент типу Stack, який реалізує поведінку стека, надаючи йому можливість для додавання / видалення елементів з вершини. Друга інтерпретація видається більш доречною в дусі об'єктно-орієнтованого підходу.

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

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

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

Прикладний програмний інтерфейс може бути також реалізацією протоколу.

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

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

КПЗ-2026

мовах, якщо виклики функції не будуть перетворені з конкретної адаптації бібліотеки.

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

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

При використанні прикладного програмного інтерфейсу в контексті веб-розробки, як правило, ППІ визначається набором повідомлень запиту HTTP, також визначається структура повідомлень-відповідей, зазвичай у розширенні мови розмітки XML або в форматі об'єктного запису JavaScript (JSON). У той час як прикладний програмний інтерфейс у Web історично був практично синонімом для веб-служби, останнім часом тенденція змінилась (так званий Web 2.0) на відхід від Simple Object Access Protocol (SOAP) на основі веб-сервісів і сервіс-орієнтованої архітектури (SOA) на більш прямі передачі репрезентативного стану (REST) стилів веб-ресурсів та ресурсів-орієнтованої архітектури (ROA).

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

NetBIOS (Network Basic Input/Output System) – протокол для роботи в локальних мережах на персональних ЕОМ типу IBM/PC, розроблений у вигляді інтерфейсу, який не залежить від фірми–виробника. Був розроблений фірмою Sytek Corporation за замовленням IBM в 1983 році. Він включає в себе інтерфейс сеансового рівня (англ. NetBIOS interface), в якості транспортних протоколів використовує TCP і UDP.

КПЗ–2026

Особливістю NetBIOS є можливість його роботи поверх різних протоколів, найпоширенішими/відомими з яких є NetBEUI, IPX і стек протоколів TCP/IP; причому якщо старі версії Windows орієнтувалися на більш легкі в реалізації і менш ресурсомісткі NetBEUI і IPX, то сучасні Windows орієнтуються на TCP/IP.

При використанні NetBEUI і IPX NetBIOS сам забезпечує надійність доставки даних (функціональність SPX не використовувати), а при використанні TCP/IP надійність доставки забезпечує TCP, за що удостоївся окремого імені «NBT».

Інтерфейс NetBIOS являє собою типовий інтерфейс взаємодії програм (API) для забезпечення мережевих операцій введення–виведення і управління транспортним протоколом.

Програми, що використовують NetBIOS API інтерфейс, можуть працювати тільки при наявності протоколу, що допускає використання такого інтерфейсу.

NetBIOS також визначає протокол, що функціонує на сеансовому/транспортному рівнях моделі OSI. Цей протокол використовується протоколами нижчих рівнів, такими як NBFP (NetBEUI) і NetBT для виконання мережевих запитів вводу-виводу і операцій, описаних в стандартному интерфейсном наборі команд NetBIOS.

Тобто, NetBIOS сам не підтримує виконання файлових операцій. Ця функція покладається на протоколи нижчих рівнів, а сам NetBIOS забезпечує тільки зв'язок з цими протоколами і NetBIOS API–інтерфейс.

Пакет NETBIOS створений для використання групою ЕОМ. Підтримує як режим сесій (робота через з'єднання), так і режим дейтаграм (без встановлення з'єднання). 16–символьні імена об'єктів в NETBIOS розподіляються динамічно.

NETBIOS має власну DNS, яка може взаємодіяти з інтернетівський. Ім'я об'єкта при роботі з NETBIOS не може починатися з символу *.

Додатки можуть знайти через NETBIOS потрібні їм ресурси, встановити зв'язок і послати або отримати інформацію. NETBIOS використовує для служби імен порт 137, для служби дейтаграм – порт 138, а для сесій – порт 139. Будь–яка

КОПІЯ – 2026

сесія починається з NETBIOS–запиту, завдання IP–адреси і визначення TCP–порту віддаленого об'єкта, далі йде обмін NETBIOS–повідомленнями, після чого сесія закривається.

Сесія здійснює обмін інформацією між двома NETBIOS – додатками. Довжина повідомлення лежить в межах від 0 до 131 071 байт. Припустимо одночасне встановлення декількох сесій між двома об'єктами.

