Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра Інформаційних управляючих систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження темпоральних методів підтримки прийняття рішень з налагодження ARM-мережних пристроїв
(тема)

Виконавець:
студент _2_ курсу, групи _ІУСТзм-21-1____
Даніїл ЛЯДОВ
(власне ім'я, прізвище)

Спеціальність ____122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми ____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма _Інформаційні управляючі системи та технології
(повна назва освітньої програми)

Керівник _професор каф. ІУС Оксана ЧАЛА___
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри _____ ___Костянтин ПЕТРОВ__
(підпис) (власне ім'я, прізвище)

2022 р.

Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання

Кафедра Інформаційних управляючих систем

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформаційні управляючі системи та технології
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«_21_» _листопада_ 20 22_ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Лядову Данііілу Артемовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження темпоральних методів підтримки прийняття рішень з налагодження ARM-мережних пристроїв

затверджена наказом університету від _14_ __листопада__2022_ р. № _185Стз_

2. Термін подання студентом роботи до екзаменаційної комісії _13.12.2022 р._____

3. Вихідні дані до роботи Удосконалений метод підтримки прийняття рішень з налагодження ARM-мережних пристроїв, Інформаційна технологія автоматизованої підтримки прийняття рішень з налагодження ARM-мережевих пристроїв, на основі темпоральної бази знань, експериментальна перевірка удосконаленого методу.

4. Перелік питань, що потрібно опрацювати в роботі Аналіз задачі підтримки прийняття рішень з налагодження ARM-мережних пристроїв. Формування рішень для налагодження ARM-мережних пристроїв. Розробка технології підтримки прийняття рішень з використанням темпоральних знань. Експериментальна перевірка вдосконаленого методу.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз задачі підтримки прийняття рішень з налагодження ARM-мережних пристроїв	10.10.22 – 17.10.22	виконано
2	Формування рішень для налагодження ARM-мережних пристроїв	18.10.22- 25.10.22	виконано
3	Розробка технології підтримки прийняття рішень з використанням темпоральних знань	26.10.22 – 7.11.22	виконано
4	Експериментальна перевірка	8.11.21 – 20.11.22	виконано
5	Оформлення пояснювальної записки та графічного матеріалу	21.10.22 – 03.12.22	виконано
6	Перевірка на плагіат	06.12.2022	виконано
7	Попередній захист кваліфікаційної роботи	13.12.2022	виконано

Дата видачі завдання _21_ ____листопада_ ___20__ р.

Студент _____
(підпис)

Керівник роботи _____ _____ _____проф. каф ІУС Оксана. ЧАЛА___
(підпис) (посада, власне ім’я, прізвище)

ЗМІСТ

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 85 с., 7 табл., 22 рис., 1 дод , 39 джерел.

ПРИЙНЯТТЯ РІШЕНЬ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ТЕМПОРАЛЬНА ЛОГІКА, ТЕМПОРАЛЬНІ ПРАВИЛА, ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ, ARM-МЕРЕЖЕВІ ПРИСТРОЇ, AMQP, BLUETOOTH, MQTT,WIFI.

Предмет дослідження – методи підтримки прийняття рішень для конфігурування ARM-мережних пристроїв з використанням темпоральних знань.

Мета дослідження – вирішення проблеми вибору процесів налагодження мережевих ARM пристроїв з використанням темпоральних знань.

У роботі проведене дослідження темпоральних методів підтримки прийняття рішень, сформовано методи підтримки прийняття рішень з використанням темпоральних знань, удосконалено методи формування з налагодження ARM-мережних пристроїв з використанням темпоральних знань, розробка програмного модуля (засобів) формування рішення щодо налагодження ARM-мережних пристрої з використанням темпоральних знань, експериментальна перевірка удосконаленого методу.

Галузь застосування задачі, яка розробляється – підприємства будь-якого типу.

ABSTRACT

Explanatory note to the qualification work: 58 pp., 7 tables, 22 figures, 1 appendices. , 39 sources.

AMQP, ARM NETWORK DEVICES, BLUETOOTH, DECISION MAKING, HARDWARE, MQTT, SOFTWARE, TEMPORAL LOGIC, TEMPORAL RULES, WIFI.

The subject of the study is decision support methods for configuring ARM network devices using temporal knowledge.

The purpose of the research is to solve the problem of choosing debugging processes of network ARM devices using temporal knowledge.

In the work, a study of temporal decision-making support methods was carried out, methods of decision-making support using temporal knowledge were developed, formation methods for debugging ARM-network devices using temporal knowledge were improved, development of a software module (tools) for decision-making about debugging ARM-network devices using of temporal knowledge, experimental verification of the improved method.

The field of application of the task being developed is enterprises of any type.

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК І ТЕРМІНІВ

БД – база даних;

БЗ – база знань;

Гб – гігабайт;

ІАД – інтелектуальний аналіз даних;

ІС – інформаційна система;

ІУС – інформаційна управляюча система;

КТЗ – комплекс технічних засобів;

ЛПР – лице, приймаюче рішення;

ОС – операційна система;

ОПР – об’єкт прийняття рішення;

ОУ – об’єкт управління;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

СПР – система прийняття рішень;

СППР – система підтримки прийняття рішень;

СУБД – система управління базами даних;

ТЗ – технічне забезпечення;

AMQP – advanced message queue protocol;

ARM – advanced RISC machine;

Bluetooth – виробнича специфікація бездротових персональних мереж;

CPU – central processing unit;

FTP – file transfer protocol;

HTTP – hypertext transfer protocol;

IoT – internet of things;

LAN – local area network;

Linux – сімейство UNIX-подібних операційних систем;

MQTT – message queue telemetry;

P2P – peer to peer;

RPC – remote process call;

SBC – single board computer;

SSH – secure shell;

TCP/IP – мережева модель передачі даних;

UDP – user datagram protocol;

Wi-Fi – технологія бездротової локальної мережі із пристроями на основі стандартів IEEE 802.11;

Windows – група сімейств комерційних пропрієтарних операційних систем корпорації Microsoft;

x86 – архітектура центрального процесора.

ВСТУП

На даний момент існує величезна кількість програмного забезпечення, яка має клієнт-серверну архітектуру. В основному, це програмне забезпечення базується на Web серверах. У свою чергу, Web сервера базуються на програмному забезпеченні, що використовує стандартні мережеві протоколи для комунікацією з клієнтом. Приладами таких мережевих протоколів є: TCP/IP, UDP, HTTP, FTP, WS. Але також присутні різні реалізації, що базуються на нетипових мережевих протоколах для IoT пристроїв, що можуть бути автономними та потребують невелику кількість електроенергії для виконання специфічних для програмного забезпечення задач (будь то виконання розрахунків, зчитування значень з приєднаних датчиків і т.д.).

У більшості випадків ці пристрої базуються на ARM архітектурі або на мікроконтролерах, які мають можливість з'єднання з клієнтом через мережевий протокол. Ці рішення пропонують низьке використання електроенергії за невелику ціну. Також існує велика кількість різноманітних конфігурацій, які можливо підібрати під конкретну ціль та задачі.

У даній роботі буде розглянуто налаштування пристрою на архітектурі ARM, тому що вони потребують набагато менше зусиль та використовують підхід x86 базованих комп'ютерів, з якими знайома більшість фахівців сектору інформаційних технологій. Також ці пристрої вимагають менше часу для налаштування, ніж пристрої, що базуються на мікроконтролерах.

Але на даний момент не існує конкретного підходу до налаштування ARM-мережених пристроїв. Тому постають такі питання: як налаштувати та підготувати ARM-мережевий пристрій до виконання поставленої задачі? Який мережевий протокол обрати для досягнення поставленої мети? Наскільки ефективним буде використання того чи іншого протоколу? Скільки часу потрібно буде для налаштування при обраному протоколі?

Кваліфікаційна робота присвячена дослідженню темпоральних методів підтримки прийняття рішень з налагодження ARM-мережних пристроїв.

Метою роботи є дослідження та порівняння підходів до налагодження ARM-мережних пристроїв, які можливо поділити на такі етапи: підготовка базового образу операційної системи для конкретного ARM-мережевого пристрою, налаштування операційної системи для роботи з обраним мережевим протоколом, підключення додаткового технічного забезпечення (при необхідності), верифікація результату виконаної роботи.

1 АНАЛІЗ ЗАДАЧІ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ З НАЛАГОДЖЕННЯ ARM-МЕРЕЖНИХ ПРИСТРОЇВ

1.1 Дослідження процесів налагодження мережних ARM пристроїв

ARM пристрої – пристрої, що базуються на ARM процесорах з (зазвичай) вбудованою оперативною пам’яттю та додатковим обладнанням, що необхідне для повного функціонування ARM пристрою. Додаткове обладнання включає до себе як і мікросхеми для організації вводу/виводу по зовнішнім шинам даних, так і для подачі енергоживлення для ARM процесору, пам’яті та додаткових мікросхем.

Основною відмінністю мережевого ARM пристрою від будь якого іншого ARM пристрою є наявність мережевого інтерфейсу або NIC (network interface card) для комунікації з використанням LAN та/або WAN мереж.

У ARM мережевих пристроях серверного (або промислового) класу інтерфейси зазвичай дротові, так як дротові інтерфейси мають низьку затримку передачі даних, споживають меншу кількість електроенергії, мають вищу стабільність та надійність зв’язку. В основному, NIC модуль представлений на даних пристроях у вигляді окремого модулю, який має можливість заміни.

