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 IoT у дата-центрах”

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

Керівник проекту
доктор технічних наук, професор
_____ Коваленко О.В.
«____» _____ 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

Колюці Артему Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи IoT у дата-центрах*
2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, професор*
(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 IoT у дата-центрах*
5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   1. Призначення та область використання.
   2.Перегляд аналогічних існуючих систем.
   3. Опис і обґрунтування проектних рішень.
   4. Етапи програмування системи.
   5. Впровадження системи в промислову експлуатацію.
   6. Висновки
6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 27 » лютого 2026 р.

Підпис керівника
_____ Коваленко О.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 27 » лютого 2026 р.

Підпис здобувача
_____ Колюка А.А.
(прізвище та ініціали)

АНОТАЦІЯ

Колюка А.А. Програмне забезпечення системи IoT у дата-центрах. 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 IoT у дата-центрах.

Метою розробки є програмне забезпечення системи IoT у дата-центрах.

Результат роботи – програмна реалізація системи IoT у дата-центрах.

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, IoT, дата-центри

ABSTRACT

Koliuka A.A. Software for IoT systems in data centers. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for the IoT system in data centers.

The purpose of the development is the software for the IoT system in data centers.

The result of the work is the software implementation of the IoT system in data centers.

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

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

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

The program was developed in the Python environment.

Keywords: computer engineering, IoT, data centers

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

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

ВСТУП

Актуальність теми. Технології інтернету речей (IoT) відкривають широкі можливості для значного підвищення рівня автоматизації керування й оптимізації процесів у центрах обробки даних. Те, що сьогодні прийнято відносити до компетенції Інтернету речей, у тім або іншому ступені вже закладений або так чи інакше з'являється в сучасних програмних комплексах керування інфраструктурою дата-центрів (центрів обробки даних (ЦОД)) – DCIM.

Інтернет речей, про яке сьогодні так багато говорять, здатний вплинути на центри обробки даних. Масове впровадження рішень на основі IoT-технологій веде не тільки до росту потреби в комп'ютерних ресурсах і збільшенню обсягів збережених у ЦОД даних, але й до певних змін архітектурних концепцій самих центрів. Прагнення забезпечити мінімізацію затримок при обробці вступників у реальному часі даних привело до створення концепції «мрячних» обчислень і граничних ЦОД.

У той же час технології IoT відкривають широкі можливості для значного підвищення рівня автоматизації керування й ефективності оптимізації процесів у центрах обробки даних. Експерти називають ЦОД досить перспективними об'єктами для цифрової трансформації на базі технологій Інтернету речей.

Впровадження IoT у ЦОД – це не тільки установка численних датчиків для виміру життєво важливих показників функціонування встаткування. Для їхнього збору, зберігання й обробки потрібна розвинена інфраструктура, щоб за допомогою засобів предиктивної аналітики аналізувати те, що відбувається в інфраструктурі ЦОД події й прогнозувати їхній можливий розвиток.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи IoT у дата-центрах.

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

– Огляд існуючих систем IoT у дата-центрах.

– Дослідження системи IoT у дата-центрах.

– Програмна реалізація системи IoT у дата-центрах.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі IoT у дата-центрах.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи IoT у дата-центрах, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Методологія Інтернету речей дозволить впроваджувати моделі керування, засновані на даних (Data-Driven Model), використовувати для вивчення й розуміння процесів статистичну інформацію, що характеризує роботу багатьох ЦОД, застосовуючи для її аналізу хмарні ресурси.

Керування на основі даних стає одним з перспективних напрямків розвитку сучасних центрів обробки даних. Такі системи здатні підвищити стабільність роботи ЦОД, розширити гарантії в угодах про якість обслуговування (Service Level Agreement), зменшити завантаження експлуатаційного персоналу.

Ті можливості, які сьогодні відносять до Інтернету речей, у тім або іншому ступені вже закладені або з'являються в сучасних програмних комплексах керування інфраструктурою ЦОД (Data Center Infrastructure Management, DCIM), які забезпечують «видимість» і планування використовуваних ресурсів, оптимізацію температурних режимів і енерговитрат, а також зниження сукупної вартості володіння ЦОД.

Оскільки ЦОД всі частіше стають великими індустріальними об'єктами, у продуктах і технологіях для моніторингу інфраструктури й керування різними системами зацікавлені й розроблювачі комплексів DCIM, і власники комерційних центрів обробки даних (КЦОД). І ті й інші пропонують свої рішення.

1.2 Область застосування

Довгий час системи DCIM здобувалися головним чином для найбільше «зрілих» ЦОД великих підприємств. Тепер, коли загальноприйнятої стає модель оренди IT-ресурсів, а підтримка власних ЦОД виявляється усе менш вигідною,

найбільший інтерес до повномасштабних комплексів DCIM проявляють провайдери послуг колокейшн і власники великих хмарних ЦОД.

Оператори ЦОД намагаються використовувати розвинену функціональність систем DCIM для надання своїм клієнтам додаткових послуг, одержання доданої вартості й залучення нових замовників. Постачальники хмарних сервісів воліють використовувати не доступні – з «полки» – комерційні рішення, а власні продукти, які розробляються з обліком вартих перед підприємством завдань.

Поява численних граничних мікроЦОД, до поширення яких, як очікується, приведе розгортання платформ IoT, породжує потреба в більше зроблених рішеннях для керування інфраструктурою ЦОД, у тому числі з можливістю їхнього використання в децентралізованих середовищах.

Свою лепту в поширення й розвиток систем керування інфраструктурою вносять і модулі високої заводської готовності (Prefabricated Data Center Modules), значно прискорювальне створення великих ЦОД. Такі модулі, складання й налагодження яких здійснюються в заводських умовах, містять елементи DCIM, що доводиться враховувати при їхньому впровадженні у великих комерційних ЦОД.

Безумовно, на розвитку сучасних систем DCIM не може не відбиватися проникнення в центри обробки даних технологій штучного інтелекту й машинного навчання. Розроблювачі комплексів DCIM і оператори ЦОД створюють і пропонують рішення, що відповідають – у тім або іншому ступені – сучасним тенденціям і враховуючі можливості новітніх технологій.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи IoT у дата-центрах, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Огляд рішення IoT CE on cloud

Рішення IBM® категорії SaaS (ПЗ як послуга) надають виділене віртуальне приватне хмарне середовище з застосунками, які необхідні вашим розроблювачам і інженерам, щоб стимулювати подальший розвиток бізнесу.

Рішення SaaS спроектовані для надання програмних продуктів IBM у хмарному середовищі. Фахівці з рішень IBM SaaS надають весь комплекс послуг із планування конфігурації й реалізації програмних продуктів IBM, забезпечуючи безперервний доступ до інфраструктури, застосунку й підтримці. Ви можете вибрати ті продукти, які повинні бути включені до складу рішення, а фахівці SaaS розгорнуть їх у ЦОД IBM і забезпечать моніторинг їхньої роботи, гарантуючи постійну готовність.

Виберіть ті рішення, які необхідні вашим фахівцям:

– Глобальна конфігурація – надається рішенням IBM Internet of Things Continuous Engineering (IoT CE) on Cloud, до складу якого входять продукти Rational solution for CLM, Rational Engineering Lifecycle Manager, Rational DOORS Next Generation і Rational Rhapsody Design Manager.

– IBM Collaborative Lifecycle Management on Cloud.

– Керування вимогами – надається рішенням IBM DOORS Next Generation on Cloud.

– Керування якістю – надається рішенням IBM Quality Manager on Cloud.

– Керування змінами й конфігурацією – надається рішенням IBM Team Concert on Cloud.

– Керування проектуванням – надається рішенням Rational Design Manager on Cloud.

– Data Collection Component – надається рішенням IBM Engineering Lifecycle Manager on Cloud/

Огляд

IBM IoT CE on Cloud працює на ресурсах IBM у безпечному ЦОД SoftLayer®. Фахівці IBM забезпечують готовність на рівні 99,9%, а також масштабованість рішення для підтримки декількох тисяч користувачів. Фахівці IBM виконують моніторинг роботи середовища для забезпечення постійної готовності й все необхідне обслуговування середовища, у тому числі установку виправлень і додаткових відновлень у зручне для клієнта час.

Рішення CLM on Cloud надають виділене середовище для ПЗ IBM. Фахівці IBM розвертають продукти IBM в IBM SoftLayer Cloud Service Center і виконують моніторинг роботи рішення для забезпечення постійної готовності. Фахівці IBM установлюють стандартну конфігурацію по заздалегідь обраному шаблоні конфігурації, забезпечують безперервну роботу інфраструктури й застосунку, а також надають вилучену підтримку. Таке середовище забезпечує наступні переваги:

– **Висока рентабельність**: використання хмарного середовища дозволяє швидше приступитися до роботи, надати своїм розроблювачам всі необхідні інструменти для досягнення успіху й забезпечити відповідність вимогам корпоративних політик.

– **Оптимізація витрат на експлуатацію**: розгортання інструментів розробки без залучення IT-фахівців, скорочення витрат на інфраструктуру й вивільнення коштовних IT-ресурсів.

– **Швидка масштабованість**: можливість швидко наростити ресурси без зниження рівня готовності IT-інфраструктури або залучення досвідчених фахівців.

Послуги Customized Managed Services

Послуги Customized Managed Service – це додаткові платні послуги по налаштуванню, інтеграції з іншими інструментами розробки, міграції даних при переході від локального до хмарного рішення, а також контролю за дотриманням певних нормативних вимог і вимог до безпеки. Ви можете вибрати таку комбінацію ключових хмарних послуг, що відповідає потребам вашої організації.

Огляд Rational solution for Collaborative Lifecycle Management

The Rational solution for Collaborative Lifecycle Management (CLM) – це застосунок IBM® Rational DOORS Next Generation, IBM Rational Team Concert і IBM Rational Quality Manager, які поставляються разом з Jazz Team Server.

Кожний застосунок CLM складається з однієї або декількох груп функцій. Розходження між застосунками й групами функцій визначається з урахуванням рольового ліцензування. Застосунок являє собою базову одиницю установки, розгортання й відновлення; група функцій – це мінімальний набір функцій, яку можна активувати за допомогою ліцензії.

Jazz Team Server пропонує базові служби, такі як керування користувачами й ліцензіями, що забезпечують спільну роботу застосунків CLM у якості одного логічного сервера. У такій конфігурації Jazz Team Server виконує роль центра інтеграції для застосунків. Після установки продуктів CLM варто встановити ключі ліцензій в Jazz Team Server, щоб одержати доступ до функцій застосунку.

– **Rational DOORS Next Generation** використовує застосунок Керування вимогами (RM), що включає в себе групу функцій Керування вимогами. Ця група функцій надає кошти для збору, організації, спільної перевірки й аналізу вимог і створення звітів, особливо пов'язаних із завданнями розробки й тестових артефактів.

– **Rational Team Concert** використовує застосунок Керування змінами й конфігурацією (CCM), що включає в себе п'ять груп функцій:

– **Керування змінами.** Головним елементом керування змінами є завдання, які відслідковують і координують сюжети, дефекти, пункти плану й звичайні завдання. Завдання й процес потоку операцій, у який вони входять, можна настроїти залежно від вимог свого проекту. Керування змінами ССМ також можна інтегрувати з керуванням змінами в Rational ClearQuest.

– **Планування.** Група функцій Планування надає інструментарій для допомоги в плануванні, моніторингу й розподілі навантаження для всього проекту, для колективів, задіяних у цих проектах, і для окремих розроблювачів. Плани доступні всім учасникам колективу; вони відбивають стан виконання роботи з випусків і ітерацій у будь-який момент часу.

– **Керування конфігурацією програмного забезпечення.** Заснована на компонентах система контролю вихідного коду надає сильну підтримку для паралельної розробки, швидкої розробки й роботи географічно вилучених друг від друга колективів. Вона тісно інтегрується з функціями моніторингу дефектів, компонування й автоматизації процесів. Крім того, передбачена можливість інтеграції з іншими системами керування вихідним кодом, такими як Rational ClearCase.

– **Автоматизація.** Група функцій Автоматизація надає колективам розробки й тестування засобу керування компонуванням. Учасники колективу можуть відслідковувати хід виконання компонування, переглядати попередження й результати компонування, запитувати компонування й трасувати взаємозв'язку компонувань в артефакти, такі як набори змін і завдання.

– **IBM Enterprise Platforms Development.** Ця група функцій надає додаткові функції для більших підприємств, що використовують ПЗ System z і Power Systems Software. До цих функцій ставиться інтеграція з компонуваннями, системами керування вихідним кодом на рівні хостів, розміщення серверів на платформах System z і Power Systems, а також тісна інтеграція із продуктами Rational Developer for System z і Rational Developer for Power Systems Software. Вона включає клієнт ISPF; контекстний пошук завдань і вихідних кодів на Java™,

C, C++ і COBOL; а також розширені функції компонування, просування, структурування й розгортання.

– **Rational Quality Manager** складається з функцій, пропонованих застосунками RM, CCM і Керування якістю (QM). Група функцій Керування тестуванням, включена в застосунок QM, підтримує ряд ролей контролю якості, таких як керівники груп тестування, керівники лабораторій тестування й учасники проекту, що створюють і запускають тести. До числа компонентів ставляться динамічні плани тестування, керовані потоки операцій, ефективність праці, аналіз охвату тесту й створення тестів вручну. Ці компоненти інтегруються з іншими артефактами життєвого циклу, такими як завдання й вимоги, а також зі звітами й зведеними панелями. Вони надають докладні аналітичні дані із гнучкими можливостями налаштування, що допомагають відслідковувати стан і хід виконання проекту.

Звіти

Ще одна важлива функція – це можливість налаштування звітів. Звіти, що налаштовуються, надають як оперативні, так і хронологічні тенденції артефактів протягом усього життєвого циклу проекту, включаючи вимоги, завдання, компонування, тестові набори й результати тестів. Колективні звіти й зведені панелі допомагають відслідковувати стан проекту. Зведені панелі дають наочне подання про запити завдань, стрічках подій, звітах і інших найважливіших елементах, по яких можна оцінити хід проекту.

Керування глобальними конфігураціями

Керування глобальними конфігураціями (GCM) – це функція складання конфігурацій з застосунків CLM, RM, QM, DM і CCM у глобальні конфігурації, які відображаються в ієрархічній структурі. GCM використовує тільки області проектів, для яких у застосунках CLM було налаштоване керування конфігураціями. У таких областях проектів CLM можливо управляти артефактами, потоками, контрольними версіями, змінами артефактів і версіями посилань на артефакти між інструментами. GCM пропонує інструменти

складання, візуалізації, керування й аналізу власних конфігурацій і конфігурацій застосунків CLM.

Ліцензії й ролі

Ліцензії на основі ролей індивідуальних користувачів визначають доступність груп функцій для цих користувачів. Наприклад, власник ліцензії розроблювача Rational Team Concert має права доступу для створення й зміни завдань і планів, але до планів тестування (створеним співробітником з ліцензією на Rational Quality Manager) йому надані права доступу тільки для читання.

Rational solution for Collaborative Lifecycle Management (CLM) надає можливість інтеграції застосунків Керування змінами й конфігурацією, Керування вимогами й Керування якістю на платформі Jazz для об'єднання зусиль аналітиків із групами розробки й тестування. Rational solution for CLM дозволяє створювати посилання на артефакти для забезпечення відслідковуємості й веб-навігації з метою створення звітів і зведених панелей у застосунках CLM. Інтеграції CLM компонуються на Jazz Foundation з метою забезпечення загального підходу до зв'язків між артефактами, зведеним панелям, безпеці й середовищам користувальницьких інтерфейсів.

Огляд Rational DOORS Next Generation

– IBM® Rational DOORS Next Generation – це інструмент керування вимогами на основі веб-клієнта й платформи IBM Rational Jazz. Rational DOORS Next Generation оснащує колективи засобами визначення й керування вимогами, системою завдань для планування й керування завданнями, а також системою створення звітів.

Застосунок Керування вимогами (RM) призначено для створення вимог і керування ними в проекті розробки життєвого циклу, а також для створення звітів про вимоги. Функції, доступні в цьому застосунку, ліцензуються як IBM Rational DOORS Next Generation.

Власники ліцензій Rational DOORS мають право на одержання ключів ліцензій Jazz для Rational DOORS Next Generation. Rational DOORS Next Generation надає функції з різних застосунків, розгорнутих на Jazz Team Server:

– Застосунок Керування вимогами (RM) пропонує візуальні функції керування вимогами й визначеннями вимог.