При організації IP–транспорту через NETBIOS IP–дейтаграмма вкладається в NETBIOS–пакет. Інформаційний обмін відбувається в цьому випадку без встановлення зв'язку між об'єктами. Імена NETBIOS повинні містити в собі IP–адреси. Так, частина NETBIOS–адреси може мати вигляд IP.XX.XX. XX. XX, де IP вказує на тип операції (IP через Netbios), а XX. XX. XX.XX – IP– адреса.

Система NETBIOS має власну систему команд (call, listen, hang up, send, receive, session status, reset, cancel, adapter status, unlink, remote program load) і примітивів для роботи з дейтаграммами (send datagram, send broadcast datagram, receive datagram , receive broadcast datagram).

Всі кінцеві вузли NETBIOS діляться на три типи:

– широкомовні («b») вузли;

– вузли точка–точка («p»);

– вузли змішаного типу («m»).

IP–адреса може асоціюватися з одним із зазначених типів. B–вузли встановлюють зв'язок зі своїм партнером за допомогою широкомовних запитів. P– і M–вузли для цієї мети використовують netbios сервер імен (NBNS) і сервер розподілу дейтаграм (NBDD).

NetBIOS забезпечує:

– реєстрацію і перевірку мережевих імен;

– встановлення і розрив з'єднань;

– зв'язок з підтвердженням доставки інформації;

– зв'язок без підтвердження доставки інформації;

– підтримку управління і моніторингу драйвера і мережевої карти.

КОПЗ_2026

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

Гнучка́ розробка програмного забезпечення (Agile software development, agile-методи) – клас методологій розробки програмного забезпечення, що базується на ітеративній розробці, в якій вимоги та розв'язки еволюціонують через співпрацю між самоорганізовуваними багатофункціональними командами.

Гнучка розробка – найкращий засіб для підвищення продуктивності розробників програмного забезпечення.

Більшість гнучких методологій націлені на мінімізацію ризиків, шляхом зведення розробки до серії коротких циклів, що мають назву ітерацій, які зазвичай тривають один-два тижні.

Кожна ітерація сама по собі виглядає як програмний проект в мініатюрі, і включає всі завдання, необхідні для видачі мінімального приросту за функціональністю: планування, аналіз вимог, проектування, кодування, тестування і документування.

Хоча окрема ітерація, як правило, недостатня для випуску нової версії продукту, мається на увазі те, що гнучкий програмний проект готовий до випуску наприкінці кожної ітерації. Після закінчення кожної ітерації, команда виконує переоцінку пріоритетів розробки.

Agile акцентує увагу на безпосередньому спілкуванні «віч-на-віч». Більшість agile команд розташовані в одному офісі, його іноді називають bullpen. Як мінімум вона включає і «замовників» (замовники, які визначають продукт, також це можуть бути менеджери продукту, бізнес аналітики або клієнти). Офіс може також включати тестувальників, дизайнерів інтерфейсу, технічних авторів і менеджерів.

Основною метрикою agile методів є робочий продукт. Віддаючи перевагу безпосередньому спілкуванню, agile-методи зменшують обсяг письмової документації в порівнянні з іншими методами. Це привело до критики цих методів як недисциплінованих.

КПЗ-2026

Agile – родина процесів розробки, а не єдиний підхід в розробці програмного забезпечення, і визначається Agile Manifesto. Agile не включає практик, а визначає цінності та принципи, якими керуються успішні команди.

Agile Manifesto розроблений і прийнятий 17 розробниками 11-13 лютого 2001 року на лижному курорті The Lodge at Snowbird в горах Юти. Маніфест підписали представники наступних методологій Extreme programming, Scrum, DSDM, Adaptive software development, Crystal Clear, Feature driven development, Pragmatic Programming. Agile Manifesto містить 4 основні ідеї та 12 принципів. Примітно, що Agile Manifesto не містить практичних порад.

Основні ідеї:

– Особистості та їхні взаємодії важливіші, ніж процеси та інструменти;

– Робоче програмне забезпечення важливіше, ніж повна документація;

– Співпраця із замовником важливіша, ніж контрактні зобов'язання;

– Реакція на зміни важливіша, ніж дотримання плану.