У ARM мережевих пристроях споживчого класу в основному використовуються бездротові методи передачі даних, так як вони більш зручні для використання у повсякденних задачах користувачів. В основному, NIC модуль представлений на даних пристроях у вигляді вбудованого модуля, що поєднує у собі декілька мережевих протоколів (найчастіше у вигляді комбінації WiFi та Bluetooth).

Основні сфери використання ARM мережевих пристроїв можливо поділити на дві основні категорії: споживча категорія та виробнича категорія.

Для споживчої категорії ARM-мережеві пристрої можуть використовуватися для побудування:

– аматорських IoT систем домашнього використання;

– мобільних пристроїв будь-якого призначення з низьким енергоспоживанням;

– аматорських обчислюваних кластерів домашнього використання;

– NAS (network attached storage) домашнього використання.

Для виробничої категорії ARM-мережеві пристрої використовуються для побудування:

– виробничі IoT системи;

– системи моніторингу механічних вузлів на виробничих підприємствах;

– високопродуктивних серверів для обчислення у хмарних системах;

– сервера загального призначення у хмарних системах;

– промислові вбудовані системи.

Основна відмінність між двома основними процесорами полягає в тому, що ARM використовує менше простору і меншу кількість електроенергії, зберігаючи енергію для довшого терміну служби батареї. У той же час x86 забезпечує набагато більше обчислювальної потужності. Вони відрізняються своєю архітектурою, оскільки ARM базується на обчисленні зі скороченим набором інструкцій, а x86 – на обчисленні зі складним набором інструкцій. ARM, що означає Advanced RISC Machines, використовує більш спрощену систему для збереження даних у пам’яті [18].

З точки зору користувача, більшість ПК і ноутбуків використовують архітектуру x86, тоді як смартфони та інші мобільні пристрої, у своїй більшості, базуються на ARM. Компанії Intel і AMD використовують для своїх чіпів архітектуру x86.

ARM використовує мінімальні обчислювальні інструкції, що вимагає меншої кількості транзисторів у схемі. Ця концепція передбачає більше

доступного простору для мікросхем або мікросхеми меншого розміру. Однак процесори Intel розроблені для виконання складних завдань, забезпечуючи при цьому більшу гнучкість.

x86 корисний для обчислювальних пристроїв, які живлять інші обчислювальні пристрої, тоді як мобільні пристрої на архітектурі ARM використовують менше енергії. Той факт, що x86 використовує більш складні інструкції, пояснює, чому ця архітектура споживає більше енергії.

Що стосується інтеграції з операційними системами, то x86 має більшу універсальність. Більшість операційних систем написані для x86. Але через масову популярність смартфонів, ОС для ARM почали ставати більш різноманітними. Microsoft навіть випустила полегшену версію Windows 10, яка працює на процесорах ARM.

З фінансової точки зору, ARM скорочує витрати на масове виробництво за рахунок меншої функціональності та нижчої обчислювальної потужності. ARM пристрої, що мають графічне ядро, мають нижчу тактову частоту, ніж x86, що призводить до меншої пропускної здатності графічного ядра. Обидва типи процесорів доволі давно існують у нашому повсякденному житті та користуються неабияким попитом у різних сферах.

Таблиця 1.1 – Порівняння ARM та X86

	ARM	X86
Розмір інструкцій	Розмір інструкцій уніфікований, виконання інструкцій у один такт.	Розмір інструкцій визначається на основі складностей.
Вибір та декодування	Процес вибору інструкції простіший. Також, доступ до кодів та операндів можливо організувати паралельно.	Процес вибору складний, так як включає до себе декодування різних форматів и декілька режимів доступа до регістрів.

Кінець таблиці 1.1

Споживання електроенергії	Має простий дизайн управляючого модулю с меншим використанням електроенергії	Має складний дизайн управляючого модулю, тому цей модуль споживає більше електроенергії.
Розмір програм	Специфічні функції вимагають багато простих інструкцій, тому програми займають більше місця.	Схожі функції потребують меншої кількості додаткових інструкцій, тому програми будуть займати менше місця.

Перша особливість з якою може зіткнутися потенцій розробник при використанні ARM мережевих пристроїв – відсутність стандартних дистрибутивів ОС для специфічного пристрою, що базується на ARM архітектурі. На даному етапі можна розглянути дистрибутиви Linux, які побудовані для ARM пристроїв.

Процес налаштування можна представити у вигляді наступного алгоритму: На даному етапі будуть перераховані кроки для налаштування ПЗ та ТЗ на ARM мережному пристрої, так як вони повністю включать до себе пункти з налаштування комплексів ПЗ та ТЗ на клієнтському пристрої.

Кроки для налаштування ARM мережевого пристрою:

1. Підготовка базового образу ОС для заливання на накопичувач;
2. Під’єднання накопичувача до ARM SBC;
3. Запуск ARM SBC;
4. Перенесення файлів для ПЗ на ARM SBC;
5. Встановлення додаткового ПЗ для функціонування системи на клієнтському ТЗ;
6. Запуск додаткового ПЗ на клієнтському ТЗ;
7. Встановлення залежності ПЗ для виклику віддалених процедур на ARM SBC;

8. Запуск ПЗ на ARM SBC.

На рисунку 1.2 представлене графічна схема процесу налагодження ARM-мережевого пристрою, описаного вище.

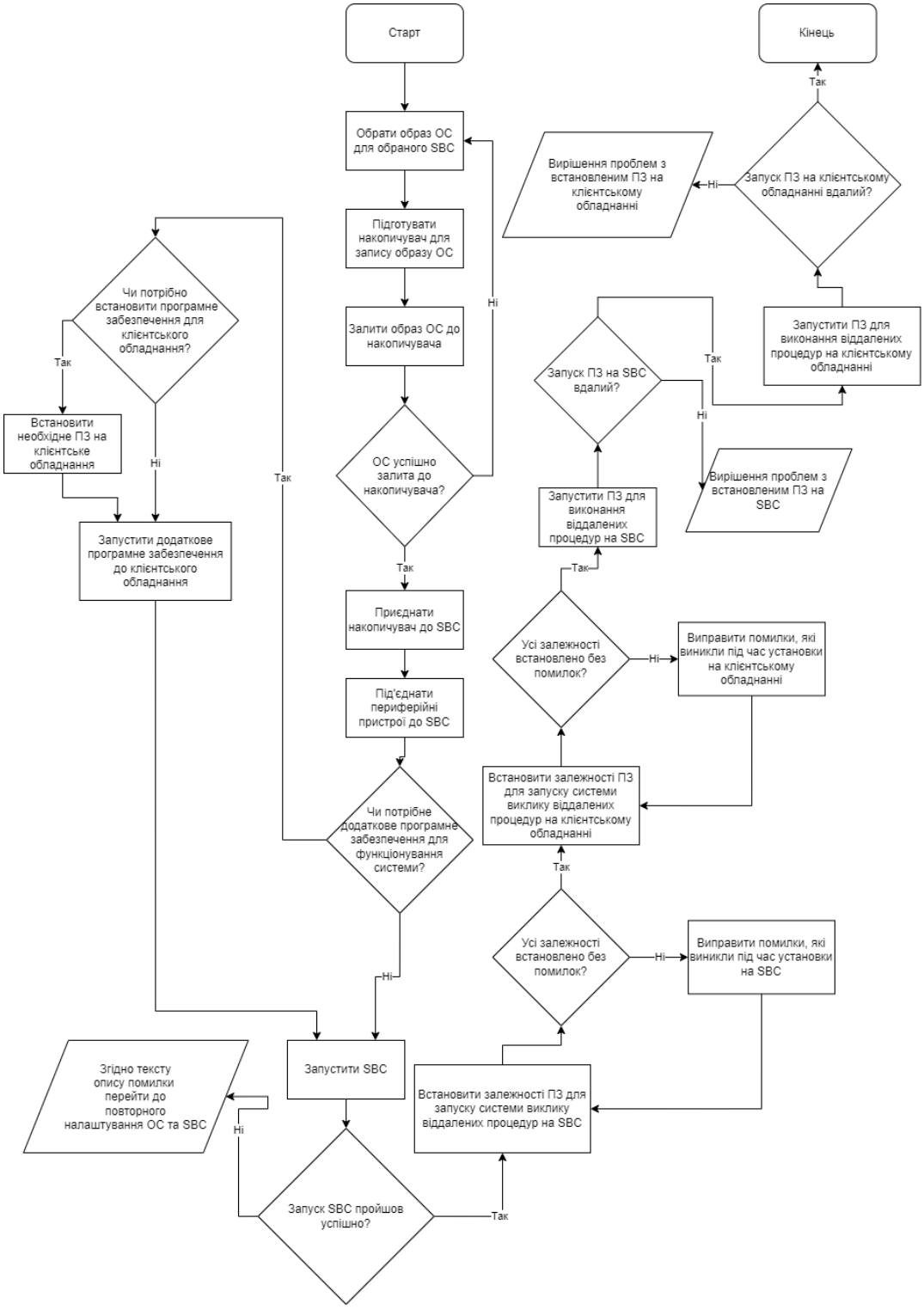

Рисунок 1.1 – Алгоритм налаштування ARM-мережевого пристрою

Даний алгоритм покриває більшість способів налаштування ARM-мережевого пристрою.