– Застосунок Керування змінами й конфігурацією (CCM) пропонує функції керування й планування, а також підтримку зведених панелей.

– Застосунок Керування якістю (QM) надає функції тестування й керування тестами.

– Jazz Reporting Service пропонує функції створення звітів для відображення ключових індикаторів продуктивності (KPI) і стану проекту.

– Застосунок Керування проектуванням пропонує функції керування проектуванням.

Функції керування вимогами

Застосунок RM дозволяє визначати, витягати, збирати, перетворювати, обговорювати й перевіряти вимоги й допоміжні артефакти в співтоваристві зацікавлених осіб. Колективи можуть управляти вимогами за допомогою атрибутів, тегів, відфільтрованих подань, користувальницьких типів артефактів і типів зв'язків, індикаторів змін і зведених панелей проектів. У контексті Rational solution for Collaborative Lifecycle Management (CLM) застосунок RM розширює функції відслідковуємості й спільної роботи на артефакти розробки, проектування й тестування.

Керування конфігураціями

Керування конфігурацією можна використовувати в застосунку RM для створення версій артефактів вимог і для зв'язку їх з іншими артефактами колективу, наприклад тестовими наборами й ескізами. Конфігурації (потоки й контрольні версії) можна застосовувати для керування повторним використанням, відслідковуємістю й паралельною розробкою. Колективи, що застосовують застосунки CLM із включеним керуванням конфігураціями,

додавати вимоги, ескізи, тести й вихідні конфігурації в глобальні конфігурації. Глобальні конфігурації забезпечують визначення посилань артефактів на правильні версії, а також спрощують повторне використання у версіях і варіантах лінії програмного забезпечення або продукту.

Завдання керування вимогами

Розроблювачі вимог, бізнес-аналітики й колективи розробки програмного забезпечення можуть використовувати застосунок RM для виконання наступних завдань:

– Збір біографічних даних, потреб зацікавлених осіб, цілей проекту й іншої інформації в артефактах у вигляді відформатованого тексту.

– Створення користувальницьких типів артефактів, атрибутів, типів даних, типів зв'язків і шаблонів проектів.

– Добування артефактів вимог з документів.

– Опис поточних і запропонованих процесів, а також вказівка використання системи у вигляді діаграм бізнес-процесів і діаграм варіантів використання.

– Створення ескізів користувальницьких інтерфейсів за допомогою діаграм і макетів.

– Створення зв'язків елементів одного артефакту з іншими артефактами проекту або зв'язків із зовнішніми веб-сайтами.

– Створення й перегляд підозрілих посилань відслідковуємості для відстеження впливу змін на зв'язані артефакти

– У проектах з підтримкою керування конфігурацією можна відслідковувати вплив змін зв'язаних артефактів за допомогою допустимості посилань.

– Колективна перевірка артефактів і додавання коментарів.

– Створення контрольних версій проектів.

– Створення й обслуговування потоків і наборів змін у проектах, у яких дозволене керування конфігурацією.

– Створення наборів артефактів.

– Створення модулів артефактів.

– Включення наборів вимог і модулів із планами розробки й планами тестування.

– Створення вимог із завданнями й тестовими наборами.

– Створення відслідковуємості між вимогами, моделями й ресурсами проектування.

– Визначення глосаріїв і створення термінів, інтегрованих в артефакти проекту.

– Створення звітів.

– Створення вимог на основі тексту документів і інших елементів.

– Імпорт артефактів вимог з файлів CSV і ReqIF.

Застосунок RM і OSLC

Застосунок RM створений на основі платформи інтеграції Jazz. Архітектура програмного забезпечення застосунку RM в CLM побудована на основі специфікації Open Services for Lifecycle Collaboration (OSLC), що використовує загальний набір ресурсів, форматів і архітектурних служб REST для спільного використання даних між застосунками CLM. OSLC пропонує відкритий, масштабований набір специфікацій для інтеграції інструментів у різнорідних середовищах розробки. Спільне використання даних підтримує зв'язку на основі протоколу HTTP, ідентифікацію ресурсів за допомогою URI і добування інформації за допомогою типів умісту на основі промислових стандартів. Захист сервера взаємодії й загальних даних забезпечується за допомогою стандартних механізмів ідентифікації HTTP і протоколу авторизації OAuth.

За допомогою протоколу інтеграції OSLC можна зв'язати вимоги в Rational DOORS з артефактами в застосунку RM. Для обміну даними вимог між продуктами застосовується ReqIF (нова версія формату обміну вимогами, розроблювальна Object Management Group (OMG)). Використання формату ReqIF

для обміну даними дозволяє співробітникам різних організацій, що використовують різні інструменти для керування вимогами, працювати над загальними специфікаціями з метою узгодження даних.

Огляд Rational Team Concert

Rational Team Concert – це інструмент колективної роботи, створений на основі масштабованої й розширюваної платформи. Rational Team Concert використовує застосунок Change and Configuration Management (CCM) для надання функцій, що інтегрують завдань проекту розгортання, включаючи планування ітерацій, визначення процесів, керування змінами, відстеження дефектів, керування вихідним кодом, автоматизацію компонування й створення звітів.

Спільна робота й інтеграція на різних етапах розробки

Rational Team Concert спрощує прямий обмін інформацією в контексті виконуваної роботи. Всім учасникам групи автоматично відправляються повідомлення про всі зміни запитів на поліпшення. Зміни можна вказувати в сеансах розмов і зв'язувати з артефактами. Зацікавлені особи можуть автоматично відслідковувати стан окремого завдання.

Кілька подань дозволяють надавати загальний доступ до інформації колективу. Можна відслідковувати роботу колективу, змінювати рівень подробиці інформації й налаштовувати відображувану інформацію.

У продукт інтегровані багато завдань життєвого циклу розробки, включаючи планування Agile, визначення процесу, керування вихідними текстами, відстеження дефектів, керування компонуванням і створення звітів. Кожний із зазначених аспектів інтегрований в окремому середовищі. Користувач може відслідковувати взаємозв'язки між артефактами й управляти ними, просувати відомі процеси розробки й збирати проектну інформацію в автоматичному режимі.

Налаштування процесу

Проекти Rational Team Concert додержуються процесу. Процес – це набір ролей, методів, правил, прав доступу й рекомендацій, що забезпечують організацію й керування потоком операцій проекту. Rational Team Concert спрощує й підвищує ефективність виконання процесів колективом. Процес змінює поводження інструмента, збільшуючи надійність відповідності користувача процесу.

За допомогою процесу визначаються ролі користувачів і їхнього права доступу для виконання операцій у контексті проекту. Первісний процес, використовуваний у проекті, визначається шаблоном процесу. Шаблон можна змінити для обліку вимог проекту й колективу. Оскільки процес підтримують усе компоненти, можна додавати правила поведінки у вигляді попередніх умов і наступних дій.

Процес надає широкі можливості по налаштуванню. Він може бути більш-менш строгим залежно від потреб колективу. Процес можна настроїти для застосування різних правил на різних етапах процесу випуску. Наприклад, до кінця випуску може знадобитися більше строгий процес, щоб зменшити ризик появи нових регресій. Один зі способів це зробити – зажадати від користувачів одержувати твердження від керівників колективів і колег для внесення змін.

Керування змінами

Головний елемент керування змінами – завдання, які відслідковують і координують завдання, сюжети, дефекти, пункти плану й запити на поліпшення. Завдання й процес потоку операцій, якому вони випливають, можна налаштовувати під конкретний проект. Завдання також можна інтегрувати з іншими системами керування змінами, наприклад IBM® Rational ClearQuest.

Планування

Компонент планування надає кошти планування, відстеження й розподіли завдань випусків і ітерацій цілих проектів для колективів, що беруть участь у цих проектах, і окремих розроблювачів. Плани доступні для всіх учасників колективу

й можуть змінюватися в міру виконання ітерації, відбиваючи положення й напрямок діяльності колективу.

Керування конфігурацією програмного забезпечення (SCM)

Убудована система керування вихідним кодом є компонентної, підтримує паралельну розробку й розробку Agile, а також географічно розподілені колективи. Вона інтегрується із засобами відстеження дефектів, компонування й автоматизації, орієнтованої на процеси. Також доступна інтеграція з іншими системами керування вихідним кодом, такими як IBM Rational ClearCase і Git.

Керування конфігурацією

Починаючи з випуску 6.0, Rational solution for CLM підтримує керування версіями артефактів у застосунках Керування проектуванням (DM), Керування якістю (QM) і Керування вимогами (RM). У цих застосунках версії артефактів відслідковуються в конфігураціях. Крім того, до складу Rational solution for CLM входить застосунок Керування глобальними конфігураціями, що дозволяє збирати конфігурації з декількох застосунків. Такий підхід дозволяє відслідковувати версії артефактів і управляти ними у всьому життєвому циклі розробки. Крім застосунків DM, QM і RM, глобальна конфігурація може містити контрольну версію SCM Rational Team Concert.

Автоматизація компонування

Функція автоматизації забезпечує інформування, контроль і трасуємість компонування для колективів розробки й тестування. Учасники колективу можуть відслідковувати хід виконання компонування, переглядати попередження й результати компонування, запитувати компонування й трасувати компонування в інші артефакти, наприклад набори змін і завдання.

Звіти

Компонент Jazz Team Reports інформує про дії, поводження й виконання роботи в колективі або проекті. Візуалізація інформації про процес розробки може виявити певні тенденції, які в противному випадку залишаються

непоміченими або неясними. Роблячи цю інформацію наочної, звіти можуть підвищити ефективність прийняття рішень.

Rational Team Concert включає велику бібліотеку шаблонів звітів, які можна використовувати для створення звітів про стан проекту. Наприклад, звіт про завдання, що блокують, звіт про результати компонування, звіт про завдання по областях колективу, звіт про завдання по пріоритеті, звіт про операції проекту та інші.

Зведені панелі

Зведена панель являє собою компонент веб-клієнта, призначений для виводу зведеної інформації про стан проекту. Передбачено можливість одержання більше повної інформації. Зведені панелі також є засобом інтеграції інформації компонентів Jazz.

Зведені панелі можна використовувати різними способами:

– Керівники проекту можуть відслідковувати стан проекту й тенденції його розвитку.

– Колективи можуть обговорити стан, використовуючи зведену панель як джерело даних.

– Розроблювачі можуть створити персональні зведені панелі, що показують інформацію про завдання, які їм призначені.

Клієнт Eclipse, клієнт Microsoft Visual Studio і веб-інтерфейси

Rational Team Concert надає користувачам інтерфейс клієнта на основі Eclipse, інтерфейс клієнта Microsoft Visual Studio і веб-інтерфейс. Інтерфейси клієнта дають розроблювачам повнофункціональне інтегроване середовище розробки для створення й доставки артефактів. Веб-інтерфейс добре підходить для адміністрування серверів і проектів. З його допомогою користувачі можуть звертатися до областей проекту, переглядати інформацію в сховище, переглядати й змінювати плани, обновляти завдання й дізнаватися про останні події.

Rational Team Concert також містить у собі наступні клієнти:

– Інтеграція Rational Team Concert Shell, що дозволяє виконувати операції системи керування вихідним кодом із Провідника Windows.

– Rational Team Concert MSSCCI Provider, що дозволяє виконувати операції системи керування вихідним кодом із засобів, інтегрованих з MSSCCI, таких як MATLAB і Rational Rhapsody.

– Інтерфейс командного рядка системи керування вихідним кодом.

– Клієнт Rational Team Concert Interactive System Productivity Facility (ISPF), що надає інтерфейс ISPF для виконання різних функцій Rational Team Concert. Цей інтерфейс можна використовувати для зміни, повернення, доставки й компонування вихідного коду, що зберігається в сховище Rational Team Concert.

– Клієнт Rational Team Concert для Microsoft Visual Studio IDE не підтримує функції Enterprise Extensions.

Можливості розробки, специфічні для System z і Power Systems Software

– Для підтримки міжплатформеної розробки до складу Rational Team Concert входить Jazz Team Server on z/OS, Linux for System z і Power Systems Software for IBM i, і інтегрований набір засобів спільної доставки для розробки під System z і IBM i, включаючи засобу керування вихідним кодом, змінами, компонуванням і процесом. Багаторівнева розробка систем і програмного забезпечення, модернізація застосунків і артефакти традиційних мов, такі як COBOL, мають спеціальну підтримку. Rational Team Concert також забезпечує компонування артефактів System z і IBM i і підтримку файлової системи через функції Enterprise Extensions.

Огляд Rational Quality Manager

– IBM® Rational Quality Manager – це веб-інструмент для спільної роботи із всебічного планування тестування, створення тестів і керування артефактами тесту на всіх етапах циклу розробки.

Продукт Rational Quality Manager заснований на платформі Jazz і має багатьма її характеристиками. Продукт Rational Quality Manager призначений для груп тестування будь-якого масштабу й підтримує різні ролі, такі як адміністратор тестування, проектувальник тестів, керівник тестування, тестувальник і адміністратор лабораторії. Застосунок також підтримує ролі, що не ставляться прямо до тестування.

Rational Quality Manager – це застосунок Керування якістю (QM) у рішеннях Rational solution for Collaborative Lifecycle Management (CLM) і IBM Internet of Things Continuous Engineering (CE). Ці рішення поєднують продукти IBM Rational для надання повного набору застосунків для розробки систем і ПЗ.

Всебічне планування тестування

План тестування, створюваний в Rational Quality Manager, управляє всією діяльністю розподілених робочих груп на всіх етапах життєвого циклу проекту. План тестування визначає мети й область тестування й містить критерії, по яких колективи визначають готовність проекту до випуску.

План тестування можна настроїти під потреби конкретної організації. План тестування використовується для рішення наступних завдань:

– Визначення бізнес-цілей і цілей тестування.

– Створення процесу перевірки й твердження плану тестування й окремих тестових наборів.

– Керування вимогами проекту й тестових наборів і встановлення взаємозв'язків між ними.

– Оцінка трудомісткості тестування.

– Створення розкладу для кожної ітерації тестування й відстеження дат інших важливих заходів, пов'язаних з тестуванням.

– Ознайомитися з переліком середовищ, що підлягають тестуванню, і створити конфігурації тестування.

– Створення моментальної копії доступної тільки для читання в певний момент часу.

– Створення критеріїв якості, критеріїв входу й виходу.

– Створення й керування тестовими наборами.

– Перегляд стану виконання тесту.

Ескіз тесту з тестовими наборами

Ескіз тестового набору й функції побудови можна використовувати для визначення загального ескізу для кожного тестового набору. Кожний тестовий набір включає редактор відформатованого тексту, що дозволяє додати фонову інформацію про тестовий набір. Також тестовий набір містить посилання на вимоги й елементи розробки. Тестовий набір можна зв'язати з іншими артефактами тестів, такими як плани тестування, тестові сценарії й записи виконання тестових наборів. Крім цього, тестові набори можна поєднувати в комплекти тестів.

Створення й повторне використання тестових сценаріїв

До складу Rational Quality Manager входить повнофункціональний редактор неавтоматизованих тестів. Крім того, неавтоматизовані тести можна автоматизувати й повторно використовувати за допомогою ключових слів.

За допомогою Rational Quality Manager можна управляти й запускати тестові сценарії, створені такими інструментами, як IBM Rational Functional Tester, IBM Rational Performance Tester, Rational Service Tester for SOA Quality і IBM Security AppScanTester Edition.

Виконання тестів

Rational Quality Manager включає інтегроване середовище тестування для виконання тестів, розроблених у продукті, а також тестів, створених за допомогою інших інструментів неавтоматизованого, функціонального тестування, тестування функціональності й захисту. Доступні різні способи виконання тестів:

– Запуск тестового набору.

– Об'єднання тестових наборів у комплекти тестів для паралельного або послідовного виконання.

– Створення записів виконання тестових наборів і комплектів тестів, щоб прямо зв'язати інформацію про середовище тестування з тестовими наборами й комплектами тестів.

Аналіз тестів, звіти й динамічні подання

До складу Rational Quality Manager входить набір стандартних звітів BIRT, що допомагають оцінити стан проекту. Jazz Reporting Service пропонує стандартні звіти по декількох проектах для керування тестами, такі як Тестових наборів на вимогу й Виконань тестів на вимогу. Звіти Jazz Reporting Service можна використовувати як відправну крапку при розробці власних звітів.