Принципи, які роз'яснює Agile Manifesto:

– задоволення клієнта за рахунок ранньої та безперебійної поставки коштовного програмного забезпечення;

– вітання змін вимог навіть наприкінці розробки (це може підвищити конкурентоспроможність отриманого продукту);

– часта поставка робочого програмного забезпечення (кожен місяць або тиждень або ще частіше);

– тісне, щоденне спілкування замовника з розробниками впродовж всього проекту;

– проектом займаються мотивовані особистості, які забезпечені потрібними умовами роботи, підтримкою і довірою;

– рекомендований метод передачі інформації – особиста розмова (віч-на-віч);

– робоче програмне забезпечення – найкращий вимірювач прогресу;

КПЗ_2026

– спонсори, розробники та користувачі повинні мати можливість підтримувати постійний темп на невизначений термін;

– постійну увагу поліпшенню технічної майстерності та зручному дизайну;

– простота – мистецтво не робити зайвої роботи;

– найкращі технічні вимоги, дизайн та архітектура виходять у самоорганізованої команди;

– постійна адаптація до мінливих обставин.

Маніфест та Принципи гнучкої розробки містять високорівневі ідеї щодо того, як потрібно вибудовувати процес розробки програмного забезпечення, щоб успішно завершувати проекти й створювати команди, в яких приємно та цікаво працювати.

Документи визначають, що потрібно для цього зробити, але не говорять, як це зробити. По-іншому й не могло бути, оскільки Маніфест та Принципи народилися внаслідок консенсусу представників різних (хоча й споріднених) напрямів, які могли знайти спільну основу лише на рівні базових цінностей та принципів.

Критика. Багато керівників проектів, що працюють у традиційних методологіях на кшталт «водоспаду», критикують agile-методи.

Один з повторюваних пунктів критики: при agile-підході часто нехтують створенням «дорожньої карти» розвитку продукту, так само як і управлінням вимогами, в процесі якого і формується така «карта». Гнучкий підхід до управління вимогами не має на увазі далекосяжних планів (по суті, управління вимогами просто не існує в даній методології), а має на увазі можливість замовника раптом і несподівано наприкінці кожної ітерації виставляти нові вимоги, що часто суперечать архітектурі вже створеного і поставленого продукту. Таке іноді призводить до катастрофічних «авралів» з масовим рефакторингом і переробками практично на кожній черговій ітерації.

Крім того вважається, що робота в agile мотивує розробників вирішувати всі прибулі завдання найпростішим і найшвидшим можливим способом, при

КПЗ – 2026

цьому часто не звертаючи уваги на коректність коду з точки зору вимог базової платформи (підхід «працює, та й добре», при цьому не враховується, що може перестати працювати при найменшій зміні або ж породити важкі до відтворення дефекти після реального розгортання у клієнта). Це призводить до зниження якості продукту і накопиченню дефектів.

Методології. Існують методології, які дотримуються цінностей і принципів заявлених в Agile Manifesto, деякі з них:

1. Agile Modeling – набір понять, принципів і прийомів (практик), що дозволяють швидко і просто виконувати моделювання і документування в проектах розробки програмного забезпечення. Не включає в себе детальну інструкцію з проектування, не містить описів, як будувати діаграми на UML.

Основна мета – ефективне моделювання і документування; але не охоплює програмування та тестування, не включає питання управління проектом, розгортання і супроводу системи. Однак включає в себе перевірку моделі кодом.

2. Agile Unified Process (AUP) спрощена версія IBM Rational Unified Process (RUP), розроблена Скоттом Амблером, яка описує просте і зрозуміле наближення (модель) для створення програмного забезпечення для бізнес-додатків.

3 Agile Data Method – група ітеративних методів розробки програмного забезпечення, в яких вимоги та рішення досягаються в рамках співпраці різних крос-функціональних команд.

4. DSDM заснований на концепції швидкої розробки додатків (Rapid Application Development, RAD). Являє собою ітеративний і інкрементний підхід, який надає особливого значення тривалій участі в процесі користувача/споживача.

5. Essential Unified Process (EssUP).

6. Екстремальне програмування (Extreme programming, XP).

7. Feature driven development (FDD) – функціонально-орієнтована розробка. Використовуване в FDD поняття функції або властивості (feature) Системи досить

КБПЗ-2026

близько до поняття прецеденту використання, використовуваному в RUP, істотна відмінність – це додаткове обмеження: «кожна функція повинна допускати реалізацію не більше, ніж за два тижні». Тобто якщо сценарій використання досить малий, його можна вважати функцією. Якщо ж великий, то його треба розбити на декілька відносно незалежних функцій.