1.2 Аналіз систем підтримки прийняття рішень

СППР – це комп’ютерна система, призначена для підтримки діяльності в процесах прийняття рішень, пов’язаних із слабо структурованими та неструктурованими проблемами. Такі системи дозволяють персоналу ОПР здійснювати пошук відповідних даних, створених системами обробки транзакцій та іншими внутрішніми джерелами інформації, і надають доступ до зовнішньої інформації щодо організації. СППР дозволяє користувачам аналізувати моделювати та інформацію таким чином, який є найбільш ефективним для прийняття конкретних рішень і підтримуватиметься в інтерактивному режимі [33].

Система СППР базується на застосуванні математичних методів прийняття рішень, математичних моделях, технологіях та програмних засобах. Тому визначення системи підтримки прийняття рішень часто асоціюється з цими методами, моделями та інструментами.

На даний момент, єдиного визначення СППР не існує. Однак, існує декілька поширених визначень СППР, які відображають основні особливості побудови, використання та ефективності застосування цих систем [6].

Відповідно до Turban [4], СППР має такі чотири основні характеристики:

1. СППР використовує і моделі, і дані;
2. СППР підтримують, а не замінюють процес прийняття рішення розробників/менеджерів;
3. СППР призначені для допомоги менеджерам/розробникам під час прийняття рішень для неструктурованих та слабко структурованих задач;

4. Мета СППР – підняття ефективності рішень.

Turban [4] запропонував перелік характеристик ідеальної СПР, яка має такі характеристики:

– оперує зі слабкоструктурованими рішеннями;

– призначена для ЛПР різного рівня;

– може бути пристосована для індивідуального та групового використання;

– підтримує як послідовні, так і взаємозалежні рішення;

– підтримує три фази процесу рішення: інтелектуальну частину, проектування та вибір;

– підтримує різноманітні стилі та методи рішення, що може бути корисно під час розв'язання задачі групою ЛПР;

– є гнучкою і адаптується як до змін свого середовища, так і організації;

– підвищує ефективність процесу прийняття рішень;

– проста у використанні та модифікації;

– дозволяє людині керувати процесом прийняття рішень завдяки комплексу технічних засобів;

– якщо може бути сформульована логіка конструкції СППР, то можливо легко побудувати СПР;

– підтримує еволюційне використання і легко адаптується до вимог, що змінюються;

– підтримує моделювання;

– позволяє використовувати знання.

Задачі, що вирішуються СППР наведені на наступній таблиці.

Таблиця 1.2 – Задачі СППР

Задача	Опис
Збір інформації	Збір базової інформації про проблему
Розпізнавання проблеми	Виявлення проблеми згідно зібраної інформації
Формулювання проблеми/задачі	Приведення задачі до формалізованого виду
Аналіз проблеми/задачі	Виявлення проблеми/задачі з додатковими даними
Пошук варіантів рішень	Пошук рішення для проблеми/задачі з додатковими даними
Вибір рішення	Вибір оптимального рішення для проблеми/задачі
Прийняття рішення	Процес прийняття рішення для проблеми/задачі

Прийняття рішень – представляє собою процес вибору найбільш сприятливого рішення з набору прийнятних рішень або ранжування набору рішень [6]. Рішення можуть прийматися на основі знань про алгоритм, процеси, що відбуваються в алгоритмі і, можливо, протягом часу, і за наявності набору показників, що характеризують ефективність і якість рішення. Тобто потрібна відповідна об'єктна модель і модель прийняття та оцінки прийнятих рішень. Моделі рішень – це формальні представлення завдань і процесів прийняття рішень.

Будь-який процес прийняття рішення проходить у кілька основних етапів [6].

Встановлення стадії проблеми. Він складається з етапу аналізу та діагностики проблеми та етапу визначення цілей вирішення. На цьому етапі ідентифікується та описується проблемна ситуація, збирається відповідна

інформація та дані, встановлюються цілі рішень, які необхідно прийняти, що дозволяє задати напрямок пошуку рішень і усунути ті, що не відповідають цілі.

Етап прийняття рішення. Ця фаза включає етапи розробки обмежень і критеріїв прийняття рішення та визначення альтернатив рішення. Також на цьому етапі визначаються обмеження, які дозволяють відокремити прийнятні варіанти від неприйнятних, а також критерії, які допомагають вибрати найкращі варіанти рішення. Потім формується набір прийнятних альтернатив, включаючи пошук і розробку альтернативних рішень.

Фаза прийняття рішення. Він складається з оцінки альтернатив і етапу вибору остаточного рішення. На цій заключній стадії варіанти оцінюються з набору прийнятних альтернатив на основі вибраних критеріїв і подальшого остаточного вибору рішення. Цінність альтернатив, звичайно, неоднакова, але можуть виникнути певні труднощі, коли один варіант неявно має перевагу над іншим.

Таблиця 1.3 – Класифікація видів рішень

Ознака	Вид рішення		
Ступінь структурованості Проблеми	Гарно структуроване	Погано структуроване	Не структуроване
Кількість етапів реалізації рішення	Статичне (один етап)		Динамічне
Рівень інформованості про стан проблеми	Умови визначеності	Умови ризику	Умови невизначеності
Кількість ОПР	Одна особа		Багато осіб
Зміст рішення	Стратегічне		Тактичне

СППР можна представити у вигляді процесів [5]. Процес функціонування СППР представлено на рисунку 1.4.

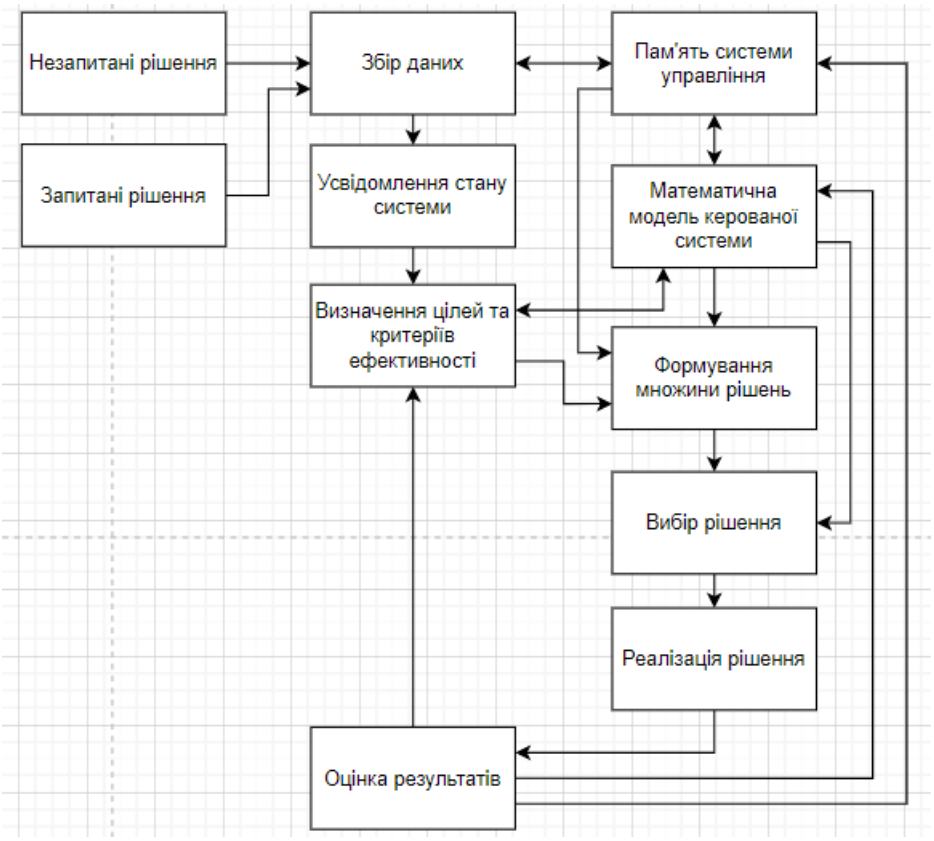

Рисунок 1.2 – Процеси системи прийняття рішень

1.3 Дослідження знання орієнтованих методів підтримки прийняття рішень

В основному виділяють три підходи до підтримки рішень:

– підхід, заснований на інформації;

– підхід, заснований на знаннях;

– підхід, заснований на інструментах.

Підхід, заснований на інформації фокусується на вдосконаленні характеру діяльності керівників підприємства, а не на наданні необхідної інформації у конкретний момент часу, за допомогою інформаційних технологій.

Підхід, заснований на знаннях виконує задачі, що показані на рисунку 1.3 даної роботи за допомогою знань, які попередньо сформовані та формалізовані. Дані знання зберігаються у БД. Також ці системи можуть використовувати моделі штучного інтелекту для побудови рішення.

Підхід, заснований на інструментах базується на інструментах, що дозволяють створити власну СППР для вирішення конкретної задачі. В основному використовуються такі інструменти:

– інструментарій СППР;

– генератори СППР;

– спеціалізовані СППР.

Доцільно розглянути модель СППР, яка ґрунтується на знаннях.

Відмінною особливістю СППР, що ґрунтуються на знаннях, є виділення нового аспекту підтримки рішень – спроможність “розуміти” проблему. Тобто, система здатна прийняти запит користувача, зібрати відповідну інформацію і підготувати звіт.

СППР, що ґрунтується на знаннях, складається з трьох взаємодіючих частин (рисунок 1.3):

– інтерфейс користувача;

– модуль підтримки рішень;

– підсистема знань.