Для перегляду поточного стану виконання тестів досить відкрити план тестування або подання виконання в списку планів тестування. Відносини між тестовими артефактами, вимогами, елементами архітектури й артефактами розробки можна відслідковувати в поданні трасуємості в списку тестових артефактів.

Якщо включено керування конфігураціями, то звіти Business Intelligence and Reporting Tools (BIRT), IBMCognos і інші звіти, що використовують сховище даних CLM як джерело даних, не будуть працювати й не будуть показані в меню Звіти. Для проектів з підтримкою конфігурацій можна використовувати засоби створення документів на основі шаблонів, убудовані в застосунки CLM, або IBM Rational Publishing Engine. Інші функції створення звітів для проектів з підтримкою конфігурації в цьому випуску доступні як пробні функції, які не призначені для застосування в робітничих середовищах.

Спільна робота

Rational Quality Manager спрощує спільне користування інформацією членами робочої групи. За допомогою системи завдань в Rational solution for Collaborative Lifecycle Management(CLM) учасники колективу можуть призначати завдання й дефекти один одному й бачити стан інших користувачів. Розроблювачі плану тестування й творці тестових наборів можуть розподіляти свою роботу для перевірки й відстеження стану кожного що перевіряє.

Учасникам колективу видимі нові й змінені вимоги. Учасники колективу також можуть переглядати тестові набори, необхідні для задоволення цих вимог. Члени групи можуть бачити, хто перебуває в системі, і над чим кожний з них працює. Учасники колективу можуть автоматично повідомлятися про зміни, вхідні дані й ітерації, що впливають на їхню роботу.

Більше того розроблювачі планів тестування, тестових наборів і тестових сценаріїв можуть блокувати свої артефакти, забороняючи іншим їх змінювати.

Керування лабораторією

За допомогою функцій керування лабораторією, якими володіє Rational Quality Manager, можна створювати запити середовищ тестування, зазначені в плані тестування. Після цього можна працювати з керівником лабораторії, щоб забезпечити доступність ресурсів лабораторії й середовищ тестування, коли це буде необхідно. Керівники лабораторії можуть стежити за всіма ресурсами лабораторії в централізованому сховищі ресурсів і запитами на обслуговування від колективів по тестуванню.

Захист веб-застосунків

Rational Quality Manager допомагає фахівцям з безпеки й IT створити захист від зовнішніх погроз і несанкціонованого доступу до даних за допомогою інтеграції з IBM Security AppScanTester Edition. Тестування захисту веб-застосунків може підвищити якість, зробити застосунку більше захищеними при прийнятних витратах.

Керування конфігурацією

В Rational Quality Manager функції керування конфігураціями можна використовувати для створення версій тестових артефактів і їхнього зв'язування з артефактами інших колективів, такими як вимоги й проекти. За допомогою конфігурацій (потоків і контрольних версій) з застосунків CLM можна управляти повторним використанням, трасуємістью й паралельною розробкою. Глобальні конфігурації дозволяють забезпечити правильну обробку посилань між артефактами з різних застосунків CLM. За допомогою застосунків CLM з

підтримкою керування конфігураціями колективи можуть додати вимоги, ескізи, тести й глобальні конфігурації в більше робітниче середовище. Глобальні конфігурації поєднують додані конфігурації в ієрархічну структуру. Глобальні конфігурації дозволяють планувати повторне використання конфігурацій і управляти ними в декількох версіях або варіантах програмного забезпечення, системи або лінійки продуктів.

При включенні керування конфігураціями в області проекту вимикається функція створення версій за замовчуванням (моментальні копії).

Керування

Rational Quality Manager допомагає забезпечити відповідність бізнес-процесів промисловим, корпоративним і галузевим стандартам і регламентам. На всіх етапах життєвого циклу тестування Rational Quality Manager надає інструменти для одержання оперативної оцінки якості програмного забезпечення й показників проекту. Завдяки всеосяжному плану тестування й інтеграції з інструментами керування й відстеження дефектів Rational Quality Manager допомагає раціоналізувати стратегію тестування й створити достовірні записи про результати тестування й хронологію проекту, які можна використовувати для контролю.

Частина співтовариства

Продукти CLM і CE розроблені на основі відкритій і розширюваній платформі Jazz. На веб-сайті Jazz.net можна завантажувати продукти і їхні стадії, відслідковувати плани розробки, приєднуватися до форумів, відкривати запити на поліпшення й взаємодіяти з розроблювачами продуктів.

Огляд компонента Керування проектуванням

Керування проектуванням це веб-інструмент спільної роботи, що дозволяє широкому колу зацікавлених осіб брати діючу участь у проектуванні продуктів, програмного забезпечення й систем. За допомогою продуктів керування проектуванням можна інтегрувати проектування в у життєві цикли розробки застосунків і систем у цілому й спільно працювати над моделями й ескізами.

Керування проектуванням підтримує підхід спільної роботи до проектування й створення архітектури програмного забезпечення й систем. Інтегруючи Керування проектуванням в Rational solution for Collaborative Lifecycle Management, можна зв'язати всі типи ескізів програмного забезпечення й систем з іншими ресурсами життєвого циклу, такими як вимоги, артефакти тестування й запити на зміни. Архітектори програмного забезпечення, технічні фахівці, розроблювачі й фахівці із планування можуть через Інтернет працювати разом з колегами, експертами по конкретному питанню, клієнтами й іншими зацікавленими особами.

Групи функцій керування проектуванням в Rational Rhapsody Design Manager підтримують інтеграцію з застосунками Jazz по керуванню змінами й конфігурації, керуванню вимогами й керуванню якістю.

Спільна робота в рамках контексту

Можна спільно використовувати моделі, створені в Rational Rhapsody, змінювати моделі через веб-клієнт і проводити перевірки ескізів у режимі онлайн разом з колегами й зацікавленими особами.

Відслідковуємість і аналіз впливу

При інтеграції компонента Керування проектуванням з CLM можна встановити зв'язку між елементами ескізу, іншими ескізами й артефактами життєвого циклу (керування архітектурою, керування вимогами, керування змінами й керування якістю). Двонаправлена відслідковуємість між вимогами й ескізами дозволяє швидше виявити прояви й зрозуміти причини впливу змін вимог або ескізів. Також можна створювати й переглядати діаграми аналізу впливу, на яких проглядаються зв'язки на вході й виході певних цільових елементів ескізу

Швидке створення ескізу

Керування проектуванням дозволяє використовувати функцію швидкого створення ескізу спільними зусиллями для втілення ідей проектування й створення архітектури. Потім можна спільно використовувати ці ескізи з

колегами й зацікавленими особами, виконувати перевірки ідей і вносити в ескізи виправлення ще до формального моделювання й проектування.

Створення звітів і документів

Можна використовувати убудовані функції створення документів, а також шаблони документів, що дозволяють створювати й спільно використовувати документи проектування, специфікації й звіти. Створюючи звіти за допомогою веб-клієнта Керування проектуванням, учасники проекту й зацікавлені особи можуть легко звертатися до інформації моделей і проектів.

Керування змінами для ескізів

Можна зберігати різні типи архітектур, ескізів і моделей і управляти ними в застосунках CLM.

Звичні інструменти проектування зберігають ескізи у файлах, керованих системами керування вихідним кодом (SCM). У компоненті Керування проектуванням ескізи й моделі обробляються як артефакти першого класу, що означає можливість імпорту моделей на сервер і безпосереднє керування ними на рівні моделі; не потрібно перетворювати елементи моделей у файли. Цей підхід спрощує потік операцій, що дозволяє успішно управляти архітектурами й ескізами в контексті колективу.

Керування конфігураціями

Функції керування конфігураціями в застосунку Керування проектуванням дозволяють створювати версії артефактів проектування й зв'язувати їх з іншими артефактами колективу, такими як вимоги й тестових наборів. За допомогою конфігурацій (потоків і контрольних версій) з застосунків CLM можна управляти повторним використанням, відслідковуємістю й паралельною розробкою. Зберіть конфігурації в глобальні конфігурації, щоб правильним образом обробляти версії артефактів у різних застосунках CLM. Колективи можуть використовувати застосунки CLM з підтримкою керування конфігураціями для додавання вимог, проектів, тестів і глобальних конфігурацій у більше масштабне робітниче середовище. Глобальні конфігурації збирають

складові конфігурації в ієрархічне дерево. Глобальні конфігурації дозволяють планувати повторне використання конфігурацій і управляти ними в декількох версіях або варіантах програмного забезпечення, системи або лінійки продуктів.

Загальне адміністрування користувачів і адміністрування проекту життєвого циклу

Можна встановити й настроїти компонент Керування проектуванням на спільне використання Jazz Team Server з застосунками CLM, для того щоб була потрібна тільки однокористувальницька база даних, а проекти ескізу були частиною проекту життєвого циклу.

Установка компонента Керування проектуванням у середовище CLM із загальним Jazz Team Server дають перевагу від використання наступних спільно використовуваних (загальних) функцій керування життєвим циклом:

– Загальне адміністрування користувачів.

– Проекти життєвого циклу, що включають ескізи.

– Приклад застосунку Гроші мають значення, що включає артефакти ескізів.

– Загальні конфігурації розгортання.

– Загальна підтримка для платформи.

Моделювання для домену

Керування проектуванням підтримує загальні домени, такі як UML або BPMN, а також домени користувальницького моделювання. Це означає наступне: якщо ваша організація працює з певним доменом, ви можете спільно використовувати моделі й ескізи даного домену, працюючи над ними разом з іншими користувачами.

Інтеграція із клієнтами

Rational Rhapsody можна інтегрувати з Design Management Server. Ця інтеграція дозволяє звертатися до моделей і ескізів на Design Management Server, шукати й запитувати ескізи, аналізувати й перевіряти ескізи й створювати звіти.

Введення в Internet of Things Continuous Engineering Solution

IBM® Internet of Things Continuous Engineering Solution (IoT) – це набір застосунків IBM, що забезпечують колективам можливість керування складністю розробки розумних з'єднаних продуктів шляхом застосування принципів безперервної розробки. Рішення працює в хмарному середовищі або на серверах підприємства й містить тісно інтегровані інструменти, що охоплюють життєвий цикл розробки програмного забезпечення й систем, включаючи керування вимогами, моделювання й імітацію, керування якістю, керування конфігураціями, а також спільне планування потоків операцій і керування цими потоками.

Безперервна розробка містить у собі методики, спрямовані на досягнення більше гнучких і стабільних поліпшень процесу розробки в нове століття Інтернету речей:

Аналіз розробки

Інтегровані інструменти й проста інтеграція даних дозволяють скоротити обсяги оброблюваних даних і прискорити прийняття оптимальних рішень, що забезпечують постійне просування в правильному напрямку.

Постійна перевірка

Переміщення традиційної перевірки, виконуваної наприкінці циклу, на більше ранні етапи й регулярні перевірки якості компонування із самого початку.

Стратегічне повторне використання

Інтелектуальне повторне використання артефактів розробки; ефективне керування конфігураціями у всіх інструментах розробки. Реалізація розробки лінійки продуктів для максимального підвищення потенціалу повторного використання.

Основні компоненти рішення

До складу рішення входять наступні функції й продукти:

– Визначення вимог і керування ними за допомогою Rational DOORS або Rational DOORS Next Generation

– Створення, проектування, опис і трасування вимог. Дозволяє зацікавленим сторонам визначити те, що їм потрібно, а також намітити мети, і допомагає колективам ефективно надавати кінцевий продукт із урахуванням неминучих змін.

Моделювання й імітація за допомогою Rational Rhapsody Design Manager або Rhapsody Model Manager

Засоби візуального моделювання дозволяють перевірити вимоги, описати архітектури й створити убудоване програмне забезпечення, що працює в режимі реального часу. Rhapsody Model Manager – це новітня технологія керування проектуванням – дозволяє колективам планувати, проектувати й розробляти системи й програмне забезпечення із застосуванням ітераційного колективного підходу.

Масштабоване гнучке планування й координування на рівнях колективу, проекту, програми й портфеля за допомогою Rational Team Concert

Інтеграція планування й виконання, автоматизація потоків операцій і керування змінами для різних технічних дисциплін і колективів розроблювачів.

Перевірка, тестування й керування якістю з помощьюRational Quality Manager

Спільне планування якості, автоматизоване тестування й керування дефектами для забезпечення якості, закладеного в проекті.

Керування складними даними проектування за допомогою різних інструментів в Rational Engineering Lifecycle Manager

Розкрийте об'єднаний потенціал методики розробки з різних застосунків для ефективного аналізу дані керування життєвим циклом продуктів.

Огляд глобальних конфігурацій

Застосунок Керування глобальними конфігураціями (GCM) поєднує свої конфігурації й конфігурації з інших застосунків-учасників Rational solution for Collaborative Lifecycle Management, щоб працюючий над продуктом колектив

одержав загальне подання про логічну структуру, або структурі конфігурацій, продукту. Конфігурацією називаються контрольна версія або потік, що містять набір артефактів з версіями. Глобальна конфігурація представляє логічний фрагмент продукту. Спільне використання локальних і глобальних конфігурацій дозволяє управляти багаторазовим використанням артефактів і відслідковуємістю артефактів, а також забезпечити паралельну розробку.

Застосунок Керування глобальними конфігураціями можна інтегрувати з наступними застосунками CLM:

– Застосунок Керування вимогами (RM) надає функції керування вимогами й визначеннями вимог.

– Застосунок Керування якістю (QM) надає функції тестування й керування тестами.

– Компонент керування вихідним кодом (SCM) застосунку Керування змінами й конфігураціями (CCM) надає кошти для роботи із завданнями й артефактами розробки.

– Застосунок Керування архітектурою (AM) надає можливості моделювання, роботи з наборами змін і керування вимогами.

– Застосунок Керування проектуванням (DM) надає функції керування проектуванням.

– Застосунки RM і QM: адміністратор застосунків повинен активувати керування конфігураціями в застосунку, а потім включити відповідну можливість у кожній області проекту, що буде доповнювати глобальні конфігурації.

– Застосунок AM: активувати функцію в застосунку не потрібно, однак адміністратор повинен включити повнофункціональне керування конфігураціями в кожній області проекту, що буде доповнювати глобальні конфігурації.

– За рахунок об'єднання потоків і контрольних версій з різних застосунків CLM, глобальні конфігурації дозволяють одержати повну картину логічної структури продукту або рішення, що дає наступні можливості:

– Планувати повторне використання конфігурацій і управляти ними в декількох версіях або варіантах лінійки фізичних або логічних продуктів.

– Паралельно працювати в декількох потоках розробки.

– Відновити старе середовище розробки, наприклад для розгалуження й створення потоку для виправлення продукту.

Учасники колективу, що працюють із застосунками, що доповнюють, CLM, можуть указати контекст глобальної конфігурації у звичних інструментах. Глобальна конфігурація пропонує контекст для різних застосунків, забезпечуючи зв'язування версій артефактів із правильними потоками й контрольними версіями в кожному застосунку.

Якщо колективи інженерів мають можливість працювати з декількома потоками розробки з ефективним повторним використанням у всій пропозиції продуктів, то керівники проектів і їхні колективи можуть краще зрозуміти створювані й повторно використовувані конфігурації. Такий підхід дозволяє ефективно й оперативно приймати рішення, досягаючи при цьому мети, пов'язані з рівнем якості.

Адміністратори конфігурацій можуть виконувати наступні завдання, пов'язані із глобальними конфігураціями. Ці завдання можуть виконувати й інші учасники колективу, у тому числі керівники проекту, керівники розробки й керівники колективу, якщо адміністратор призначить їм відповідні права доступу:

– Налаштування глобальних конфігурацій з метою створення робочого контексту між застосунками для пропозиції продукту.

– Створення, зміна й видалення атрибутів, типів даних і типів посилань для користувачів, що додаються в глобальні конфігурації.

– Додавання конфігурацій для застосунку Керування конфігураціями й додатковими застосунками CLM у потік, а також керування цими конфігураціями.

– Створення глобальної контрольної версії для запису віхи.

– Створення варіантів конфігурацій за допомогою галузей.

– Порівняння конфігурацій.

– Запуск звіту, щоб виявити кілька конфігурацій компонента.

– Визначення конфігурацій, у яких використовується потік або контрольна версія.

– Перегляд, переміщення, зміна порядку, видалення й спільне використання локальних конфігурацій у дереві глобальної конфігурації. Конфігурації відображаються як один логічний блок.