8. Getting Real – ітераційний підхід без функціональних специфікацій, що використовується для веб-додатків. У даному методі спершу розробляється інтерфейс програми, а потім її функціональна частина.

9. OpenUP – це ітераційно-інкрементний метод розробки програмного забезпечення. Позиціюється, як легкий і гнучкий варіант RUP. OpenUP ділить життєвий цикл проекту на чотири фази: початкова фаза, фази уточнення, конструювання та передачі. Життєвий цикл проекту забезпечує надання зацікавленим особам та членам колективу точок ознайомлення і прийняття рішень впродовж усього проекту. Це дозволяє ефективно контролювати ситуацію і вчасно приймати рішення про задовільність результатів. План проекту визначає життєвий цикл, а кінцевим результатом є остаточний додаток.

10. Scrum встановлює правила керування процесом розробки та дозволяє використовувати вже існуючі практики кодування, коректуючи вимоги або вносячи тактичні зміни. Використання цієї методології дає можливість виявляти і усувати відхилення від бажаного результату на більш ранніх етапах розробки програмного продукту.

11. Бережлива розробка програмного забезпечення (lean software development). Використовує підходи з концепції бережливого виробництва.

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

Екстремальне програмування (Extreme Programming, далі XP) це методологія розробки програмного забезпечення, найпопулярніша серед так званих гнучких методологій. Має на меті поліпшення якості програмного

КОПЗ-2026

забезпечення та чутливість до змін у вимогах замовників. Як вид гнучких методологій, XP радить часті "випуски" програми у коротких циклах розробки, що має на меті поліпшити продуктивність праці та покращити можливості виконання вимог замовника що змінюються. Авторами даної методології є Кент Бек, Ворд Каннінгем, Мартін Фаулер та інші.

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

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

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

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

Хоча саме екстремальне програмування є відносно новим, багато її практик вже існували і використовувались протягом певного часу; однак, методологія підносить "найкращі практики" до екстремального рівня. Для прикладу, практика по плануванню і написанню тестів перед написанням кожної маленької частини коду було використано раніше в проекті НАСА "Меркурій". Для зменшення часу на розробку ПЗ деякі формальні документи тестування (такі

КОПЗ-2026

як приймальне тестування) писались паралельно (або й раніше) з написанням самого ПЗ. Незалежна група тестування НАСА може писати процедури тестування базуючись на формальних вимогах до продукту до того як програмне забезпечення розроблене та інтегроване в систему. В XP ця концепція піднесена до "екстремального рівня" завдяки написанню автоматичних тестів які перевіряють поведінку навіть малих частинок коду, а не тільки значних функціональних частин ПЗ.

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

– Спроба примирити гуманність і продуктивність.

– Механізм для соціальної зміни.

– Шлях до удосконалення.

– Стиль розвитку.

Дисципліна розробки програмного забезпечення.

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

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

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

XP намагається зменшити ціну зміни вимог до ПЗ завдяки малим циклам розробки, а не одним довгим циклом. Екстремальне програмування сприймає зміни до вимог як звичайні, неминучі та бажані аспекти розробки ПЗ, і ці зміни

КПЗ–2026

мають бути очікуваним. Основна ідея полягає в тому що неможливо розробити самодостатній пакет вимог до ПЗ, зміни в вимогах – неминучі.

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

Екстремальне програмування описує чотири базові активності що виконуються при розробці програмного забезпечення: написання коду, тестування, слухання та дизайн. Написання коду. Прихильники XP заявляють що єдиним дійсно важливим результатом розробки ПЗ є код: без готового коду нема продукту.

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

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

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

1.1. Розробка через тестування (Test driven development).

1.2 Гра в планування (Planning game).

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

1.4 Парне програмування (Pair programming).

2. Безперервний, а не пакетний процес.

2.1 Безперервна інтеграція (Continuous Integration).

2.2 Рефакторинг (Design Improvement, Refactor).

2.3 Часті невеликі релізи (Small Releases).

3. Розуміння, що поділяється всіма учасниками.

3.1 Простота (Simple design).

3.2 Метафора системи (System metaphor).

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

КПЗ-2026

3.4 Стандарт кодування (Coding standard or Coding conventions).