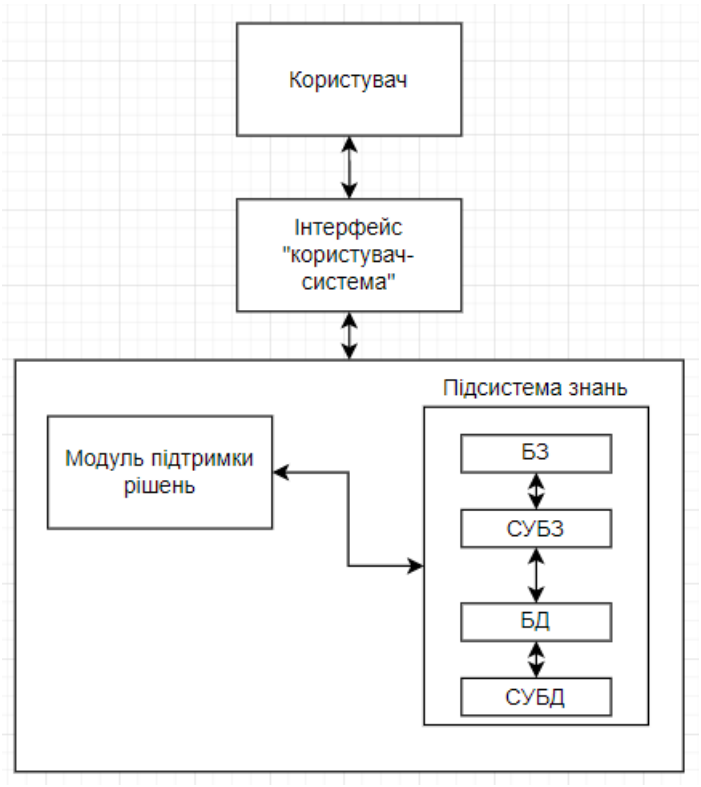

Рисунок 1.3 – Структура СППР, на основі знань

Підсистема знань вміщує інформацію стосовно предметної області. Типи даних систем відрізняються за характером уявлення в них даних і моделями формалізації знань (графи, ієрархічні структури, фрейми, семантичні мережі, тощо).

Модуль підтримки рішень відповідає за генерацію та прийняття рішень, створених на основі знань з підсистеми знань.

Темпоральні знання – знання, що враховують специфічний взаємозв’язок між станами системи у визначений часовий проміжок та враховують моменти переходу з одного стану системи до іншого. В основному використовуються у СППР для опису послідовностей явищ та їх взаємозв'язку з часовою шкалою.

1.4 Аналіз особливостей темпоральних знань

Частково структуровані задачі характеризуються якісними та кількісними залежностями, тому розв’язуються переважно з використанням індивідуальних знань з ОПР.

Автоматизація вирішення частково структурованих задач у рамках ІУС потребує формалізації залежностей та побудови бази знань, яка надасть можливість підтримувати управлінські рішення.

Якісні залежності відображають особисті знання ОПР та виконавців [10], що ускладнює їх формалізацію. Існуюча можливість виявлення залежностей базується на аналізі поведінки організаційного підрозділу та представлена у вигляді журналів подій. Цей тип аналізу може виявити часові залежності між подіями, записаними в журналі виконання операцій в алгоритмі. Таким чином, задані часові правила встановлюють переваги, коли виконуються дії, тим самим визначаючи послідовність цих дій у часі. Вищевикладене демонструє актуальність автоматичної побудови часових правил, які формують уявлення знань у ІУС із процесним та функціональним підходом до управління.

Це представлення знань має імовірнісний характер [11]. Крім того, це представлення також використовується для підтримки управлінських рішень шляхом вибору однієї з найбільш імовірних послідовностей дій у поточній ситуації [12]. Призначена послідовність формується на основі ймовірнісних виведень з бази знань, представлених набором зважених часових правил [13].

Існуючі методи опису алгоритмічної поведінки на основі аналізу журналу подій здебільшого полягають у формуванні графічної моделі процесів підприємства [14]. Ці методи в основному зосереджені на процесному підході управління та спрямовані на порівняння відомої та фактичної поведінки бізнес-процесів [15].

З іншого боку, підходи ІАД головним чином зосереджені на встановленні статичних залежностей між об’єктами предметної області [16]. Запропоновані методи [17] для вилучення залежностей між подіями та атрибутами подій дозволяють ідентифікувати контекстні залежності для виконання дій, але вони не враховують відмінності у вхідних даних – журналах подій для управління процесами та функціональними можливостями. У той же час ці відмінності мають значні наслідки для побудови ймовірнісних представлень знань.

Тимчасові правила відображають хронологічний порядок змін алгоритму. Знання про такі послідовності можуть бути представлені трьома типами модальних залежностей: NeXt, Future, Until.

Залежність типу NeXt визначає, що вказана операція слідує безпосередньо за поточною операцією. Залежність типу Future визначає майбутнє вказаної дії після поточної дії (тобто після однієї, двох, n дій). Залежність типу Until визначає майбутню операцію в результаті зміни контексту операції. Дії цього типу повинні перевіряти поточний контекст. Контекст визначається як фрагмент предметної області, який впливає на порядок операцій в алгоритмі.

Різниця між часовими залежностями NeXt, Future і Until полягає в тому, що перші два типи дозволяють враховувати лише порядок виконання дій, тоді як третій тип вимагає визначення умов, за яких вони будуть виконуватися.

Це дозволяє детально зрозуміти поведінку алгоритму, враховуючи зміни в поточному робочому стані.

Вхідними даними для побудови тимчасових правил є журнал подій. Останній містить дискретну послідовність станів ОП у вигляді послідовності подій. Кожна подія журналу – це запис окремої дії, виконаної в алгоритмі, і містить інформацію про стан артефактів, які виникли після виконання дії. Під артефактами ми розуміємо окремі неподільні об’єкти, які є частиною вільно структурованого алгоритму.

Структура журналу подій залежить від методу управління, реалізованого конкретним підприємством. За основу розглянемо процедурний і функціональний підходи. Процесний підхід засновується на визначенні горизонтальних зв’язків між виконавцями, постачальниками, користувачами продукції компанії. Цей зв'язок формалізується у вигляді бізнес-процесу. Функціональний режим реалізує вертикальну передачу інформації відповідно до організаційної ієрархії підприємства. Колективна робота виконавців може бути організована в рамках функціонального підходу. Враховуючи сучасну організаційну структуру бізнесу, інші відомі методи управління використовують поєднання горизонтальних і вертикальних інформаційних зв'язків.

Кожен журнал подій із підходом до управління процесами містить інформацію про один бізнес-процес і відповідно структрурований до його шляху. Кожна траса π_i містить інформацію про виконання бізнес-процесу у вигляді набору впорядкованих подій. Загальна діяльність підприємства з використанням процесного підходу представлена у вигляді набору бізнес-процесів. Інформація про виконання цих процесів відображається в кількох журналах подій. Таким чином, знання про поведінку бізнес-процесу можна формувати окремо для кожного бізнес-процесу у вигляді тимчасових правил на основі аналізу журналів подій кожного бізнес-процесу. Інформація про діяльність підприємства з використанням функціонального підходу управління представлена у вигляді журналів подій π_i e_1 e_2 e_j відділу або підприємства.

Кожен такий журнал зазвичай містить трасування, яке складається з подій, що фіксують виконання різних функціональних завдань. Тобто інформація про завдання та процеси підприємства «змішується» і подається у вигляді лише однієї послідовності подій. Це не дозволяє виявити знання про типові закономірності вирішення функціональних задач без фільтрації властивостей заданих подій.

Основна ідея фільтрації полягає у визначенні послідовностей подій, які відповідають життєвому циклу цільового артефакту, наприклад платіжних документів, вузлів вибору компанії тощо. Ідентифікація підмножин послідовностей подій для різних артефактів дає змогу виявити типові часові залежності між цими подіями.

Слід зазначити, що ймовірнісне представлення знань на основі логічної мережі Маркова потребує визначення ваги логічних правил. Ці вагові коефіцієнти розраховуються відповідно до методу, запропонованого в [19], який базується на ймовірності появи відповідного треку в журналі подій у разі керування процесом і враховує ймовірність виконання часу життя артефакту у випадку функціонального контролю.

На схемі, зображеній на рисунку 1.4 показано маршрути p1 і p2. Правило типу NeXt з'єднує послідовні події, наприклад, тимчасове правило типу Future визначає зв'язок між подіями e1 і e3 із проміжною подією e2 між ними. Правило часу типу Until визначає настання умови для виконання дії і тому залежить від властивостей події. Таким чином, якщо апріорно визначена різниця між підмножиною властивостей подій e1 і e2 задає умову виконання для дії, записаної в події e2, то ми маємо правило типу Until.

Тому завдання побудови правил для підтримки прийняття рішень на основі прогнозування поведінки алгоритму в поточних (у тому числі нестандартних) ситуаціях вимагає власного вирішення.