Учасники колективу можуть виконувати наступні завдання, пов'язані із глобальними конфігураціями:

– У застосунку CLM можна вказати глобальну конфігурацію як робочий контекст для створення й перегляду посилань на артефакти в інших застосунках CLM.

– Перегляд локальних конфігурацій у дереві глобальної конфігурації. Конфігурації відображаються як один логічний блок.

– Додавання й видалення користувальницьких атрибутів, тегів і посилань, а також атрибутів, тегів і посилань галузей для глобальних конфігурацій.

Учасники колективу можуть виконувати й ряд інших завдань, у тому числі створювати потоки й контрольні версії, якщо адміністратор призначить їм відповідні права доступу.

Партнери по інтеграції

Застосунок GCM інтегрує й збирає воєдино конфігурації, що містять версії артефактів, створені в застосунках Rational DOORS Next Generation, Rational Quality Manager, Rational Team Concert, Rhapsody Model Manager і Rational Rhapsody Design Manager.

Служби інтеграції Rational Jazz

Застосунок Керування глобальними конфігураціями створено на основі платформи інтеграції Jazz. Архітектура програмного забезпечення застосунку Керування глобальними конфігураціями побудована на основі специфікації Open

Services for Lifecycle Collaboration (OSLC). OSLC використовує загальний набір ресурсів, форматів і архітектурних служб REST для спільного використання даних між застосунками CLM. Специфікація керування конфігураціями Organization for the Advancement of Structured Information Standards (OASIS) OSLC описує спосіб опису артефактів з підтримкою версій і створення посилань на них. Застосунки повинні відповідати цій специфікації для роботи із глобальною конфігурацією й версіями артефактів. Застосунку CLM відповідають специфікації керування конфігураціями OASIS OSLC.

OSLC надає відкритий розширюваний набір специфікацій для інтеграції інструментів у середовища розробки, що містять застосунки, створені самотужки, або інструменти інших постачальників. Обмін даними підтримує наступні транзакції:

– Підключення за протоколом HTTP.
– Ідентифікація ресурсів по URI.
– Пошук інформації з використанням стандартних типів носіїв.

Захист сервера взаємодії й загальних даних забезпечується за допомогою стандартних механізмів ідентифікації HTTP і протоколу авторизації OAuth.

У застосунку Rational Global Configuration Management (Керування глобальними конфігураціями) оновлені деякі функції й додані нові, що спрощують використання й підвищують рівень автоматизації.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки вебзастосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за

простоту у вивченні, ефективність та мультиплатформність. Python – скриптова мова програмування з досить простим синтаксисом. Для розуміння достатньо порівняти принципи написання найпростішої програми, яка виводить на екран текстове повідомлення. Саме тому мова програмування Python більш доступна для новачків, а професіонали встигли адаптувати її для вирішення великої кількості завдань. Це мультиплатформене рішення, тому знання Python дає можливість працювати у різних сферах: від розробки мобільних застосунків до ігрової індустрії та штучного інтелекту.

У мови програмування динамічна типізація: є можливість передавати до функцій будь-який тип даних без попереднього вказання. Інтерпретованість дозволяє знаходити помилки у коді ще до повної збірки у робочий застосунок. При цьому Python дуже чітко дає зрозуміти, де та через що виникла помилка.

Це мова об'єктно-орієнтованого програмування (ООП). Програмне забезпечення на Python оформлене у вигляді моделей, які можуть бути зібраними у пакети. Тип та структуру кожного об’єкта можна запитати під час виконання програми. Для кожного з об’єктів можна отримати всю інформацію щодо його внутрішньої структури. Окрім того:

– у мови логічний синтаксис, завдяки чому вихідний код легко читати та розуміти;

– гнучкість та масштабованість Python дозволяє адаптувати високорівневу логіку та розширяти складні застосунки, як тільки виникне така необхідність;

– розробка на Python у більшості випадків проходить швидше, ніж на інших мовах програмування;

– Python – інтерпретована мова програмування. Це значить, що код можна написати у будь-якому текстовому файлі на будь-якій платформі, і потім успішно запустити;

– у Python – колосальна спільнота однодумців. Тож будь-які складнощі конкретних розробників вирішуються колективно.

Проте є декілька особливостей, які можна віднести до недоліків. Це повільність (ця мова програмування хоч і універсальна, проте повільніша за інші), велика кількість ресурсів, необхідних для роботи та «прив'язаність» до системних бібліотек.

Мова програмування Python використовується у наступних сферах:

1. Розробка програмних застосунків будь-якого напрямку.

2. Розробка серверної частини мобільних застосунків (найпопулярніший напрямок).

3. Ігри. Багато сучасних ігор для комп'ютерів (наприклад, World of Tanks) частково чи повністю написані на Python.

4. Вбудовані системи для різних пристроїв. Дуже часто Python використовують для написання внутрішніх платформ управління банкоматами.

5. Скрипти та плагіни до уже реалізованих програм для автоматизації процесів чи створення інших рішень.

6. Тестування (автоматизація цього процесу).

7. Машинне навчання. – основна мова для написання алгоритмів і аналітичних застосунків у сфері Machine Learning.

Бібліотеки Python

Різні бібліотеки Python використовують для виконання конкретних завдань. Наприклад, Matplotlib підходить для відображення даних у двовимірній та тривимірній графіці. Pandas підходить для зручної роботи з даними. NumPy дозволяє створювати масиви та керувати ними. Requests використовується для веброзробки. OpenCV-Python відкриває можливості для обробки зображень з метою оптимізації систем «машинного зору».

Найвідоміші фреймворки для мови програмування Python

Фреймворки Python допомагають створити зручне та функціональне середовище для розробки. У них міститься набір інструментів, модулів та бібліотек, корисних для виконання конкретних завдань. Це значно полегшує роботу: наприклад, дає змогу не витрачати час на розписування дій, які

повторюються, а використати релевантний інструмент. Тож є можливість позбутися рутинних процесів та сконцентруватися на логіці проєкту.

Серед найпопулярніших фреймворків для Python:

– Django – найстаріший та найвідоміший. Створений для реалізації великих інтерактивних проєктів;

– Pyramid – зручний у налаштуваннях, і дає можливість реалізувати складні нестандартні ідеї;

– Web2py – підходить в першу чергу для вебзастосунків і може використовуватись на будь-яких архітектурах.

Популярні Python IDE

IDE або інтегровані середовища розробки – це програмне забезпечення, яке надає розробникам необхідні інструменти для написання, редагування, тестування та налаштування коду. Для розробки на Python найчастіше використовують IDE PyCharm, IDLE, Spyder та Atom.

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 IoT у дата-центрах.

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

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

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

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

програми та підпрограми;

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

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Керування периферійною інфраструктурою й пристроями Інтернету речей за допомогою системи IoT у дата-центрах

Система IoT у дата-центрах – це безпечне рішення корпоративного класу з керування пристроями Інтернету речей і периферійною інфраструктурою для IT-відділів і відділів експлуатації. Воно підходить для будь-яких сценаріїв використання Інтернету речей. За допомогою системи IoT у дата-центрах можна з легкістю впровадити технології Інтернету речей, автоматизувати керування з можливістю масштабування, забезпечити відповідність стандартам IT-безпеки для периферійної інфраструктури й інфраструктури Інтернету речей, а також підвищити цінність даних Інтернету речей.

Огляд продукту системи IoT у дата-центрах

Система IoT у дата-центрах забезпечує додавання, налаштування, адміністрування, моніторинг і захист різнорідних систем на периметрі Інтернету речей, а також підключених пристроїв з можливістю масштабування.

Переваги керування життєвим циклом пристроїв Інтернету речей за допомогою рішення системи IoT у дата-центрах

Широкі можливості керування

Оптимізація керування з можливістю масштабування за допомогою платформи погодженого керування й моніторингу для різнорідних пристроїв Інтернету речей і застосунків, яку можна масштабувати до декількох мільйонів пристроїв.

Поліпшений захист

Забезпечення відповідності стандартам IT-безпеки для периферійної інфраструктури й інфраструктури Інтернету речей за допомогою засобів точної візуалізації й контролю всіх підключених пристроїв, застосунків і мереж.

Швидке масштабування

Спрощене впровадження й масштабування технологій Інтернету речей завдяки стандартизованим процесам реєстрації, підключення й додавання пристроїв. Крім того, Системи IoT у дата-центрах автоматизує збір показників, що допомагає оптимізувати процеси.

Ефективна експлуатація

Використання технології Інтернету речей у повсякденній діяльності організації завдяки оптимізації збору даних і їх оркестрації на будь-яких пристроях, а також у будь-яких застосунках і хмарах.

Можливості системи IoT у дата-центрах

Зручне додавання пристроїв і застосунків

Нескладна реєстрація, пакетне додавання й шаблони пристроїв допомагають знизити витрати часу й вимоги до технічних навичок для більше швидкого додавання, ініціалізації й налаштування різних пристроїв і застосунків Інтернету речей з можливістю масштабування.

Погоджене керування

Уніфікована консоль і єдине подання різнорідних пристроїв, застосунків і мікропрограм Інтернету речей і периферійної інфраструктури з можливістю масштабування допомагають усунути розрізненість засобів керування Інтернетом речей.

Саморегулююча оркестрація даних

Організації можуть ефективно використовувати дані Інтернету речей, спростивши їхнє переміщення й інтегрувавши гнучкі потоки даних у бізнес-процеси.

Масштабованість на всіх рівнях: від демонстраційних установок до виробничих середовищ

Швидке розгортання пілотних проектів завдяки моделі «ПЗ як послуга» (SaaS) і перехід до локальних середовищ із урахуванням мінливих потреб бізнесу.

Інтеграція з корпоративним середовищем

Повна інтеграція з існуючими серверними засобами моніторингу й оповіщень за допомогою API-інтерфейсів REST, а також гнучка інтеграція на стороні клієнта за допомогою комплекту SDK на базі C++.

Модель комплексної довіри

Забезпечення відповідності стандартам IT-безпеки для периферійної інфраструктури й інфраструктури Інтернету речей за допомогою погодженої платформи безпеки. Ця платформа надає такі можливості, як виконання застосунків в агенті на основі принципу мінімальних привілеїв, призначення унікальних ідентифікаторів пристроям, безпечний обмін даними по протоколі TLS, контроль доступу на основі ролей і призначення користувальницьких ролей.

Керування безпекою пристроїв

Швидке налаштування й безпечне надання відновлень і виправлень для мікропрограм і ПЗ з деталізованим контролем їхнього твердження, планування, активації й установки.

Виявлення й ескалація погроз

Виявлення аномалій завдяки моніторингу граничних значень показників, що налаштовуються, пристроїв; ескалація й запуск дій по усуненню проблем за допомогою оповіщень на основі правил і інтеграції із системами сторонніх постачальників на основі API-інтерфейсів.

Автентифікація й авторизація

Для взаємодії із сервером і захисту від спуфінгу кожний шлюз використовує авторизовані списки керування доступом. Для автентифікації пристроїв при взаємодії із сервером використовується ключ доступу HMAC.

Безпека мережі

Системи IoT у дата-центрах забезпечує відстеження й виявлення погроз із використанням рішень сторонніх постачальників. У майбутніх версіях буде реалізована підтримка мікросегментації завдяки інтеграції з NSX.

Додаткові можливості

Як системи IoT у дата-центрах спрощує керування інфраструктурою Інтернету речей?

Відстеження й адміністрування об'єктів

До мереж підключається усе більше інтелектуальних пристроїв і шлюзів, і організаціям необхідні можливості по відстеженню їхнього місця розташування й забезпеченню їхньої безпеки, а також засобу для зручного адміністрування, ініціалізації й налаштування цих пристроїв і шлюзів. За допомогою системи IoT у дата-центрах можна забезпечити моніторинг граничних значень показників, що налаштовуються, пристроїв, а також запускати автоматизовані дії по усуненню проблем за допомогою оповіщень на основі правил і інтеграції із системами сторонніх постачальників на основі API-інтерфейсів.

Моніторинг працездатності інфраструктури Інтернету речей

Більшість пристроїв Інтернету речей, наприклад датчики в нафтових свердловинах, на стелях, у верстатах, на будівельних кранах або в реактивних двигунах, розташовані в областях без доступу персоналу. Система IoT у дата-центрах підтримує моніторинг і оповіщення в режимі реального часу для безперервного відстеження працездатності використовуваної інфраструктури Інтернету речей.

Засоби аналізу експлуатації

Візуалізація продуктивності інфраструктури Інтернету речей допомагає виявляти аномалії шляхом добування даних про експлуатацію (наприклад, відомостей про використання процесора й рівні заряду батареї) і зіставлення їх з ретроспективними даними, а також з інформацією від інших пристроїв.

Відновлення програмного забезпечення й системи безпеки по мережі

Життєвий цикл ПЗ на всіх пристроях Інтернету речей варто постійно підтримувати, що особливо важливо з погляду безпеки. Система IoT у дата-центрах надає єдину консоль для гнучкої візуалізації й точного контролю всіх підключених пристроїв з можливістю масштабування. Завдяки цьому можна візуалізувати і переглядати стан пристроїв, реагувати на погрози безпеки, установлювати виправлення системи безпеки й управляти ними, а також обновляти мікропрограми по мережі.

Збір і оркестрація даних

У периферійних пристроях і в пристроях Інтернету речей створюються безперервні потоки даних, і користувачам необхідні можливості їхнього адміністрування й добування з них практичної користі. Система IoT у дата-центрах допомагає підвищити цінність даних завдяки збору потоків даних і їх оркестрації на будь-яких пристроях, у будь-яких застосунках і хмарах.

Інтерактивний огляд Інтернету речей

Крім застосування Інтернету речей у дата-центрах, існує ще велика область застосування Інтернету речей. Наведемо огляд сценаріїв використання Інтернету речей, щоб довідатися, як Інтернет речей допомагає вирішувати галузеві завдання.

Інтелектуальне виробництво

Виробники, що використовують інтелектуальні процеси, стають технологічними компаніями. Вони використовують результати аналізу даних для оптимізації процесів, подальшої автоматизації виробництва й надання замовникам більше персоналізованих продуктів і нових послуг.

Мережні технології в транспортній сфері

Інтернет речей перетворить транспортну галузь: від проектування й виробництва транспортних засобів до логістики, страхування й роздрібних продажів. Об'єднання транспортних засобів, споживачів і транспортної

інфраструктури в єдину мережу відкриває величезні можливості для підвищення зручності роботи й безпеки водіїв, а також скорочення витрат.

Мережні технології в роздрібній торгівлі

Інтернет речей допомагає одержати конкурентну перевагу за рахунок швидкого реагування на мінливі потреби замовників, підвищення ефективності ланцюжка поставок і виявлення нових алгоритмів поводження замовників, а також сприяє збільшенню обсягу продажів і поліпшенню репутації бренда завдяки оптимізації обслуговування замовників.

Інтелектуальні банківські послуги

Інтернет речей забезпечує високий рівень зв'язку, завдяки якому виявляються недоліки в робочих процесах, підвищується якість банківських послуг і значно поліпшуються можливості персоналізації продуктів і послуг для клієнтів.

Мережні технології в охороні здоров'я

До 2027 р. обсяг ринку Інтернету речей у секторі глобальної економіки, зв'язаному з охороною здоров'я і якістю життя, складе 1,6 трильйона доларів США. Інтернет речей надає всім – пацієнтам, медичним установам, страховим компаніям – візуалізацію в режимі реального часу, що підвищує зручність і ефективність роботи, скорочує витрати й підвищує якість обслуговування пацієнтів.

Інтелектуальні міста

Інтернет речей в інтелектуальних містах забезпечує більше висока якість життя для всіх жителів завдяки скороченню витрат і споживання ресурсів, а також підвищенню рівня безпеки. Пристрої Інтернету речей, наприклад системи керування дорожнім рухом у режимі реального часу й інтелектуальні системи вуличного висвітлення, допоможуть містам по усьому світі заощадити 4,6 трильйони доларів.

Мережні технології в енергетику й нафтогазовій галузі

Використання Інтернету речей у сполученні з вилученим керуванням активами, інтелектуальним розрахунком споживання й інтелектуальних енергосистем сприяє більше ефективній експлуатації устаткування, у результаті чого скорочуються витрати, зберігаються природні ресурси й підвищується рівень безпеки співробітників.