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

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Twofish, який є симетричним алгоритмом блочного шифрування з розміром блоку 128 біт і довжиною ключа до 256 біт. Число раундів 16. Розроблено групою фахівців на чолі з Брюсом Шнайером. Був одним з п'яти фіналістів другого етапу конкурсу AES. Алгоритм розроблений на основі алгоритмів Blowfish, SAFER і Square.

Відмінними особливостями алгоритму є використання попередньо обчислюваних та залежних від ключа S-box'ів і складна схема розгортки підключення шифрування. Половина n-бітного ключа шифрування використовується як власне ключ шифрування, інша – для модифікації алгоритму (від неї залежать S-box'и).

Twofish розроблявся спеціально з урахуванням вимог та рекомендацій NIST для конкурсу AES [1]:

– 128-бітний блочний симетричний шифр.

– Довжина ключів 128, 192 і 256 біт.

– Відсутність слабких ключів.

– Ефективна програмна (в першу чергу на 32-бітних процесорах) та апаратна реалізація.

– Гнучкість (можливість використання додаткових довжин ключа, використання в поточному шифруванні, хеш-функціях і т.д.).

– Простота алгоритму – для можливості його ефективного аналізу.

Однак саме складність структури алгоритму і, відповідно, складність його аналізу на предмет слабких ключів або прихованих зв'язків, а також досить

КПЗ_2026

повільне час шифрування порівняно з Rijndael на більшості платформ, зіграло не на його користь.[2]

Алгоритм Twofish виник в результаті спроби модифіковані алгоритм Blowfish для 128-бітового вхідного блоку. Новий алгоритм повинен був бути легко реалізованим апаратно (у тому числі використовувати таблиці меншого розміру), мати досконалішу систему розширення ключа (key schedule) і мати однозначну функцію F.

В результаті, алгоритм був реалізований у вигляді змішаної мережі Фейстеля з чотирма гілками, які модифікують один одну з використанням криптоперетворень Адамара (Pseudo-Hadamar Transform, PHT).

Можливість ефективної реалізації на сучасних (для того часу) 32-бітних процесорах (а також в смарт-картах і подібних пристроях) – один з ключових принципів, яким керувалися розробники Twofish.

Наприклад, у функції F при обчисленні PHT і складання з частиною ключа K навмисно використовується додавання, замість традиційного xor. Це дає можливість використовувати команду LEA сімейства процесорів Pentium, яка за один такт дозволяє обчислити перетворення Адамара. (Правда в такому випадку код доводиться компілювати під конкретне значення ключа).

Алгоритм Twofish не запатентований і може бути використаний ким завгодно без будь-якої плати або відрахувань. Він використовується в багатьох програмах шифрування, хоча і отримав менше поширення, ніж Blowfish.

Опис алгоритму

Twofish розбиває вхідний 128-бітний блок даних на чотири 32-бітних підблоки, над якими, після процедури вхідного відбілювання (input whitening), проводиться 16 раундів перетворень. Після останнього раунду виконується вихідна відбілювання (output whitening).

Відбілювання (whitening)

Відбілювання – це процедура xor'a даних з підключами перед першим раундом і після останнього раунду. Вперше ця техніка була використана в шифрі

КОПЗ-2026

Khufu/Khare і, незалежно, Рональдом Рівестом (англ. Ron Rivest) в алгоритмі шифрування DESX. Joe Killian (NEC) і Phillip Rogaway (Каліфорнійський університет) показали, що відбілювання дійсно ускладнює завдання пошуку ключа повним перебором (exhaustive key-search) в DESX[3] . Розробники Twofish стверджують[4], що в Twofish відбілювання також істотно ускладнює завдання підбору ключа, оскільки криптоаналітика не може дізнатися, які дані потрапляють на вхід функції F першого раунду.

Тим не менш, виявилися і негативні сторони. Цікаве дослідження провели фахівці дослідницького центру компанії IBM.[5] Вони виконали реалізацію алгоритму Twofish для типової смарт-карти з CMOS-архітектурою і проаналізували можливість атаки за допомогою диференціального аналізу споживаної потужності (DPA – Differential Power Analysis). Атаці піддавалася саме процедура вхідного вибілювання, оскільки вона безпосередньо використовує xor підключів з вхідними даними. У результаті дослідники показали, що можна повністю обчислити 128-бітовий ключ проаналізувавши всього 100 операцій шифрування довільних блоків.