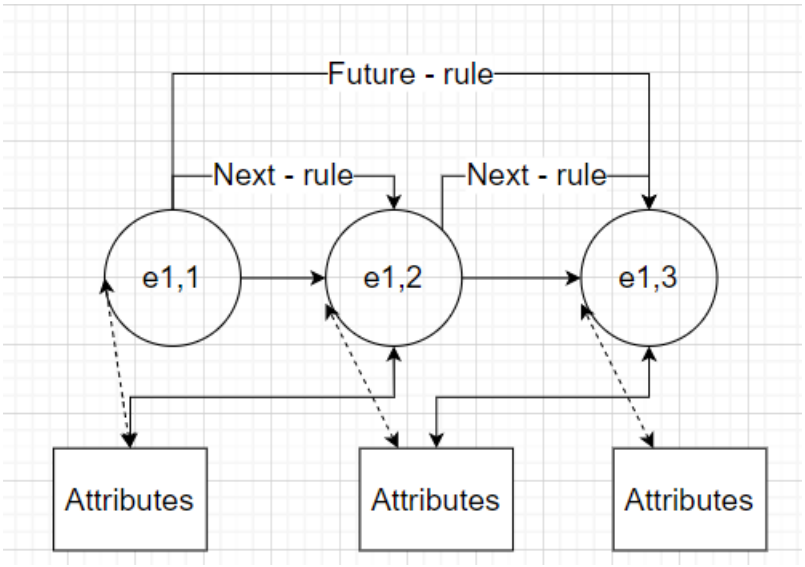

Рисунок 1.4 – Зв'язок подій та правил для процесного управління

Даний метод побудови СППР повністю підходить для формалізації процесу налагодження ARM-мережевих пристроїв у силу специфічності задачі.

1.5 Постановка задачі дослідження

Об’єкт дослідження – процеси підтримки прийняття рішень з налагодження ARM-мережних пристроїв.

Предмет дослідження – методи підтримки прийняття рішень для конфігурування ARM-мережних пристроїв з використанням темпоральних знань.

Мета дослідження – вирішення проблеми вибору процесів налагодження мережевих ARM пристроїв з використанням темпоральних знань.

Для виконання дослідження необхідно провести наступні кроки:

– сформувати методи підтримки прийняття рішень з використанням темпоральних знань;

– удосконалити методи формування з налагодження ARM-мережних пристроїв з використанням темпоральних знань;

– розробка програмного модуля (засобів) формування рішення щодо налагодження ARM-мережних пристрої з використанням темпоральних знань;

– експериментальна перевірка удосконаленого методу.

2 ФОРМУВАННЯ РІШЕНЬ ДЛЯ НАЛАГОДЖЕННЯ ARM-МЕРЕЖНИХ ПРИСТРОЇВ

2.1 Методи підтримки прийняття рішень з використанням темпоральних знань

Концептуальний підхід до автоматизованого керування часовими базами знань базується на використанні об’єктних і часових атрибутів для організації ієрархій завдань управління.

Об'єктні властивості структурованих і частково структурованих завдань зумовлюють ці особливості формування часових знань.

Перш за все, завдяки вирішенню організаційних завдань управління на оперативно-тактичному рівні інформація про керовані об’єкти та їх складові фіксується в базі даних підприємства:

– відомий стан керованого об’єкта (або його компонентів), який виникає під час виконання структурованого або частково структурованого завдання;

– час настання відомого стану алгоритму;

– управляюча дія, яка викликає фіксований стан алгоритму.

Стан об'єкта управління в цілому визначається станом набору артефактів, частиною якого він є. Стан кожного компонента алгоритму визначається як набір вибраних властивостей підмножини артефактів, що містяться в цьому компоненті. Інформація про стани з темпоральними мітками дозволяє виділити їх послідовність у часі, пов’язану, наприклад із виробництвом групи товарів або наданням комплексу послуг.

Розглянута інформація з часовими позначками, таким чином, відображає послідовний набір виконання всіх структурованих і частково

структурованих завдань, що вирішуються в рамках організаційного управління.

По-друге, інформація про часовий порядок про послідовності станів містить тимчасові залежності і, таким чином, може бути використана для визначення зважених часових правил.

Останнє являє собою знання для виконання типового комплексу управлінських дій у рамках успішної реалізації управлінських рішень. Тимчасові правила визначають такі тимчасові залежності, які є спільними для підмножини альтернативних варіантів реалізації для частково структурованих і неструктурованих завдань.

По-третє, отримані часові правила можуть враховувати рівень організаційної ієрархії, тобто загальний або детальний. Формування загального/детального правила виконується шляхом вибору відповідних підмножин артефактів і атрибутів цих артефактів. Формування правил, які ієрархічно впорядковані з різними деталями, не тільки забезпечує можливість підтримувати прийняття рішень на різних рівнях організаційної ієрархії.

Часові аспекти побудови баз знань відображають відомі властивості часу: спрямованість, лінійність, безперервність, нескінченність, однорідність. Розглянемо більш детально особливості використання цих властивостей при вирішенні автоматизованого управління тимчасовими базами знань.

Властивість спрямованості відображає перебіг процесу з минулого в майбутнє, що дає можливість задати порядок стану алгоритму в часі за допомогою часових залежностей.

Лінійність часу зазвичай представлена лінійною послідовністю дискретних моментів часу. Ця властивість визначає використання тимчасових операторів, які визначають лінійне впорядкування станів у часі.

Безперервність вказує на можливість нескінченного збільшення часових масштабів.

Нескінченність означає, що час може нескінченно тривати в напрямку майбутнього. Тимчасові залежності повинні визначати будь-яку кількість

послідовностей станів, щоб забезпечити досягнення цільового стану під час виконання управлінського рішення.

Часова однорідність означає, що будь-яка послідовність станів може бути перенесена вздовж шкали часу з минулого в майбутнє і навпаки, без зміни послідовності станів. Ця властивість дає можливість будувати часові знання без посилання на абсолютні моменти, що забезпечує часові залежності між переходами від послідовностей станів у масштабі «минуле-теперішнє-майбутнє» до «ранніх» станів. - пізніше» масштаб.

Приклад виділення темпоральних залежностей на основі станів із одного потоку часу, а також використання отриманих залежностей шляхом переходу з одного стану до іншого наведено на рисунку 2.1.

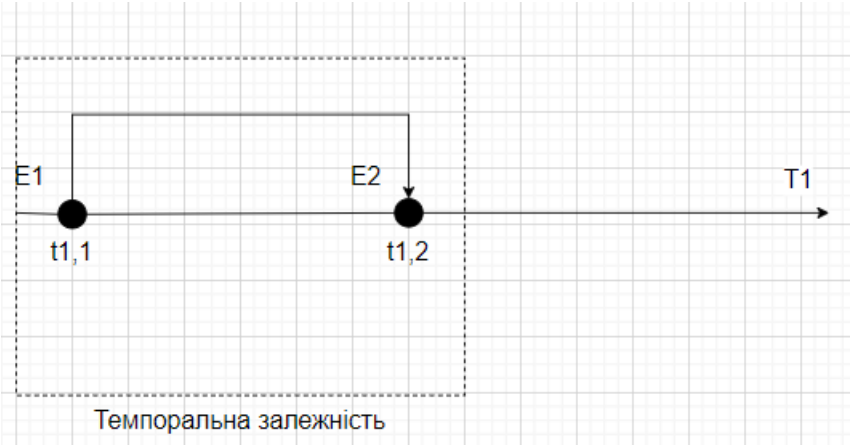

Рисунок 2.1 – Виділення темпоральних залежностей на основі станів

Послідовність станів алгоритму згідно властивостей часу спрямованості, лінійності та безперервності є строго упорядкованою. Такі властивості послідовності Пі дають можливість визначити транзитивні та антисиметричні темпоральні залежності r ∈ R для пар станів з $\boldsymbol{C_i}$:

$$\forall i \forall (e_{i,j}) \in C_i \exists r_{i,j} \qquad (2.1)$$

Темпоральний метод забезпечує формування часових правил типу NeXt і Future, які задають переваги в часі при виконанні послідовності операцій в контексті процесного та функціонального управління підприємством. Різниця в підходах пов’язана з різницею в журналі подій для керування процесами та функціями. Конструктор тимчасових правил для типів NeXt приймає такі вхідні дані. 1. У разі керування процесом – записувати події у вигляді наборів трасування {pi} та вибирати критерії трасування: час (наприклад, остання трасування бізнес-процесу); за артефактом (наприклад, трасування обробки даного документу). У випадку функціонального управління саме артефакти та їхні властивості виявляють часові залежності. Метод включає наступні фази.

Для формування знань для СППР необхідно формалізувати та побудувати базові знання для подальшого збереження у БД та використанні їх при формуванні рішень щодо налаштування ARM-мережевого пристрою. Так як був обраний темпоральний підхід при формуванні знань, необхідно детальніше розглянути фази побудування темпоральних правил. Фази побудування темпоральних правил:

Фаза 1. Створення підмножини подій для виявлення правил. У разі управління процесами маршрути призначаються за часовими критеріями або за критеріями обробки необхідних артефактів. Послідовності подій, що відповідають життєвому циклу артефакту виділяються у випадку функціонального управління. У разі функціонального управління диференціюється послідовність подій, що відповідає життєвому циклу артефакту.

Фаза 2. Визначення правил типу NeXt для кожної пари послідовних подій поточної доріжки у разі керування процесом або для кожної пари поточних подій життєвого циклу артефакту у разі функціонального керування. Крок 3. Зіставте події з різних шляхів/життєвих циклів, порівнюючи значення всіх властивостей кожної події. Умова еквівалентності події виглядає так:

Фаза 3. Зіставлення подій з різних шляхів/життєвих циклів, порівнюючи значення всіх властивостей кожної події. Умова еквівалентності події виглядає так:

$$s_{k,m} \equiv s_{1,n}: \forall k\, \beta^{r}_{k,m} = \beta^{r}_{f,r} \qquad (2.2)$$

Фаза 4. Поєднання правил C^{π}_{next} і C^{φ}_{next} за умови, що події з різних шляхів або життєвих циклів (10) еквівалентні:

Метод побудови тимчасових правил типу Future схожий на попередній метод, за винятком порядку, в якому правила будуються з пар подій:

$$C_{next\,k,m} \equiv C_{next\,1,\ x}: s_{k,m} \equiv s_{l,m} \wedge s_{k,m} \equiv s_{l,x+1} \qquad (2.3)$$

Головною відмінністю методу побудови темпоральних правил типу Future є послідовність побудови правил із пар подій. Даний метод містить у собі наступні фази.