Відкритий вихідний код

Photon OS

Photon Operating System (OS) забезпечує безпечне середовище виконання для ефективного використання контейнерів. Photon OS – це дистрибутив Linux з відкритим вихідним кодом, оптимізований для хмарних застосунків, платформ і інфраструктури VMware. Сполучення Photon OS і системи IoT у дата-центрах підтримує виконання сучасних хмарних і контейнерних застосунків, у тому числі засобів аналізу й фільтрації даних, розміщених близько до джерела даних на периметрі.

Edge

Edge Foundry – це гнучка платформа мікрослужб із відкритим вихідним кодом для створення самоналаштувальних рішень для середовищ периметра Інтернету речей. В основі проекту лежить платформа взаємодії, розміщена на повністю незалежній від устаткування й ОС програмній платформі, що надає екосистему самоналаштувальних компонентів, уніфікує доступ до них і прискорює розгортання рішень Інтернету речей.

Liota

Liota – це рішення на стороні клієнта для системи IoT у дата-центрах, що розгортається на периферійних шлюзах і елементах Інтернету речей і оптимізує передачу даних від цих елементів до системи IoT у дата-центрах. Агент Little IoT (Liota) – це комплект SDK без прив'язки до постачальника й з відкритим вихідним кодом, призначений для створення застосунків для Інтернету речей, що здійснюють моніторинг і адміністрування даних на всьому шляху від пристрою до шлюзу, хмарі й ЦОД.

3.2 Розробка структурної схеми

Серед забезпечуваних можливостей, властивим сучасним засобам керування інфраструктурою системи IoT у дата-центрах (DCIM), – підвищення операційної ефективності на основі моніторингу споживання електроенергії й роботи системи охолодження; збір даних у реальному часі, виявлення тенденцій і формування відповідних звітів; забезпечення «видимості» процесів у дата-центрах (ЦОД) і їхня візуалізація з використанням тривимірної графіки.

Дані надаються орендарям послуг, що дозволяє їм здійснювати безперервне спостереження за станом своєї інфраструктури, її енергоспоживанням, обробкою запитів на обслуговування. Для цього використовуються портал і інтерфейси прикладного програмування.

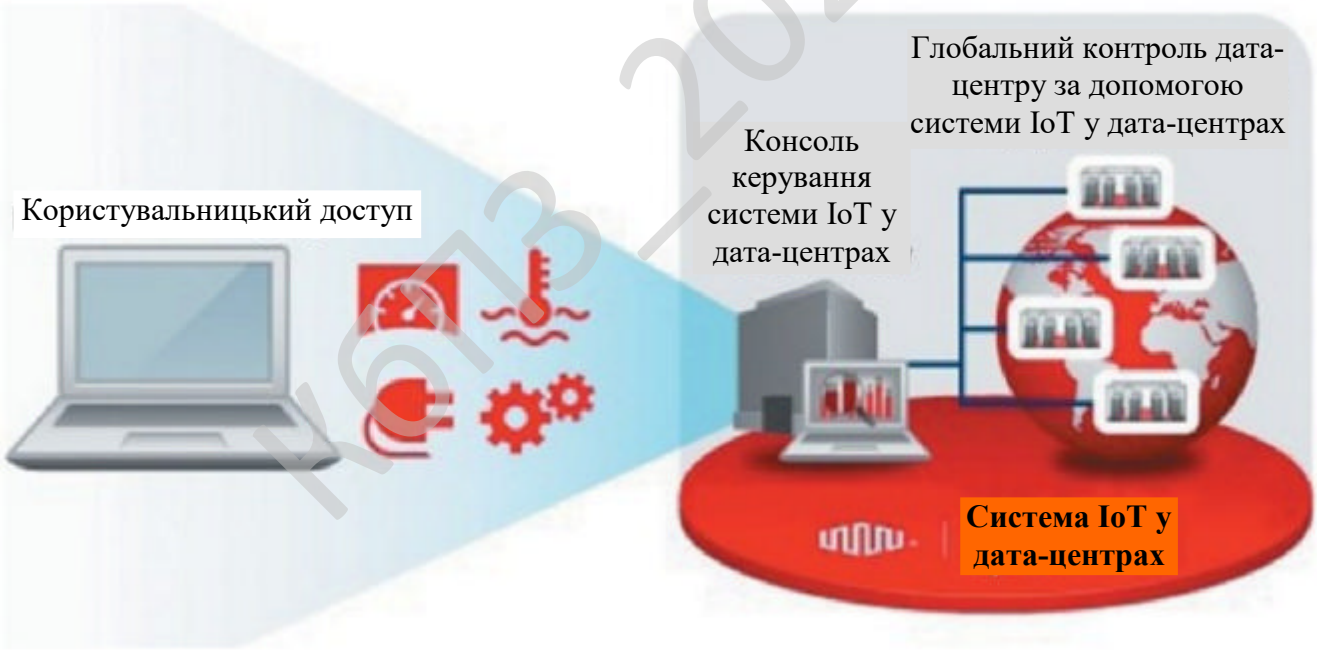

Рисунок 3.1 – Структурна схема системи

Через єдиний портал клієнти компанії в реальному часі одержують інформацію про встаткування, кліматичну ситуацію й аварійні сигнали у дата-центрі.

DCIM збирає в режимі реального часу відомості про основні параметри роботи ЦОД, інформує про тенденції їхніх змін і формує попереджуючі оповіщення.

Крім цього, кожному клієнтові надаються дані про стан його фізичної інфраструктури, енергоспоживанні, температурі в приміщеннях і про інших не менш важливих для орендарів параметрах.

Сервіси системи IoT у дата-центрах IT дозволяють використовувати режим, що аналітики назвали «керування центром обробки даних як сервіс» (Data Center Management as a Service, DMaaS).

Використовуючи по підписці набір сервісів DMaaS, клієнти в реальному часі одержують через єдиний портал інформацію про встаткування, кліматичну ситуацію й аварійні сигнали. Крім того, вони мають доступ до засобів моніторингу й прогнозування обсягу споживаних ресурсів.

DMaaS-пропозиція, містить набір нових хмарних сервісів, що здійснюють прогнозування й запобігання інцидентів і збоїв за допомогою інструментів DCIM. Для цього вони агрегують і аналізують інформацію, що надходить із безлічі центрів обробки даних.

Набір інтерфейсів прикладного програмування дозволяє реалізувати взаємодія з DCIM – інструментами різних виробників, які орендарі послуг можуть інсталювати в зонах розміщення власного встаткування.

Аналітики затверджують, що когнітивні можливості DCIM особливо ефективні при обробці значних обсягів даних, отриманих з різних ЦОД. Аналіз такого роду Великих Даних дозволяє виявити типові ситуації, що виникають при експлуатації різних центрів обробки даних, а потім використовувати їх для оцінки подій, що відбуваються, прогнозування й запобігання небажаних і небезпечних проблем.

Чи відносяться рішення, що розширюють функціональність DCIM, до Інтернету речей? У загальному й цілому – так, тим більше що сьогодні очевидно тенденцію розглядати всі проекти по автоматизації моніторингу будь-якого

встаткування й керування їм як рішення IoT. І центри обробки даних тут не є виключенням.

Проекти IoT припускають використання значних обсягів даних, а також засобів їхньої обробки й аналізу, що дозволяють витягати з накопичених даних нові знання, установлювати не замічені раніше залежності, пророкувати й запобігати поява небажаних ситуацій.

Методи предиктивної аналітики вже застосовуються в комплексах DCIM. Один із прикладів – система керування встаткуванням охолодження ЦОД, заснована на технологіях машинного навчання.

Рішення, у якому використовується мережа бездротових датчиків, визначає взаємозв'язок між такими змінними, як температура в стійках з IT-системами, налаштування режимів роботи охолодного встаткування, його потужність, параметри резервування, енергоспоживання й ризики відмови.

Однак, як підкреслюють фахівці, щоб новітні технології аналізу й обробки даних допомогли реально підвищити ефективність DCIM-рішень, їх необхідно забезпечити якісними даними.

Новий підхід до збору даних, необхідних для керування інфраструктурою ЦОД, а також до впровадження в процеси керування технологій Інтернету речей покладений в основу сервісів. Ці сервіси, покликані спростити створення застосунків для Інтернету речей.

Керована хмарна платформа забезпечує взаємодію підключених пристроїв і із хмарними застосунками, і з іншими пристроями. Вона розрахована на масові індустріальні застосунки й, по завіреннях розроблювача, здатна підтримувати «мільярди пристроїв, обробляти й маршрутизувати трильйони повідомлень».

DCIM не є адресним ринком. Однак подібні сервіси здатні вплинути на розробку нових систем, створення DMaaS-рішень для розподілених середовищ і привести до появи альтернативних пропозицій, здатних конкурувати із продуктами традиційних постачальників DCIM.

3.3 Розробка функціональної схеми

У розробленому програмному забезпеченні архітектуру, у якій об'єднані продукти й технології різних підрозділів, трансформували в трирівневу платформу, що опирається на концепцію Інтернету речей. Як затверджується, ця платформа дозволяє й фахівцям самої компанії, і її партнерам, і кінцевим користувачам розробляти масштабовані рішення, де інформаційні технології сполучаються з операційними.

Система IoT у дата-центрах складається із трьох рівнів:

– пристрої, які підключаються;

– рішення для збору й первинної обробки їхніх даних;

– застосунки, сервіси й засоби аналітики.

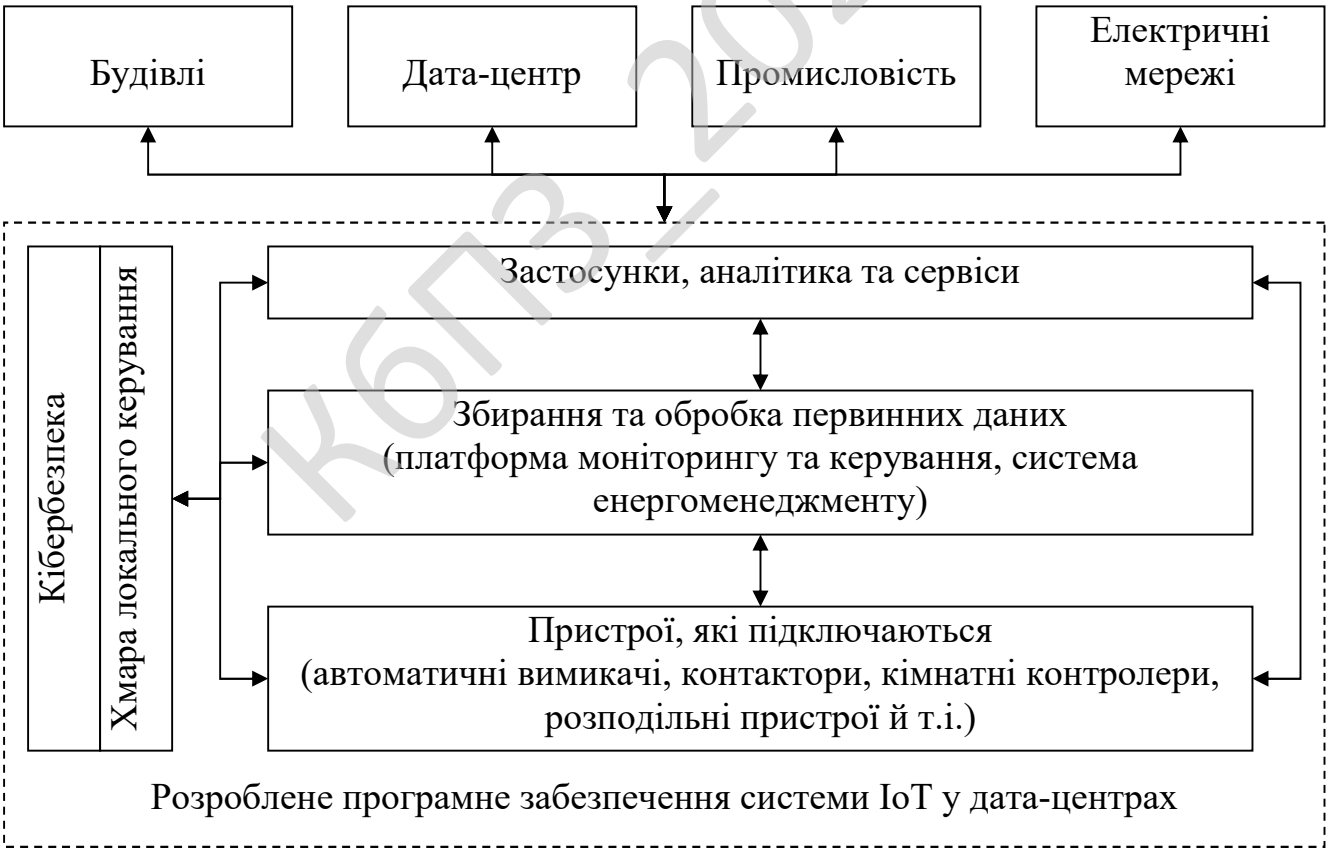

Рисунок 3.2 – Функціональна схема системи

На нижньому рівні перебувають інженерне встаткування, системи безпеки, засоби моніторингу навколишнього середовища. Установлені тут мережні інтерфейси й здатні передавати значні обсяги інформації, що описує їхній стан і працездатність.

Отримані дані збираються й обробляються комплексами EdgeControl, відповідальними за моніторинг устаткування й керування ресурсами ЦОД. Такі функції виконуються класичними локальними системами Data Center Infrastructure Management, архітектура яких обмежується першими двома рівнями.

Тепер до них додаються рішення третього рівня, що містять засоби для моделювання процесів експлуатації центрів обробки даних, планування їхніх ресурсів, а крім того, включаються цифрові сервіси, що зв'язують фізичну інфраструктуру ЦОД із хмарою.

Система IoT у дата-центрах IT збирає дані від інфраструктурного встаткування різних виробників і розміщає їх у хмарному сховищі, де формується, обробляється й аналізується «озеро даних системи IoT у дата-центрах».

До переваг такого підходу розроблювачі рішень системи IoT у дата-центрах відносять можливість збору даних з множини джерел, їхню агрегацію й аналіз на основі технологій Великих Даних і машинного навчання.

Дослідження масивів накопичених «історичних» даних відкриває можливість для прогнозування й запобігання потенційних проблем. Сервіси системи IoT у дата-центрах IT дозволяють використовувати режим, що аналітики назвали «керування центром обробки даних як сервіс» (Data Center Management as a Service, DMaaS). Інструменти DMaaS переносять у хмару ряд функцій локальних систем DCIM (у тому числі дистанційний моніторинг устаткування, енерговитрат і параметрів навколишнього середовища), здійснюють обробку даних і формування звітів і надають інші можливості. Вони здатні сповіщати персонал ЦОД про події, що відбуваються, направляючи SMS або повідомлення в

мобільні застосунки. Як думають, сервіси DMaaS розширюють можливості служб експлуатації й відкривають доступ до функціональності DCIM навіть для невеликих ЦОД, де необхідних для цього ресурсів раніше не було.

3.4 Розробка діаграми процесів

Розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3. Основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей. Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

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

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

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

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

Діаграми потоків даних містять чотири типи елементів:

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

– Сховища даних (репозиторії).