Функція g

Функція g – основа алгоритму Twofish. На вхід функції подається 32-бітове число X, яке потім розбивається на чотири байти x0, x1, x2, x3. Кожен з вийшов байтів пропускається через свій S-box. (Слід зазначити, що S-box'и в алгоритмі не фіксовані, а залежать від ключа). Отримані 4 байти на виходах S-box'ов інтерпретуються як вектор з чотирма компонентами. Цей вектор множиться на фіксовану матрицю MDS (maximum distance separable) розміром 4x4, причому обчислення проводяться в скінченному полі по модулю непривідного многочлена

MDS матриця – це така матриця над кінцевим полем K, що якщо взяти її як матрицю лінійного перетворення з простору у простір , то будь-які два вектори з простору виду (x, f (x)) будуть мати як мінімум m+1 відмінностей в компонентах. Тобто набір векторів вигляду (x, f (x)) утворює код, що володіє

КПЗ_2026

властивістю максимального рознесення (maximum distance separable code). Таким кодом, наприклад, є код Ріда-Соломона.

В Twofish властивість максимальної рознесеність матриці MDS означає, що загальна кількість змінних байт вектора a і вектора не менше п'яти. Іншими словами, будь-яка зміна тільки одного байта в a призводить до зміни всіх чотирьох байтів в b.

Криптоперетворення Адамара (Pseudo-Hadamar Transform, PHT)

Криптоперетворення Адамара – оборотне перетворення бітового рядка довжиною 2n. Рядок розбивається на дві частини a і b однакової довжини в n біт. Перетворення обчислюється таким чином:

)

)

Ця операція часто використовується для «розсіювання» коду (наприклад в шифрі SAFER).

В Twofish це перетворення використовується при змішуванні результатів двох g-функцій (n = 32).

Циклічний зсув на 1 біт

У кожному раунді два правих 32-бітових блоки, які xor-яться з результатами функції F, додатково циклічно зрушуються на один біт. Третій блок зрушується до операції xor, четвертий блок – після. Ці зрушення спеціально додані, щоб порушити вирівнювання по байтах, яке властиво S-box'ам та операції множення на MDS-матрицю. Проте шифр перестає бути повністю симетричним, так як при шифруванні й розшифровці зрушення слід здійснювати в протилежні сторони.

Генерація ключів

Twofish розрахований на роботу з ключами довжиною 128, 192 і 256 біт. З вихідного ключа генерується 40 32-бітних підключів, перші вісім з яких використовуються тільки в операціях вхідного і вихідного вибілювання, а решта 32 – в раундах шифрування, по два підключі на раунд. Особливістю Twofish є те,

КБПЗ_2026

що вихідний ключ використовується також і для зміни самого алгоритму шифрування, так як використовуються у функції g S-box'и не фіксовані, а залежать від ключа.

Для формування раундових підключів вихідний ключ M розбивається з перестановкою байт на два однакові блоки M_o і M_e. Потім за допомогою блоку M_o і функції h шифрується значення 2 * i, а за допомогою блоку M_e шифрується значення 2*i+1, де i – номер поточного раунду (0 – 15). Отримані зашифровані блоки змішуються криптоперетвореням Адамара, і потім використовуються як раундові підключі.

КБПЗ_2026

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у бакалаврської дипломної роботі система проектування структурованої кабельної мережі ЦОД. 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.