Фаза 1. Побудова підмножини життєвих циклів (трас) для виявлення правил.

Фаза 2. Визначення пар подій виду:

$$(s_{k,m}, s_{k,x}), m = \overline{1, M} - 2, n = \overline{1, M}, x - m > 1 \qquad (2.4)$$

Фаза 3. Формування правил типу Future для кожної пари відповідно (11).

Фаза 4. Порівняння значень усіх атрибутів для кожної події для Встановлення відповідності між подіями з різних життєвих циклів (трас) (9).

Фаза 5. Поєднання правил C^{π}_{Future} та C^{φ}_{Future} , якщо подіїї з різних трас або життєвих циклів еквівалентні [10].

2.2 Удосконалення методу формування рішень з налагодження ARM-мережних пристроїв з використанням темпоральних знань

Так як існуючі методи не враховують атрибути, які надаються ОПР, а також не можуть генерувати кодову базу на основі узагальнених темпоральних правил, було вирішено додати два етапи до існуючого методу формування рішень. Дані етапи представлені під номером 3 та 5 на рисунку 2.2.

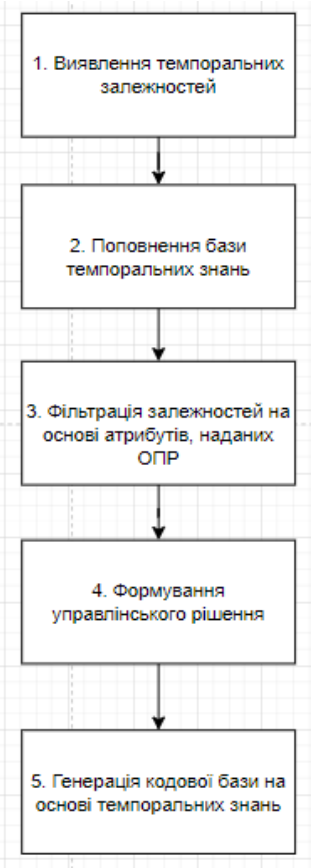

Рисунок 2.2 – Схема вдосконаленого методу методу формування рішень з налагодження ARM-мережних пристроїв з використанням темпоральних знань

Етап попередної обробки даних, представлених у вигляді множини атрибутів.

Подія представлена у вигляді:

$$e_i = \left\{ a_{i,j} \right\}, \tag{2.5}$$

де $a_{i,j}$ – атрибут події e_i.

Множина правил представлена у вигляді:

$$Rl = \left\{ r_j^i : \right\}, \tag{2.6}$$

де r_j^i – темпоральне правило.

Темпоральне правило представлене у вигляді:

$$r_j^i = e_i \overset{o}{\rightarrow} e_j, \tag{2.7}$$

де e – подія темпорального правила;

o – оператор темпорального правила (Next, Future).

Множина відібраних подій:

$$\begin{aligned} E^* &= \left\{ e_i^* \right\}, \\ e_i^* &= \left\{ a_{i,j} : \exists j : a_{i,j} \in A^{ОПР} \right\} \end{aligned} \tag{2.8}$$

Множина відібраних правил:

$$Rl = \left\{ r_j^i \mid \exists j : e_i, e_j \in E^* \right\}, \tag{2.9}$$

де r_j^i – темпоральне правило;

e – подія темпорального правила;

E* – множина відібраних подій.

Етап генерації коду відповідає за генерацію базового коду налаштування ARM-мережного пристрою на основі узагальнених темпоральних правил, що були виведені на попередньому кроці (формування управлінського рішення).

$$e_i = \left\{ a_{i,j} : \forall i \exists a_{i,j} = "operator" \right\}, \tag{2.10}$$

де $a_{i,j}$ – атрибут події e_i, хоча б один з атрибутів події e_i повинен мати значення “operator”, яке репрезентує строчку коду.

Також завдяки атрибутам, які надала ОПР, можлива додаткова модифікація сгенерованої кодової бази. Наприклад, інтеграція кодової бази, що була надана ОПР при взаємодії з СППР.

2.3 Приклад формування рішення з налагодження ARM-мережних пристроїв на основі темпоральних правил

Таблиця 2.2 – Темпоральні правила для кожного з підходів

Послідовність станів	Коментар
$\mathrm{P}_1 = \langle \mathrm{E1}, \mathrm{E2}, \mathrm{E4}, \mathrm{E5}, \mathrm{E6}, \mathrm{E7} \rangle$	P1 - налагодження ARM-мережевого пристрою для обміну повідомленнями з клієнтським ТЗ з використанням протоколу TCP

Кінець таблиці 2.2

$P_2 = \langle E1, E2, E4, E5, E6, E7 \rangle$	P2 - налагодження ARM-мережевого пристрою для обміну повідомленнями з клієнтським ТЗ через P2P бездротовий мережевий протокол Bluetooth
$P_3 = \langle E1, E2, E3, E4, E5, E6, E7 \rangle$	P3 - налагодження ARM-мережевого пристрою для обміну повідомленнями з клієнтським ТЗ через бездротову мережу Wi-Fi з використанням протоколу MQTT з інсталяцією додаткового ПЗ
$P_4 = \langle E1, E2, E3, E4, E5, E6, E7 \rangle$	P4 - налагодження ARM-мережевого пристрою для обміну повідомленнями з клієнтським ТЗ через бездротову мережу Wi-Fi з використанням протоколу AMQP з інсталяцією додаткового ПЗ

Після виведення усіх трас можливо буде формалізувати процес налагодження ARM-мережних пристроїв таким виразом:

$$P_n = E0 \xrightarrow{Next} E1, E1 \xrightarrow{Next} E2, E2 \xrightarrow{Next} E3, E3 \xrightarrow{Next} E4, E4 \xrightarrow{Next} E5, \quad (2.7)$$
$$E5 \xrightarrow{Next} E6, E6 \xrightarrow{Next} E7, E1 \xrightarrow{Future} E4$$

3 РОЗРОБКА ТЕХНОЛОГІЇ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ З ВИКОРИСТАННЯМ ТЕМПОРАЛЬНИХ ЗНАНЬ

3.1 Розробка інформаційної технології формування альтернативних варіантів рішень з налагодження ARM-мережевих пристроїв, з використанням темпоральних знань

Запропонована технологія (рисунок 3.1) інтегрує моделі представлення темпоральних залежностей, метод представлення темпоральних залежностей, метод побудови темпоральних правил, виявлення аномального стану об'єкту управління, вдосконалений метод побудови темпоральних правил для формування множини темпоральних знань щодо процесу налагодження ARM-мережевих пристроїв. Наступним етапом є генерація базового коду для налагодження ARM-мережевих пристроїв.

Процес прийняття рішення виконується в умовах невизначеності. Цей процес складається із послідовного вирішення задач пошуку та вибору й імплементації рішень.

При вирішенні першої задачі необхідно виявити поточний стан системи та формалізувати експертні знання (тобто привести їх до уніфікованого формату уявлення). Також необхідно врахувати поточні побажання розробника, що були надані у вигляді множини атрибутів на етапі взаємодії з системою.

Вибір множини рішень базується на основі усіх представлених експертних знань у системі та виявлення взаємозв’язків з наданими знаннями про поточний стан системи та наданих побажань розробника на етапі першої взаємодії з системою (які представлені у вигляді множини атрибутів). Якщо побажання не надані, то система за замовчуванням обирає усі можливі способи з налагодження ARM-мережних пристроїв.

Технологія використовує такі вхідні дані:

– експертні знання з налагодження ARM-мережевого пристрою;

– знання про поточний стан процесу налагодження ARM-мережевого пристрою;

– вхідна інформація, що надана розробником системи у вигляді множини атрибутів.

При вирішенні другої задачі використовується вдосконалений метод побудови темпоральних правил, завдяки якому формується відфільтрована множина трас для налаштування ARM-мережних пристроїв, з урахуванням атрибутів, що були надані на попередньому кроці.

Технологія на даному етапі використовує такі вхідні дані:

– масив актуальних темпоральних знань щодо процесу налагодження ARM-мережевих пристроїв з наданими атрибутами.

При вирішенні третьої задачі формується множина темпоральних знань на основі вхідної інформації, що була надана на попередньому кроці.

Технологія на даному етапі використовує такі вхідні дані:

– відфільтрована множина трас для налаштування ARM-мережних пристроїв.

Завдяки наданим атрибутам даний крок проводить рекурсивну вибірку множини, що задовольняє наданим атрибутам.

При вирішенні останньої задачі використовується метод автоматичної генерації коду на основі темпоральних правил, що були надані на попередньому кроці.

Технологія на даному етапі використовує такі вхідні дані:

– множна сформованих темпоральних знань з урахуванням вхідних інформації, наданої розробником.

Завдяки уніфікованим темпоральним знанням система в змозі автоматично згенерувати базовий код, який можливо використати при побудові системи на основі експертних знань, що були надані раніше. Якщо

множина атрибутів порожня, то система обирає перший підхід, який був виданий множиною сформованих темпоральних знань, що була сформована на попередньому кроці.