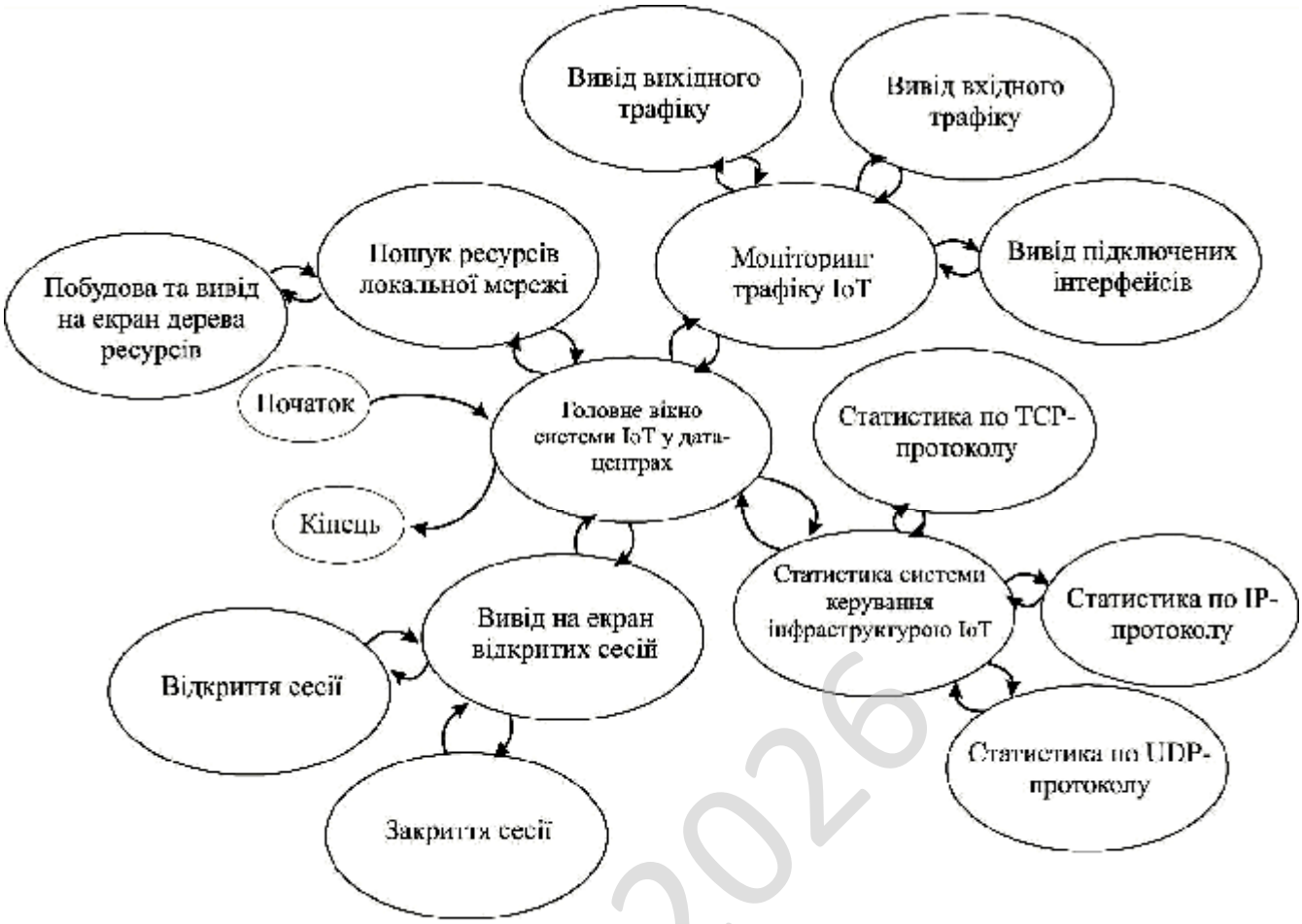

Рисунок 3.3 – Діаграма взаємодії процесів

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

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

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач. Термінатор це елемент

відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія. Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції. Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу.

Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи. Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані. Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи). З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

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

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма

буде об’єктно-орієнтована що вимагає високого рівня декомпозиції задач на класи.

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

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

Систе́ма керування ба́зами да́них (СКБД, Database Management System, DBMS) – набір взаємопов'язаних даних (база даних) і програм для доступу до цих даних. Надає можливості створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних.

Першим поколінням СКБД прийнято вважати ієрархічні й мережеві системи. Ці системи отримали широке поширення в 1970-х роках, а першою комерційною системою цього типу була система IMS компанії IBM.

У 1980-х роках ці системи були витіснені системами другого покоління – повсюдно використовуваними і донині реляційними СКБД. У цих системах використовувалися непроцедурні мови управління даними (SQL) і передбачався значний ступінь незалежності даних. Реляційні системи внесли значні удосконалення в управління даними: графічний користувацький інтерфейс (GUI), клієнт-серверні застосунки, розподілені бази даних, паралельний пошук даних та інтелектуальний аналіз даних.

Але вже до кінця 1980-х років існуюча тоді реляційна модель перестала задовольняти розробників в силу низки обмежень. Відповіддю на зростаючу складність програм баз даних стали два нових напрямки розвитку СКБД: об'єктно-орієнтовані СКБД і об'єктно-реляційні СКБД.

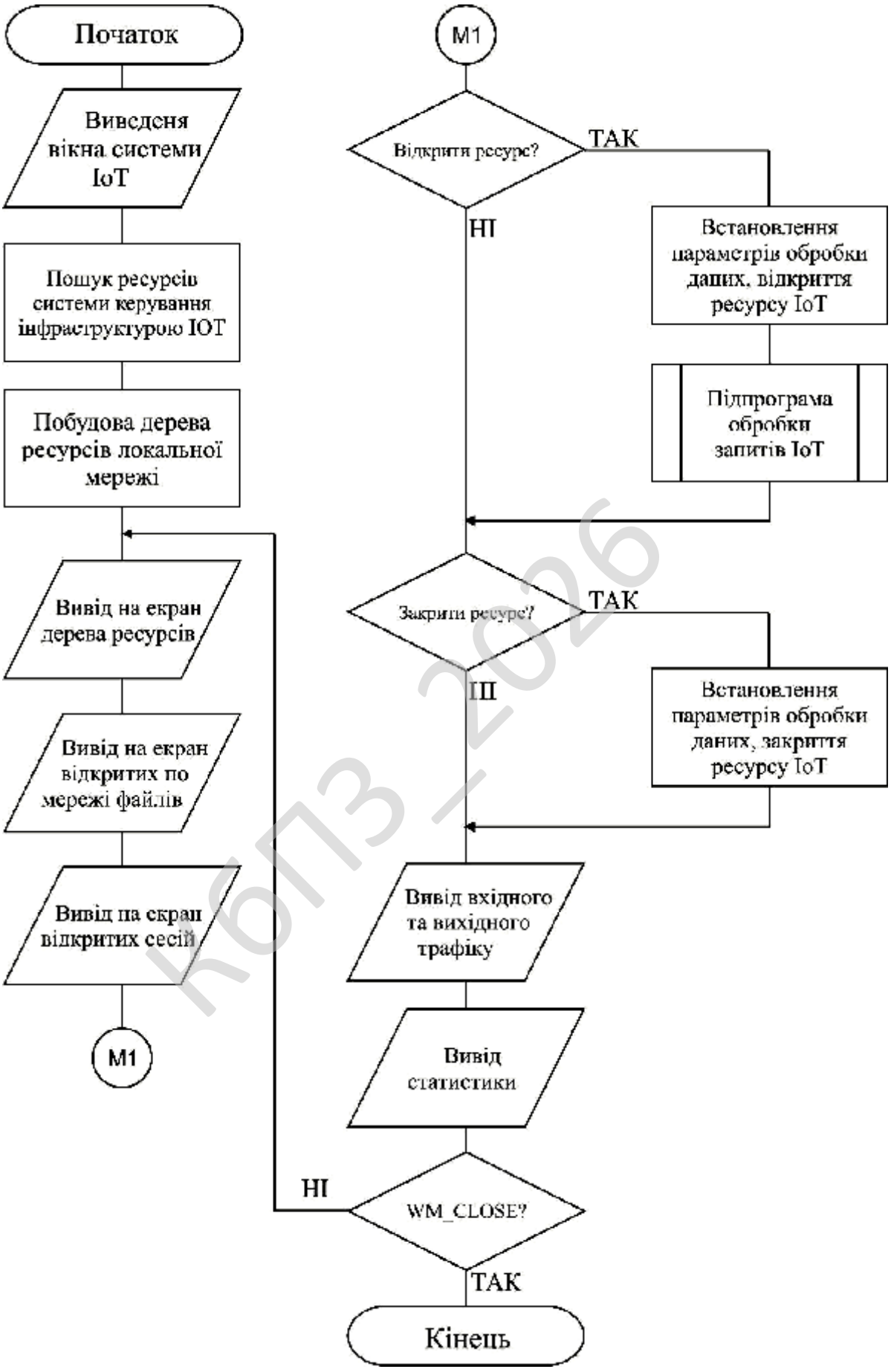

Рисунок 4.1 – Блок-схема основної програми

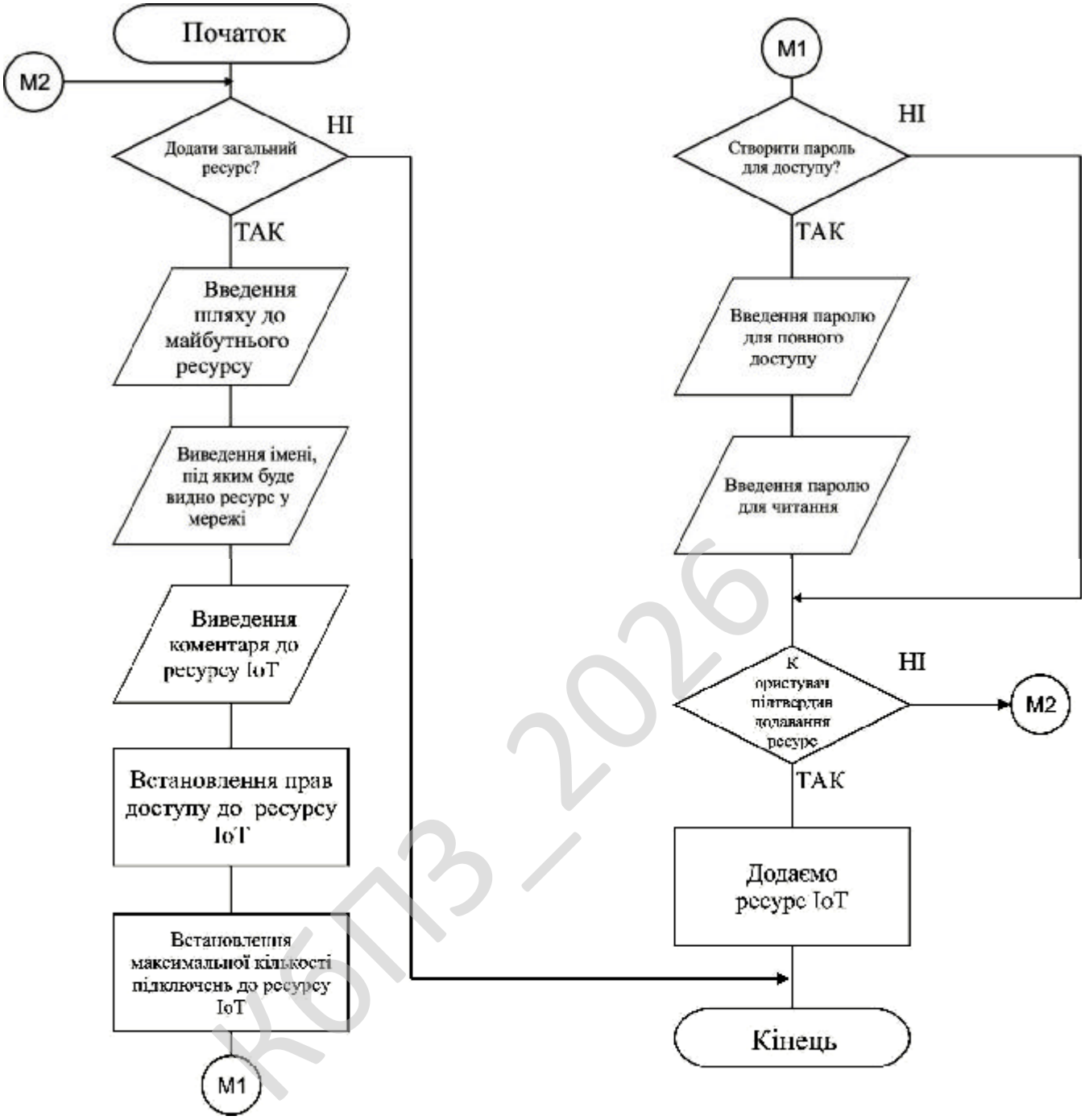

Рисунок 4.2 – Блок-схема роботи підпрограми

У 1991 був утворений консорціум ODMG, основною метою якого стало вироблення промислового стандарту об'єктно-орієнтованих баз даних. Між 1993 та 2001 роками ODMG опублікувала п'ять ревізій своїх специфікацій. Остання версія стандарту має індекс 3.0, після чого група розпустилася. До кінця 1990-х існувало близько десяти компаній, що виробляли комерційні продукти, що

позиціонуються на ринку як ООСКБД. Найбільш відомими системами даного класу стали Objectivity, Versant виробництва однойменних компаній, а також СКБД Jasmine, випущена компанією CA. Незважаючи на переваги, що дозволяють ефективніше вирішувати певний ряд завдань, об'єктно-орієнтовані системи так і не змогли завоювати значущу частку ринку СКБД, залишившись «нішевим» продуктом.

Постачальниками традиційних реляційних СКБД також була проведена значна робота з об'єднання об'єктно-орієнтованих і реляційних систем. Розробники постаралися розширити мову SQL, щоб включити в неї концепції об'єктно-орієнтованого підходу, зберігаючи переваги реляційної моделі (об'єктні розширення мови SQL були зафіксовані в стандарті SQL:1999).

Основний принцип – це еволюційний розвиток можливостей СКБД без поломки попередніх підходів та зі збереженням наступності з системами попереднього покоління.

Поняття СКБД третього покоління, якими, власне кажучи, і є об'єктно-реляційні СКБД, з'явилося після опублікування групою відомих фахівців в області баз даних «Маніфесту систем баз даних третього покоління». Основні принципи СКБД третього покоління, позначені в маніфесті:

Крім традиційних послуг з управління даними, СКБД третього покоління повинні забезпечити підтримку розвиненіших структур об'єктів і правил. Розвинутіша структура об'єктів характеризує засоби, необхідні для зберігання і маніпулювання нетрадиційними елементами даних (тексти, просторові дані, мультимедіа).

СКБД третього покоління повинні включити в себе СКБД другого покоління. Системи другого покоління внесли вирішальний вклад у двох областях – непроцедурний доступ за допомогою мови запитів SQL і незалежність даних. Ці досягнення обов'язково повинні враховуватися в системах третього покоління. СКБД третього покоління повинні бути відкриті для інших підсистем. Це включає оснащення різноманітними інструментами підтримки прийняття

рішень, доступом з багатьох мов програмування, інтерфейсами до існуючих популярних систем і бізнес-застосунків, можливістю запуску програм з бази даних на іншій машині і розподілені СКБД. Весь набір інструментів і СКБД має ефективно функціонувати на різноманітних апаратних платформах з різними операційними системами. Крім того, СКБД, що розраховує на широку сферу застосування, повинна бути оснащена мовою четвертого покоління (4GL).

У середині 1990 років було лише кілька дослідних прототипів СКБД, які поєднали найкращі риси реляційних і об'єктно-орієнтованих СКБД. Першим комерційним продуктом, якому були властиві об'єктно-реляційні риси, став Universal Server компанії Informix(згодом була поглинена IBM). В даний час більшість цих ідей вже втілено в реальних комерційних рішеннях, в тому числі і в продуктах основних постачальників СКБД (Oracle Database і IBM DB2).

Розвиток індустрії систем керування базами даних базується на значних фундаментальних наукових дослідженнях. Найчастіше, між самими дослідженнями та їхньою конкретною реалізацією в прикладних рішеннях минають роки, а іноді й десятиліття. Роботу в області управління даними проводять як університетські дослідницькі групи (MIT, Berkeley), так і центри розробок основних постачальників СКБД (Oracle, IBM, Microsoft). Інвестування в управління даними – це довгострокове, і разом з тим, вигідне вкладення коштів. В даний час дослідники мають у своєму розпорядженні засоби, що дозволяють ефективно реалізувати найскладніші запити, що маніпулюють терабайтами й петабайтами різних даних. Основними тенденціями, які дали привід для проведення різних масштабних досліджень в області баз даних стали:

Експонентний ріст даних. Обсяг даних, у тому числі синтетичних, що генеруються автоматизованими системами, значно зріс. Збільшилося і число прикладних областей, в яких вимагається обробка великих обсягів даних. До таких областей тепер відносяться не тільки традиційні корпоративні програми, пошук у веб, але також і наукові дослідження, обробка природних мов, аналіз соціальних мереж тощо. Значне ускладнення структур використовуваних даних.

Прості види даних у вигляді чисел і символьних рядків стали доповняться численною мультимедійною інформацією, просторовими, процедурними даними та великою кількістю інших складних форматів.

Широке поширення дешевих високопродуктивних апаратних засобів. Щорічно ми спостерігаємо зростання обчислювальних можливостей мікропроцесорів, збільшення ємності і зниження вартості доступних і зручних в експлуатації пристроїв дискової оперативної пам'яті.