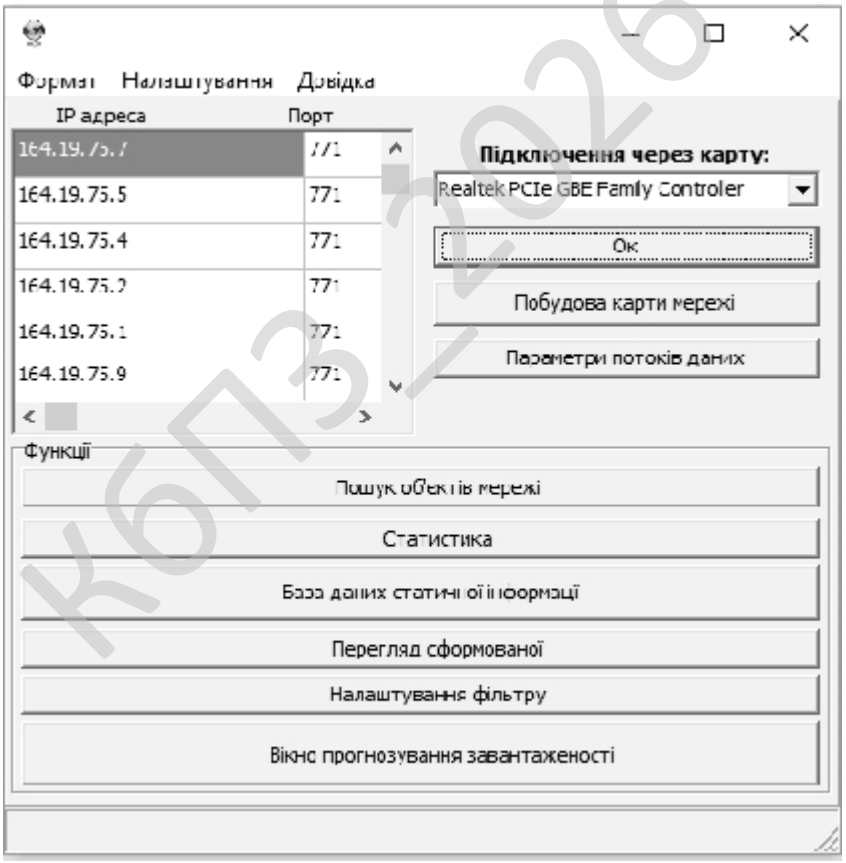


Рисунок 5.1 – Головне вікно ПЗ

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.

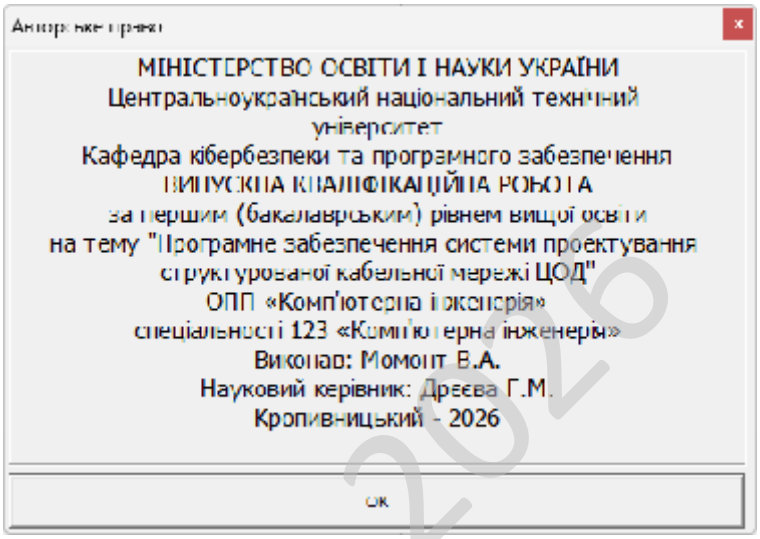


Рисунок 5.2 – Авторське право

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

КБПЗ_2026

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 IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT 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 IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути

КПЗ-2026

невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

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

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

КБПЗ-2026

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

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

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

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

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

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

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

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.

КПЗ-2026

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 ITc).

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

Будь-який спосіб тестування «чорної скриньки» повинен:

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

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

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

КБПЗ_2026

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.

КбПЗ – 2026

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи проектування структурованої кабельної мережі ЦОД.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем проектування структурованої кабельної мережі ЦОД.

– Досліджена система проектування структурованої кабельної мережі ЦОД.

– На основі отриманих результатів досліджень створена програмна реалізація системи проектування структурованої кабельної мережі ЦОД.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання проектування структурованої кабельної мережі ЦОД.

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 проектування структурованої кабельної мережі ЦОД. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на

КОПЗ-2026

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 Twofish.

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.

КБПЗ_2026

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

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

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

КПЗ-2026

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

КОПІЯ 2026

15. 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.

16. 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.

17. 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.

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

19. 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.

20. 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.

21. 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

КБПЗ_2026

Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

КБПЗ-2026

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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

36. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у

КБПЗ_2026

комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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).

43. 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

КБПЗ-2026

44. 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.

45. 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.

46. 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.

47. 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.

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

49. 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.

50. 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.

КОПЗ_2026