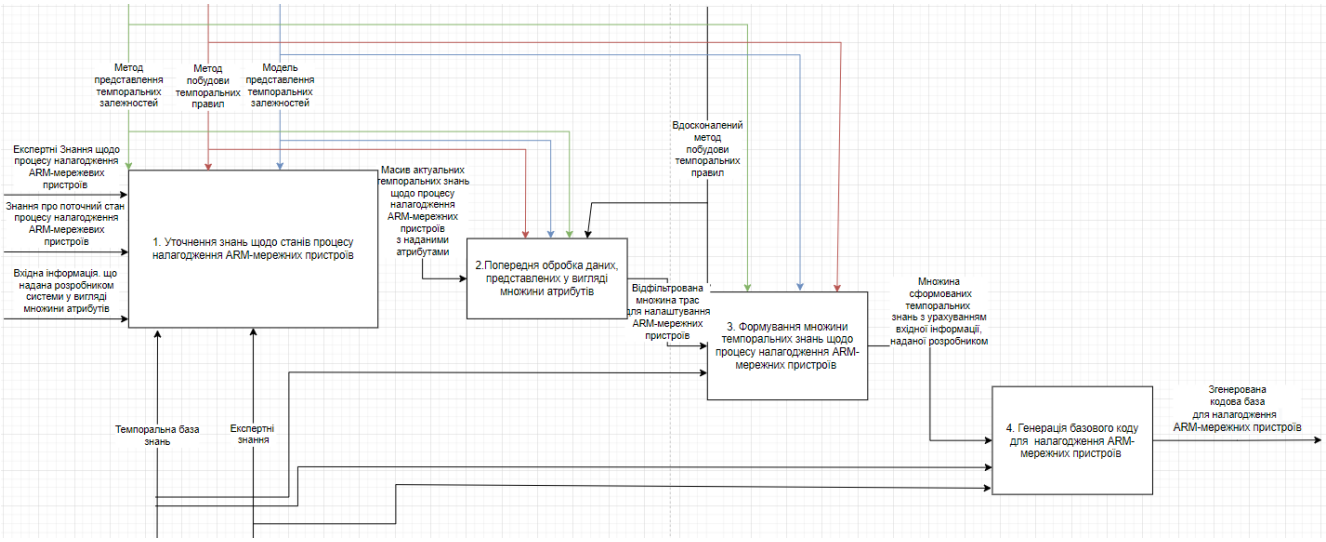

Рисунок 3.1 – Інформаційна технологія формування альтернативних варіантів рішень з налагодження ARM-мережевих пристроїв, з використанням темпоральних знань

4 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА УДОСКОНАЛЕНОГО МЕТОДУ ФОРМУВАННЯ РІШЕНЬ З НАЛАГОДЖЕННЯ ARM-МЕРЕЖНИХ ПРИСТРОЇВ

4.1 Розробка програмного модуля (засобів) формування рішення щодо налагодження ARM-мережних пристрої з використанням темпоральних знань

Вся взаємодія з розробником системи проводиться через графічний інтерфейс, наведений на рисунках 4.1, 4.2.

Далі наведено опис елементів для головної екранної форми.

Network protocol – мережевий протокол, який на основі якого буде створено кодову базу. Для поточної реалізації включає до себе такі опції для вибору:

– Not set – опцію не обрано, значення за замовчуванням;

– TCP + WiFi – опція яка включає до себе реалізацію кодової бази на основі TCP сокету з використанням бездротової мережі WiFi/дротової мережі;

– Bluetooth + RFCOMM – опція яка включає до себе реалізацію кодової бази на основі RFCOMM сокету з використанням бездротової мережі Bluetooth;

– MQTT – опція яка включає до себе реалізацію кодової бази на основі MQTT з використанням бездротової мережі WiFi/дротової мережі та встановленням додаткового ПЗ;

– AMQP – опція яка включає до себе реалізацію кодової бази на основі AMQP з використанням бездротової мережі WiFi/дротової мережі та встановленням додаткового ПЗ;

Paradigm to choose – парадигма, на якій буде засновуватися кодова база. Для поточної реалізації включає до себе такі опції для вибору:

– Not set – опцію не обрано, значення за замовчуванням;

– P2P – опція яка включає до себе реалізацію кодової бази на основі P2P мережевих протоколів (наприклад Bluetooth);

System generation level – уявляє собою індикатор для генерації повної кодової бази або часткової. На даний момент включає до себе такі варіанти вибору:

– Full code generation – значення за замовчуванням, генерація повної кодової бази;

– Partial code generation – опція що ідентифікує генерацію часткової кодової бази;

Code to integrate – код, який необхідно інтегрувати до кодової бази ARM-мережевого пристрою.

Generate code! – кнопка, після якої проходить виконання усіх 3 етапів, описаних на рисунку 3.1.

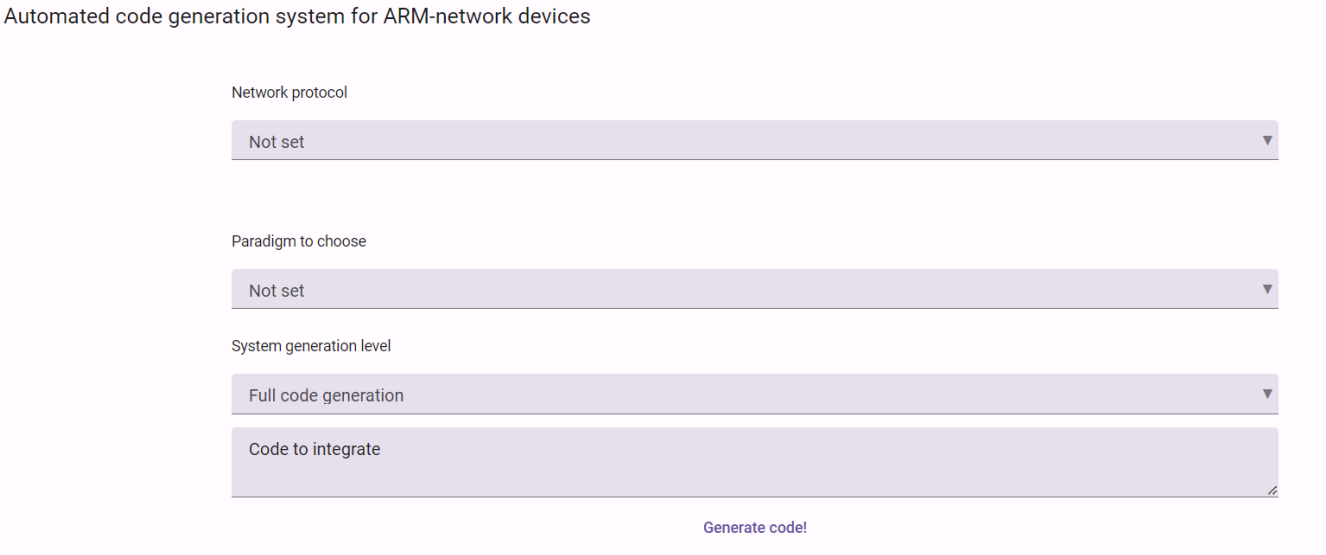

Рисунок 4.1 – Головна екранна форма

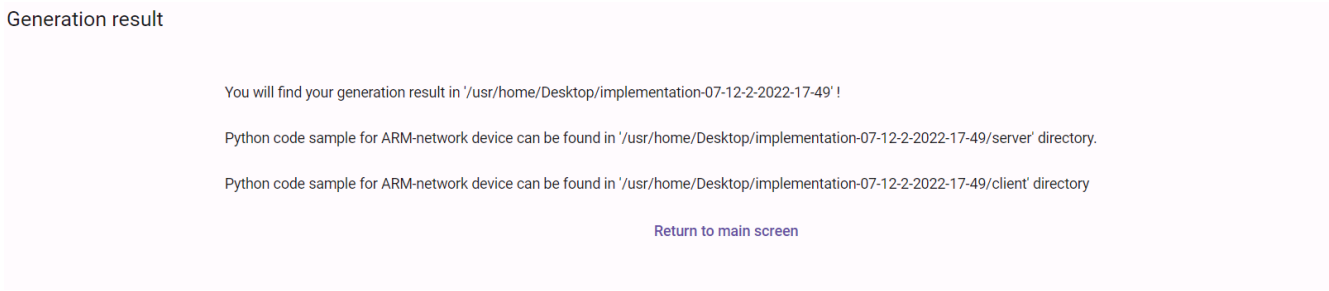

Рисунок 4.2 – Екранна форма з виведенням шляхів до сгенерованого коду

```
$ install-dependencies.sh
1   sudo apt-get update;
2   sudo apt-get install nmap;
3   sudo apt install python3.9 python3.9-dev;
4   sudo apt install apt-transport-https ca-certificates curl software-properties-common;
5   curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg;
6   echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stabl
7   sudo apt update;
8   apt-cache policy docker-ce;
9   sudo apt install docker-ce;
10  sudo systemctl status docker;
11  sudo usermod -aG docker ${USER};
12  nmap 192.168.0.0/24 10.80.0.0/24;
13  IP = ifconfig | grep -Eo 'inet (addr:)?([0-9]*\.){3}[0-9]*' | grep -Eo '([0-9]*\.){3}[0-9]*' | grep -v '127.0.0.1';
14  cd ../implementation/;
15  python3.9 -m pip install -r requirements.txt;
16  python3.9 server.py addr=$IP;
```