Активний розвиток засобів комунікації та «всесвітньої павутини» World Wide Web. WWW стає єдиним інформаційним середовищем, що пронизує весь світ і об'єднує величезне число користувачів та електронних пристроїв.

Поява нових важливих областей застосування СКБД. У першу чергу, це пов'язано з інтелектуальним аналізом даних, сховищами даних, а останнім часом – з паралельними обчисленнями і хмарними технологіями.

Основні характеристики СКБД

1. Контроль за надлишковістю даних.
2. Несуперечливість даних.
3. Підтримка цілісності бази даних (коректність та несуперечливість).
4. Цілісність описується за допомогою обмежень.
5. Незалежність прикладних програм від даних.
6. Спільне використання даних.
7. Підвищений рівень безпеки.

Можливості СКБД

1. Дозволяється створювати БД (здійснюється за допомогою мови визначення даних DDL (Data Definition Language)).

2. Дозволяється додавання, оновлення, видалення та читання інформації з БД (за допомогою мови маніпулювання даними DML, яку часто називають мовою запитів).

3. Можна надавати контрольований доступ до БД за допомогою: Системи забезпечення захисту, яка запобігає несанкціонованому доступу до БД; Системи

керування паралельною роботою прикладних програм, яка контролює процеси спільного доступу до БД; Система відновлення – дозволяє відновлювати БД до попереднього несуперечливого стану, що був порушений в результаті збою апаратного або програмного забезпечення.

Основні компоненти середовища СКБД

1. Апаратне забезпечення.

2. Програмне забезпечення.

3. Дані.

4. Процедури – інструкції та правила, які повинні враховуватись при проектуванні та використанні БД.

5. Користувачі: адміністратори даних (керування даними, проектування БД, розробка алгоритмів, процедур) та БД (фізичне проектування, відповідальність за безпеку та цілісність даних); розробники БД; прикладні програмісти; кінцеві користувачі.

Архітектура СКБД

Існує трирівнева система організації СКБД ANSI-SPARC, при якій існує незалежний рівень для ізоляції програми від особливостей представлення даних на нижчому рівні.

Рівні:

1. Зовнішній – представлення БД з точки зору користувача.

2. Концептуальний – узагальнене представлення БД, описує які дані зберігаються в БД і зв'язки між ними. Підтримує зовнішні представлення, підтримується внутрішнім рівнем.

3. Внутрішній – фізичне представлення БД в комп'ютері.

Логічна незалежність – повна захищеність зовнішніх моделей від змін, що вносяться в концептуальну модель.

Фізична незалежність – захищеність концептуальної моделі від змін, які вносяться у внутрішню модель.

Firebird (також Firebird SQL) – компактна, крос-платформова, вільна реляційна система керування базами даних, що реалізує більшість функцій стандарту SQL 2003. Вона може запускатись на більшості UNIX-подібних систем (в тому числі Linux та FreeBSD) та Windows.

Основні можливості

Відповідність вимогам ACID: Firebird спеціально спроектовано таким чином, щоб задовільняти вимоги «атомарності, несуперечності, ізоляції та довговічності» транзакцій «Atomicity, Consistency, Isolation and Durability».

Версійна архітектура: основна особливість Firebird – версійна архітектура, що дозволяє серверу обробляти різні версії одного запису в будь-який час таким чином, що кожна транзакція бачить свою версію даних, не заважаючи сусіднім. Таким чином, транзакції, що читають, не блокують транзакції, що пишуть і навпаки. Окрім того, це дає можливість відмовитись від логу транзакцій і таким чином зменшити ймовірність пошкодження службової інформації бази даних.

Збережені процедури: за допомогою мови PSQL (процедурна SQL) можна створювати складні збережені процедури для обробки даних на боці сервера. Таким чином можна виносити на сторону сервера значну частину бізнес-логіки програмного пакету чи формувати дані для звітів.

Події: збережені процедури та тригери можуть ґенерувати події, на які, в свою чергу, може підписатися клієнтська програма і відповідним чином їх обробляти.

Генератори: дають можливість просто реалізовувати автоінкрементні поля; оскільки вони працюють незалежно від транзакцій, то можуть використовуватись для генерації первинних ключів чи керування тривалими запитами в інших транзакціях.

Бази даних в режимі «лише читання»: спрощують поширення даних наприклад на компакт-дисках, особливо в поєднанні з вбудованою (embedded) версією сервера.

Повний контроль над транзакціями: одна клієнтська програма може одночасно виконувати декілька транзакцій, включно з різними рівнями ізоляції. Окрім того, доступні протокол двофазного підтвердження транзакцій (що забезпечує гарантовану стійкість при роботі з кількома БД), оптимістичне блокування даних (блокується не вся сторінка даних, а лише змінені записи), точки збереження транзакцій та автономні транзакції (починаючи з версії 2.5).

Резервне копіювання на лету: завдяки в тому числі і версійній архітектурі немає потреби зупиняти сервер для резервування бази даних. Процес резервного копіювання зберігає стан бази даних на момент початку резервування, не шкодячи роботі інших клієнтів. Додатково існує можливість інкрементного резервування бази даних (починаючи з версії 2.0).

Тригери: для будь-якої таблиці можна назначити декілька тригерів, що спрацюють до чи після додавання, оновлення чи вилучення даних. У тригерах використовується мова PSQL, що дозволяє задавати початкові значення даних, перевіряти їх цілісність, збуджувати події та ін. Починаючи з версії 1.5 у Firebird з'явились універсальні тригери, що дозволяють обробляти вставку, оновлення та вилучення даних в одному місці.

Зовнішні функції: можна реалізувати за допомогою будь-якої мови програмування та у вигляді бібліотек користувацьких функцій (англ. UDF – User Defined Function) під'єднаних до сервера.

Набори символів: Firebird підтримує багато міжнародних наборів символів з багатьма варіантами сортування, зокрема типовий для українських користувачів Windows набір символів Win1251 підтримує три варіанти сортування, в тому числі win1251_ua, що дозволяє коректно сортувати отримані з бази дані з українськими символами на боці сервера. Окрім того, Firebird повністю підтримує Unicode, що дає можливість працювати з будь-якими наборами символів.

Firebird повністю підтримує стандарт SQL-92 та реалізує більшу частину стандартів SQL-99 і SQL-2003. Сюди входять вирази DML/DDL, об'єднання запитів, вирази UNION, DISTINCT, підзапити (IN, EXISTS), агрегатні функції

(AVG, SUM, MIN, MAX, LIST (починаючи з версії 2.1)), вбудовані функції (ABS, CEIL, REPLACE, GEN_UUID тощо), обмеження цілісності (PRIMARY KEY, UNIQUE, FOREIGN KEY), та всі загальні типи даних SQL.

Особливості доступу та оброблення даних в Firebird

PSQL (Procedural SQL) – підмножина SQL в Firebird, за допомогою якої пишуться збережені процедури та тригери. Надає можливість програмісту обробки даних у процедурному стилі – наприклад, за допомогою циклів.

DSQL (Dynamic SQL) – підмножина SQL, за допомогою якої здійснюються запити до даних. Підтримуються неіменовані параметри.

```
select some_field1 from some_table where some_field2=? and some_field3
```

Доступ до баз даних Firebird може здійснюватися через розподілену бібліотеку доступу (dll або so, залежно від платформи) – такий спосіб є стандартним; сама бібліотека є в комплекті дистрибутиву. Також є можливість доступу через java і .net провайдери.

Бібліотека доступу реалізує набір функцій для маніпуляції з даними, які можуть бути експортовані і використані в будь-якій мові програмування.

Окрім того, є достатньо багато обгорток під різні мови програмування для цієї бібліотеки, що звільняють програміста від рутинної низькорівневої роботи. Так, для Delphi популярними є бібліотеки IBX, FibPlus, UIB. Для C++ – IBPP. Підтримка вбудована і в популярні скриптові мови, такі як PHP і Python.

Обмеження Firebird

Розмір бази даних – необмежено (залежить від можливостей файлової системи). Розмір таблиці бази даних – 32 ТБ. Максимальна ширина вибірки (сумарно всі поля; не враховуючи блоби; враховується фактичний розмір рядкових даних) – 64 Кб. Індексів на таблицю – 850. Довжина об'єкта метаданих (назва таблиці, процедури тощо) – 27 символів.

Апаратно-програмні вимоги та обмеження

Firebird існує у версіях для Unix (Linux, FreeBSD, Solaris, MacOS, HP-UX) та Windows і вимоги до апаратного забезпечення залежатимуть також від типу ОС, котра обслуговує сервер. Окрім того вимоги знаходяться в прямій залежності

від очікуваного завантаження сервера баз даних, обсягу оброблюваних даних та кількості одночасно працюючих користувачів і говорити про конкретні цифри непросто. Проте загалом ці вимоги доволі низькі: при незначних навантаженнях та обсягах баз даних можна очікувати пристойної роботи на сервері з центральним процесором частотою 100–200 МГц та обсягом оперативної пам'яті 96-128 МБ.

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

Подія може бути визначена як значна зміна стану. Наприклад, коли споживач купує автомобіль, стан автомобіля змінюється з "на продаж" до "продано". Архітектура системи дилера автомобілів може трактувати цю зміну стану як подію, поява якої може стати відомою іншим програмам даної архітектури.

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

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

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

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

Стоки несуть відповідальність за здійснення реагування на появу події. Реакція не завжди може бути повністю забезпечена самим стоком. Наприклад, стік, може бути відповідальним лише за фільтрацію, трансформацію і відправку

події до іншого компонента або він може забезпечити повністю самостійну реакцію на таку подію. Перша категорія стоків може бути заснована на традиційних компонентах, таких як проміжне програмне забезпечення, орієнтоване на обробку повідомлень (message oriented middleware, MOM), в той час, як друга категорія стоків (самостійна реакція в режимі он-лайн) може вимагати більш придатної платформи (фреймворку) для виконання транзакцій.

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

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

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

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

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

Подія може складатися з двох частин: заголовка події та тіла події. Заголовок події може включати в себе інформацію таку як, наприклад, назва події, часова мітка події і тип події. Тіло події – це частина, яка описує факт, що стався в дійсності. Тіло події не слід плутати з шаблоном або логікою, яка може бути застосована як реакція на саму події.

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

Генератор подій.

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

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

Канал подій.

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

Це може бути з'єднання TCP/IP або вхідний файл будь-якого типу (простий текст, формат XML, e-mail тощо). В один і той же час може бути відкрито кілька каналів подій. Як правило, оскільки обробник подій повинен працювати в режимі, наближеному до реального часу, канали подій зчитуються асинхронно. Події зберігаються в черзі, очікуючи наступної обробки механізмом обробки подій.

Механізм обробки подій.

Механізм обробки подій (event processing engine) є місцем, де подія ідентифікується і вибирається відповідна реакція на нього, яка потім виконується. Це також може призвести до породження ряду тверджень. Якщо подія, яка надійшла до механізму обробки подій, є наприклад такою «Запаси продукту ID досягли нижнього допустимого рівня», це може ініціювати, наприклад, такі реакції як «Замовити продукт ID» і «Сповістити персонал».

Наступна подійно-орієнтована дія (післядія).

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

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

Проста обробка подій.

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

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

Обробка потоку подій.

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

Обробка складних подій.

Обробка складних подій (Complex event processing (CEP)) дозволяє за шаблонами простих і звичайних подій проводити аналіз того, чи наступила складна подія. Обробка складних подій полягає в оцінюванні взаємного впливу

подій і в наступному виконанні дій. При цьому, типи подій (відомих або звичайних) можуть перетинатись, а події можуть виникати протягом тривалого періоду часу.

Кореляція подій може бути причинною, тимчасовою або просторовою. CEP вимагає використання складних інтерпретаторів подій, визначення і підбору шаблонів подій, а також відповідних кореляційних методів. Обробка складних подій зазвичай використовується для виявлення і реагування на аномальну поведінку, загрози і можливості у бізнесі.

Хоча я реалізовував програму сам, було використано підходи Scrum для саморозвитку та пришвидшенню розробки, розглянемо цей метод. Scrum – підхід управління проектами для гнучкої розробки програмного забезпечення. Скрам чітко робить акцент на якісному контролі процесу розробки.

Підхід вперше описали Гіротака Такеучі та Ікуджіро Нонака в статті The New New Product Development Game (Гарвардський Діловий Огляд, січ–лют 1986). Вони відзначили, що проекти, над якими працюють невеликі, крос–функціональні команди, зазвичай систематично продукують кращі результати, і пояснили це, як «підхід регбі». У 1991 році ДеҐрейс та Шталь у книжці Злі проблеми, справедливі рішення послалися на цей підхід, як на Scrum (штовханина; сутичка навколо м'яча (у регбі)), спортивний термін, згаданий в статті Такеучі і Нонака. Кен Швабер на початку 1990–х використовував підхід який привів Scrum в його компанію.

Вперше метод Scrum було представлено на загальний огляд задокументованим, чітко сформульованим та описаним спільно Сазерлендом та Швабером на OOPSLA'96 в Остіні. Швабер та Сазерленд протягом наступних років працювали разом щоб обробити та описати весь їхній досвід та найкращі практичні зразки для індустрії в одне ціле, в ту методологію, що відома сьогодні як Scrum. Швабер об'єднав зусилля з Майком Бідлом в 2001, щоб детально описати метод в книжці Agile Software Development with SCRUM. Не зважаючи на те, що для Scrum нарікли долю управління проектами з розробки ПЗ, він може

також використовуватися в роботі команд обслуговувань програмного забезпечення (software maintenance teams), або як підхід управління розробкою і супроводом програм: Scrum of Scrums.

Scrum – це кістяк процесу, який включає набір методів і попередньо визначених ролей. Головні дійові особи – ScrumMaster, той хто опікується процесами, веде їх і працює як керівник проекту, Власник Продукту, людина, що представляє інтереси кінцевих користувачів та інших зацікавлених в продукті сторін, та Команду, яка включає розробників.

Протягом кожного спринту, 15–30 денного періоду (тривалість визначається командою), працівники створюють функціональний ріст програмного забезпечення.

Набір можливостей, які імплементуються кожного спринту, приходять з етапу, що має назву product backlog (документація запитів на виконання робіт), який має найвищу пріоритетність за рівнем вимог до роботи, що повинна бути виконана.

Запити на виконання робіт (backlog items), що визначені протягом наради з планування спринту (sprint planning meeting), переміщуються в етап спринту. Протягом цієї наради Власник Продукту інформує про завдання, які він хоче, аби були виконані. Тоді Команда визначає, скільки з бажаного вони можуть зробити, щоб завершити необхідні частини протягом наступного спринту. Протягом спринту команда виконує визначений фіксований список завдань (т.з. backlog items). Впродовж цього періоду ніхто не має права змінювати перелік запитів на виконання робіт, що слід розуміти, як заморожування вимог (requirements) протягом спринту.

Product backlog – це документ, який має список вимог до функціональності, які упорядковані згідно зі ступенем важливості. Product backlog представляє список того, що повинно бути реалізовано. Елементи цього списку називається «історіями» (user story) або елементами backlog–у (backlog items). Product backlog відкритий для редагування усім учасникам Scrum–процесу.

Обов'язкові поля:

1. ID – унікальний ідентифікатор, порядковий номер, який використовується для ідентифікації історій у разі їх перейменування.

2. Назва (Name) – стислий опис історії. Він повинен бути однозначним, щоб і розробники і product owner могли зрозуміти, про що йдеться і відрізнити одну історію від іншої.

3. Важливість (Importance) – ступінь важливості даної історії на погляд product owner ’а. Зазвичай являє собою натуральне число, іноді для цієї цілі використовуються числа Фібоначчі. Чим більше значення, тим більше пріоритет.

4. Попередня оцінка (initial estimate) – початкова оцінка об'єму робіт, необхідного для реалізації історії порівняно з іншими історіями. Вимірюється у story point'ах. Приблизно відповідає числу «ідеальних людино-днів».

5. Як продемонструвати (how to demo) – стисле пояснення того, як завершена задача буде продемонстрована у кінці спринта. Дане поле може являти собою код автоматизованого приймального тесту.

Додаткові поля. Іноді, також, використовуються додаткові поля у product backlog, в основному для того, щоб допомогти product owner'у визначитися з його пріоритетами. Категорія (track). Наприклад, «панель управління» чи «оптимізація». За допомогою цього поля product owner може легко вибрати усі пункти категорії «оптимізація» і задати їм низький пріоритет. Компоненти (components) – указує, які компоненти (наприклад, база даних, сервер, клієнт) будуть зачеплені при реалізації історії. Дане поле складається з групи checkbox'ів, які відмічаються, якщо відповідні компоненти потребують змін. Ініціатор запиту (requestor). Product owner може захотіти зберігати інформацію про усіх замовників, зацікавлених у даній задачі. Це потрібно для того, щоб тримати їх у курсі діла про хід виконання робіт. ID у системі обліку помилок (bug tracking ID) – якщо ви використовуєте окрему систему обліку помилок, тоді у описі історії корисно зберігати посилання на всі дефекти, які до неї відносяться. Sprint backlog – містить функціональність, обрану Product Owner із Product Backlog. Всі

функції розбиті по задачах, кожна з яких оцінюється командою. Кожен день команда оцінює об'єм роботи, який необхідно провести для завершення задачі. Burndown chart – показує, скільки вже виконано і скільки ще залишається зробити.

Планування спринта (Sprint Planning Meeting)

Проходить на початку нової ітерації Спринта:

– Із Product Backlog обираються задачі, зобов'язання по виконанню яких за спринт приймає на себе команда;

– На основі обраних задач створюється Sprint Backlog. Кожна задача оцінюється у ідеальних людино–годинах;

– Рішення задачі не повинно займати більше 12 годин або одного дня. При необхідності задача розбивається на підзадачі;

– Обговорюється та визначається, яким чином буде реалізовано цей об'єм робіт;

– Тривалість наради обмежена зверху 4–8 годинами в залежності від тривалості ітерації, досвіду команди тощо;

– (перша частина наради) Беруть участь Product Owner + Команда: обирають задачі із Product Backlog;

– (друга частина наради) Бере участь лише команда: обговорюють технічні деталі реалізації, наповнюють Sprint Backlog.

Щоденна нарада (Daily Scrum Meeting)

Відбувається кожен день протягом спринта. Є «пульсом» ходу спринта. Нараді властиві наступні обмеження:

– починається точно вчасно;

– всі можуть спостерігати, але говорять тільки обрані;

– триває не більш ніж 15 хвилин;

– проводиться в одному і тому ж місці протягом одного спринта.

Протягом наради кожен член команди відповідає на 3 запитання:

– Що зроблено з моменту попередньої щоденної наради?

– Що буде зроблено з моменту поточної наради до наступної?

– Які проблеми заважають досягненню цілей спринта? (Над рішенням цих проблем працює ScrumMaster. Зазвичай це рішення проходить за рамками щоденної наради і у складі осіб, що безпосередньо займаються даною перешкодою.)

Демонстрація (Sprint Review Meeting):

– Проходить у кінці ітерації (спринта).

– Команда демонструє внесок функціональності до продукту всім зацікавленим особам.

– Залучається максимальна кількість глядачів.

– Усі члени команди беруть участь у демонстрації (одна людина на демонстрацію або кожен показує, що зробив за спринт).

– Обмежена 4–ма годинами в залежності від тривалості ітерації і змін у продукті.

Ретроспектива (Sprint Retrospective):

– Члени команди висловлюють свою думку про минулий спринт.

– Відповідають на два основних запитання: Що було зроблено добре у минулому спринті?; Що потрібно покращити в наступному?.

– Виконують покращення процесу розробки (вирішують питання та фіксують вдалі рішення).

– Обмежена 1-3ма годинами.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму DES. DES є класичною мережею Фейштеля із двома гілками. Дані шифруються 64-бітними блоками, використовуючи 56-бітний ключ. Алгоритм перетворить за кілька раундів 64-бітний вхід в 64-бітний вихід. Довжина ключа дорівнює 56 бітам. Процес шифрування складається із чотирьох етапів. На першому з них виконується початкова перестановка (IP) 64-бітного

вихідного тексту (забілювання), під час якої біти переставляються у відповідності зі стандартною таблицею. Наступний етап складається з 16 раундів однієї й тої ж функції, що використовує операції зрушення й підстановки. На третьому етапі ліва й права половини виходу останньої (16-й) ітерації міняються місцями. Нарешті, на четвертому етапі виконується перестановка IP^{-1} результату, отриманого на третьому етапі. Перестановка IP^{-1} інверсна початковій перестановці.

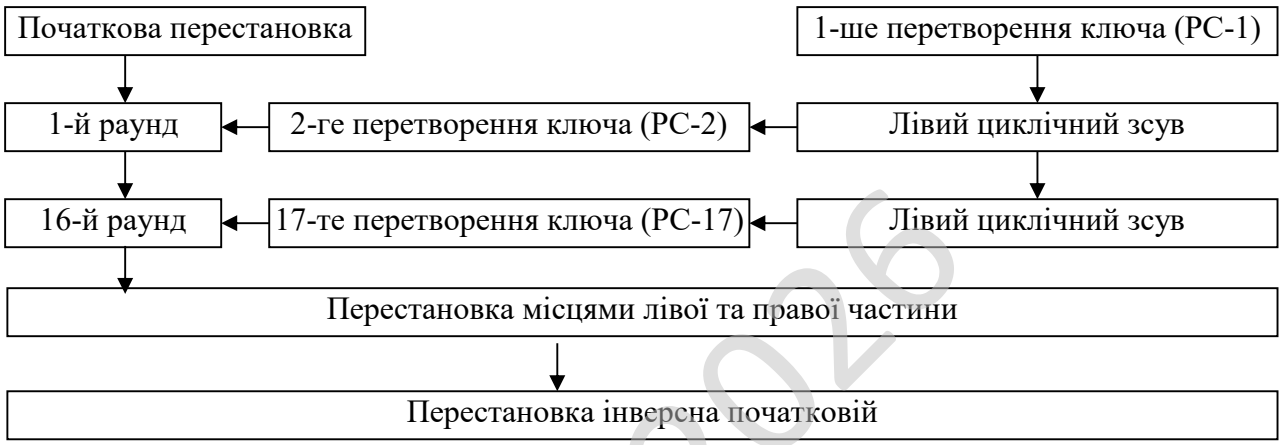

Рисунок 4.3 – Загальна схема DES

Праворуч на рисунку показаний спосіб, яким використовується 56-бітний ключ. Спочатку ключ подається на вхід функції перестановки. Потім для кожного з 16 раундів підключ K_i є комбінацією лівого циклічного зрушення й перестановки. Функція перестановки та сама для кожного раунду, але підключи K_i для кожного раунду виходять різні внаслідок повторюваного зрушення біт ключа.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у бакалаврської дипломної роботі система IoT у дата-центрах. 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; Розділу обрання групи; Розділу виведення результату роботи системи.

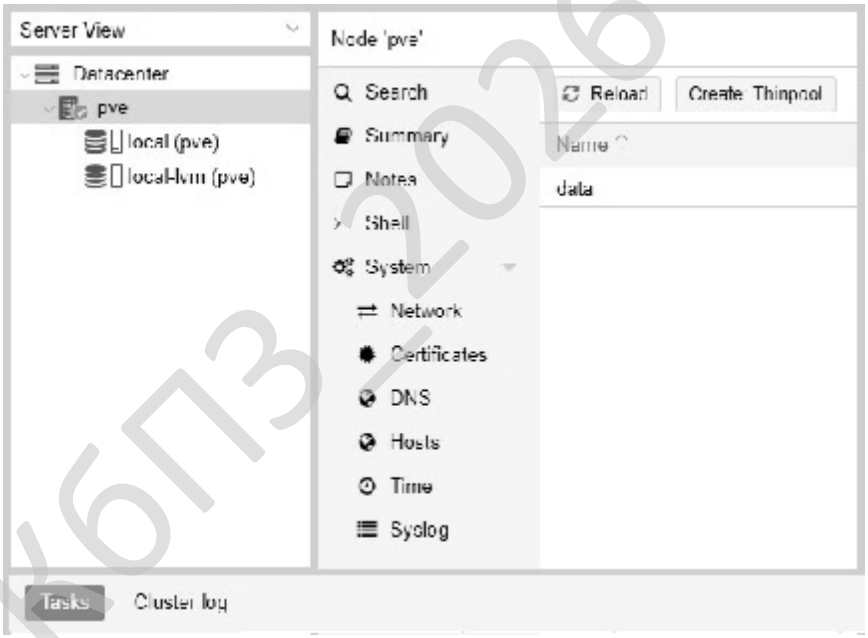

Рисунок 5.1 – Головне вікно ПЗ

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.

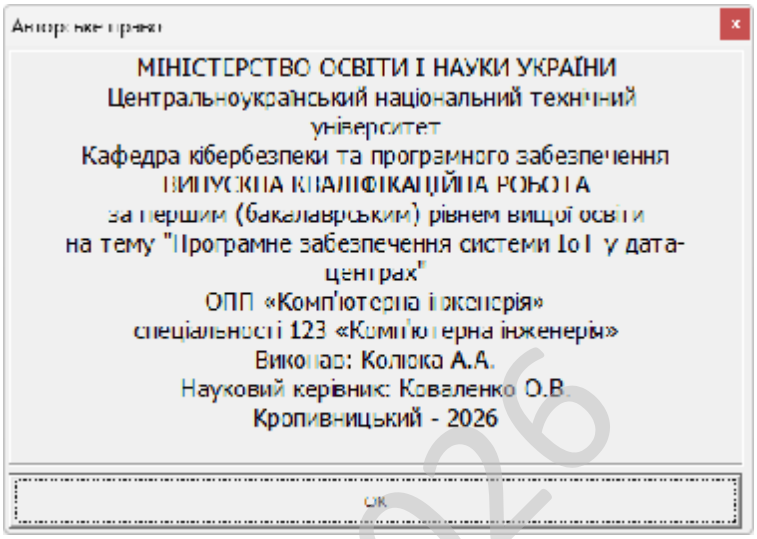

Рисунок 5.2 – Авторське право

Методологія Інтернету речей дозволить впроваджувати моделі керування, засновані на даних (Data-Driven Model), використовувати для вивчення й розуміння процесів статистичну інформацію, що характеризує роботу багатьох ЦОД, застосовуючи для її аналізу хмарні ресурси. Керування на основі даних стає одним з перспективних напрямків розвитку сучасних центрів обробки даних. Такі системи здатні підвищити стабільність роботи ЦОД, розширити гарантії в угодах про якість обслуговування (Service Level Agreement), зменшити завантаження експлуатаційного персоналу. Ті можливості, які сьогодні відносять до Інтернету речей, у тім або іншому ступені вже закладені або з'являються в сучасних програмних комплексах керування інфраструктурою ЦОД (Data Center Infrastructure Management, DCIM), які забезпечують «видимість» і планування використовуваних ресурсів, оптимізацію температурних режимів і енерговитрат, а також зниження сукупної вартості володіння ЦОД. Оскільки ЦОД всі частіше

стають великими індустріальними об'єктами, у продуктах і технологіях для моніторингу інфраструктури й керування різними системами зацікавлені й розроблювачі комплексів DCIM, і власники комерційних центрів обробки даних (КЦОД). І ті й інші пропонують свої рішення.

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

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

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

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

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

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

– У програмі можуть бути пропущені деякі маршрути.

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

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

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

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

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

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

Обрано умови розповсюдження – Freeware. Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів. Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення. Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи IoT у дата-центрах.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем IoT у дата-центрах.

– Досліджена система IoT у дата-центрах.

– На основі отриманих результатів досліджень створена програмна реалізація системи IoT у дата-центрах.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання IoT у дата-центрах.

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 IoT у дата-центрах. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне

програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм DES.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Peter Hoddie, Lizzie Prader «IoT Development for ESP32 and ESP8266 with JavaScript: A Practical Guide to XS and the Moddable SDK» ISBN-13 (pbk): 978-1-4842-5069-3 ISBN-13 (electronic): 978-1-4842-5070-9

2. STM32CubeMX for STM32 configuration and initialization C code generation. User manual. June 2022. 397 p.

3. І.В.Чихіра, А.Г. Микитишин Конспект лекцій з дисципліни «Програмування систем реального часу» / Укладачі : Чихіра І.В., Микитишин А.Г., – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя , 2016. – 76 с.

4. Jack Ganssle and Michael Barr. 2003. Embedded Systems Dictionary. CMP Books.

5. Технології інтернету речей. Навчальний посібник [Електронний ресурс]: / Б. Ю. Жураковський, І.О. Зенів; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 12,5 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 271 с.

6. Greg Dunko, Joydeep Misra, Josh Robertson, Tom Snyder “A reference guide to the Internet of Things” / 2017 Bridgera LLC, RIoT.

7. Donald Norris “Programming with STM32. Getting started with the Nucleo Board and C/C++” 416 p. 2018.

8. Neil Kolban “Kolban’s book on ESP32”. Texas, USA. 951 p.

9. Вінтенко Б.Ю., Миронець І.В., Смірнов О.А. «Модель комп’ютерно-орієнтованої процедури системи підтримки оперативного персоналу АЕС». *IV Міжнародна науково-практична Інтернет-конференція «Інновації та перспективні шляхи розвитку інформаційних технологій (ІПШРІТ-2025)»* м.Черкаси 25 листопада 2025 року – Черкаси: ЧДТУ.– 2025. – С.101-103.

10. Kuznetsov O., Frontoni E., Kuznetsova Y., Chevardin V., Smirnov O. «Architectural foundations for adaptive security in edge computing systems». *Cybersecurity Defensive Walls in Edge Computing,* 2025. pp. 21-61.

11. Вінтенко, Б.Ю., Миронець, І.В., Смірнов, О.А., Коваленко, О.В., Усік, П.С., Буравченко, К.О., Лисенко, І.А. «Логіко-структурна модель комп'ютерно-орієнтованої процедури системи підтримки оперативного персоналу АЕС». *Кібербезпека: освіта, наука, техніка.* 2025. Том 2 № 30. С. 413-427, 2025.

12. Смірнова, Т.В. «Дослідження методів, моделей та сучасних ІТ-рішень для підтримки технологічних процесів у критичній інфраструктурі держави». *Кібербезпека: освіта, наука, техніка.* 2025. Том 2 № 30. С.195-208, 2025.

13. Kuznetsov, O., Frontoni, E., Kuznetsova, K., Arnesano, M., Smirnov, O. «A secure biometric authentication architecture for blockchain-driven cyber-physical systems». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 193–224.

14. Kuznetsov, O., Atzeni, G., Arnesano, M., Randieri, C., Smirnov, O. «Secure IoT-based smart wheelchair system: From implementation to security enhancement strategy». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 225–257.

15. Kuznetsov, O., Smirnov, O., Kuznetsova, T., Shaikhanova, A., Svatowsky, I. «Privacy-utility trade-offs in IoT networks: A comparative analysis of differential privacy mechanisms for sensor data aggregation». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 589–622.

16. 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.

17. 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.

18. Вінтенко Б.Ю., Смірнов О.А., Миронець І.В., Смірнова Т.В., Коваленко О.В., Мацуй А.М. «Модель шляхів отримання вхідних даних комп'ютерної інтелектуальної системи підтримки оперативного персоналу АЕС». *Центральноукраїнський науковий вісник. Технічні науки.* 2025. Вип. 11(42), ч. II. С.52-62.

19. Вінтенко Б.Ю., Смірнов О.А., Миронець І.В., Смірнова Т.В. «Методи забезпечення відмовостійкості інтелектуальних систем підтримки оператора». *VIII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології",* м. Кропивницький. 24-25 квітня 2025 р. – Кропивницький: ЦНТУ. – 2025. – С. 44-46.

20. Вінтенко, Б., Миронець, І., Смірнов, О., Коваленко, А., Конопліцька-Слободенюк, О., Смірнова, Т., Константинова, Л. «Дослідження застосування систем підтримки оперативного персоналу об'єкту критичної інфраструктури при керуванні енергоблоком АЕС з реактором типу ВВЕР-1000». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 6-26.

21. 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.

22. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки

програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

23. 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.

24. Kuznetsov, O., Kuznetsova, Y., Smirnov, O., Kostenko, O., Zvieriev, V. «Evaluating Hashing Algorithms in the Age of ASIC Resistance». *CEUR Workshop Proceedings*, 2023, 3628, pp. 93-105.

25. 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.

26. 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.

27. Вінтенко Б.Ю., Смірнов О.А., Коваленко А.С., Смірнов С.А., Буравченко К.О. «Дослідження вимог міжнародних стандартів IEC60880 та IEC62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 3(73), С. 155-166.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

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

36. 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.

37. 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.

38. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

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

52. 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

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

54. 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.