Рисунок 4.3 – скрипт для налаштування ARM-мережевого пристрою

```
$ connect_to_client.sh
1   sudo apt-get update
2   sudo apt install python3.9 python3.9-dev
3   IP = ifconfig | grep -Eo 'inet (addr:)?([0-9]*\.){3}[0-9]*' | grep -Eo '([0-9]*\.){3}[0-9]*' | grep -v '127.0.0.1'
4   cd ../implementation/
5   python3.9 -m pip install -r requirements.txt
6   python3.9 client.py addr=$IP
7
8
9
```

Рисунок 4.4 – скрипт для налаштування клієнтського ТЗ

mqtt_based > docker-compose.yml

```yaml
version: "3.7"

services:

  mqtt_broker:
    image: eclipse-mosquitto:2.0.15
    ports:
      - "1883:1883"
      - "9001:9001"
    volumes:
      - ./mqtt_config:/mosquitto/config/:ro
```

Рисунок 4.5 – приклад файлу опису додаткового ПЗ для клієнтського ТЗ для протоколу MQTT

amqp_based > docker-compose.yml

```yaml
version: "3.7"

services:

  rabbitmq_broker:
    image: rabbitmq:3.11-management
    ports:
      - "5672:5672"
      - "15672:15672"
```

Рисунок 4.6 – приклад файлу опису додаткового ПЗ для клієнтського ТЗ для протоколу AMQP

Для реалізації програмної частини було використано мову програмування Python та модулі, що забезпечують можливість роботи з протоколами TCP, RFCOMM, MQTT, AMQP.

Основна мова програмування, що буде використана для проведення експерименту – Python.

Часто прототипи або реальні системи IoT потрібно розробляти швидко та ефективно. Коли це відбувається, одразу постають два завдання: програмування пристроїв IoT та організація серверної частини, яка взаємодіє з цими пристроями.

В обох завданнях ви можете використовувати Python як мову розробки. Або ви можете використовувати повністю функціональну і практичну версію MicroPython для роботи на пристроях з невеликими обчислювальними ресурсами, і відповідно, за дуже низькою ціною. Давайте подивимося, як можна використовувати Python для програмування пристроїв IoT і створити серверну частину для їх роботи.

IoT з python став ефективним інструментом для створення прототипів, розробки та роботи різноманітних пристроїв і систем Інтернету речей. Python можна ефективно використовувати для програмування пристроїв IoT, а також для розробки відповідного серверу. Швидкість розробки, низький поріг для вивчення Python і великий набір бібліотек Python роблять цю мову програмування незамінною для IoT.

Також додатково будуть використані такі модулі для Python:

– Socket – модуль, що дозволяє передавати дані через протоколи UDP, TCP;

– Pybluez – модуль, що дозволяє обмінюватися даними через протокол Bluetooth та RFCOMM сокет;

– Paho-MQTT + Mosquitto broker – модуль, що дозволяє передавати дані через протокол MQTT та брокер повідомлень для MQTT протоколу;

– Pika + RabbitMQ – модуль, що дозволяє передавати дані через протокол AMQP та брокер повідомлень для AMQP протоколу;

В даній реалізації будуть використовуватися тільки бездротові протоколи, адже вони вимагають менше зусиль з налаштування для

розробника. Протоколи, що будуть використовуватися для передачі даних між клієнтським ТЗ та ARM-мережевим пристроєм:

– Wi-Fi + TCP – Wi-Fi дозволяє локальним мережам працювати без кабелів і проводів, що робить його популярним вибором для домашніх і бізнес-мереж. Wi-Fi також можна використовувати для забезпечення бездротового широкосмугового доступу до Інтернету для багатьох сучасних пристроїв, таких як ноутбуки, смартфони, планшетні комп'ютери та електронні ігрові консолі.;

– Bluetooth + RFCOMM – дозволяє пристроям спілкуватися один з одним без кабелів або проводів. Bluetooth покладається на радіочастоти малого радіусу дії, і будь-який пристрій, який використовує цю технологію, може спілкуватися, якщо він знаходиться на необхідній відстані;

– MQTT (v3.1.1) – це стандартний протокол обміну повідомленнями для Інтернету речей (IoT). Він розроблений як надзвичайно легкий транспорт для публікації/підписки, який ідеально підходить для підключення віддалених пристроїв із невеликим кодом і мінімальною пропускною здатністю мережі.

– AMQP (v 0.9.1) – Всесвітньо визнаний стандарт, який працює в основному на прикладному рівні, він в основному використовується для розробки неперевершеної оперативності зв'язку між сторонами клієнта та брокера. Видавець несе відповідальність за створення повідомлень, а клієнти збирають і адмініструють їх.

Також на даному етапі будуть перераховані кроки для налаштування ПЗ та ТЗ на ARM мережному пристрої, так як вони повністю включать до себе пункти з налаштування комплексів ПЗ та ТЗ на клієнтському пристрої.

Кроки для налаштування ARM мережевого пристрою:

1. Підготовка базового образу ОС для заливання на накопичувач. Даний етап потрібен для запуску базового ПЗ на ARM-мережевому пристрої (він же ARM SBC);

2. Під’єднання накопичувача до ARM SBC;

3. (Опціональний крок) Встановлення додаткового ПЗ для функціонування системи. Цей крок включає до себе інсталяцію додаткового ПЗ для роботи зі специфічною протоколом на клієнтському ТЗ.

4. (Опціональний крок) Запуск додаткового ПЗ на клієнтському ТЗ. Цей крок включає до себе запуск необхідного для функціонування системи ПЗ у фоновому режимі.

5. Запуск ARM SBC. Цей крок включає до себе верифікацію успішного запуску ARM SBC. Також цей крок включає до себе базове налаштування ARM SBC через графічний інтерфейс. Додатково можливе налаштування протоколу SSH для віддаленого доступу до ARM SBC через мережу. У даному дослідженні доступ через SSH не буде налаштовуватися, так як цей крок включає до себе додаткові маніпуляції з боку користувача системи, а також необхідне під’єднання до локальної мережі (LAN) або під’єднання до LAN через безпровідний режим Wi-Fi, що може підвищити енергоспоживання ARM SBC. Дане дослідження також фокусується на мінімізації енергоспоживання, тому даний крок необхідно пропустити.

6. Перенести файли для ПЗ на ARM SBC (та клієнтське ТЗ). Це можливо зробити за допомогою зовнішнього накопичувача або через протокол SSH за допомогою утиліти SCP (secure copy). Передача даних через SSH при даному дослідженні використовуватися не буде, з причин зазначених у пункті 5.

7. Встановити залежності ПЗ для виклику віддалених процедур на ARM SBC. На цьому етапі можуть виникнути помилки при інсталяції, які потрібно буде вирішити кінцевому користувачеві.

8. Запустити ПЗ на ARM SBC. Перед цим необхідно активувати необхідні ТЗ для з’єднання з клієнтом, якщо протокол працює по принципу P2P. Також на цьому етапі можуть виникнути помилки при запуску, які потрібно буде вирішити кінцевому користувачеві.

Для пункту “Налаштування ПЗ на стороні клієнтського пристрою для комунікації з ARM-мережних пристроєм” потрібно повторити кроки, що зазначені вище, у такому порядку: 3, 4, 5, 6, 7, 8.

Для початку потрібно встановити, що клієнтське ТЗ та ARM мережевий пристрій мають вбудовані модулі Wi-Fi та Bluetooth (так як вони фактично встановлюються до більшості автономних пристроїв). Також, для даного експерименту було обрано використовувати чотири підходи до налагодження ARM-мережевого пристрою:

– налагодження ARM-мережевого пристрою для обміну повідомленнями з клієнтським ТЗ через бездротову мережу Wi-Fi з використанням протоколу TCP;

– налагодження ARM-мережевого пристрою для обміну повідомленнями з клієнтським ТЗ через P2P бездротовий мережевий протокол Bluetooth;

– налагодження ARM-мережевого пристрою для обміну повідомленнями з клієнтським ТЗ через бездротову мережу Wi-Fi з використанням протоколу MQTT з інсталяцією додаткового ПЗ;

– налагодження ARM-мережевого пристрою для обміну повідомленнями з клієнтським ТЗ через бездротову мережу Wi-Fi з використанням протоколу AMQP з інсталяцією додаткового ПЗ.

На даному етапі потрібно сформувати темпоральні правила для кожного з підходів з зазначеними умовами в пункті 2.3 даної роботи.

Для перелічених у пункті 2.2 даної роботи кроків також необхідно виділити атрибути для кожного стану системи у різні часові проміжки. У початковому стані всі атрибути системи не істинні:

– A0: ОС встановлена на накопичувач ARM SBC;

– A1: накопичувач під’єднаний до ARM SBC;

– A2: додаткове ПЗ для функціонування системи встановлено на клієнтське ТЗ;

– A3: додаткове ПЗ запущено на клієнтському ТЗ;

– A4: ARM SBC запущено;

– A5: ОС ARM SBC налаштовано;

– A6: файли для запуску системи перенесено на ARM SBC;

– A7: залежності для ПЗ системи встановлено на ARM SBC;

– A8: ПЗ для системи запущено на ARM SBC.

Також необхідно виділити стани системи для подальшої побудови темпоральних виразів. Таблиця 4.1 демонструє значення атрибутів системи для усіх можливих кроків з налаштування ARM-мережевого пристрою.

Таблиця 4.1 – Опис станів системи для виклику віддалених процедур

Умовне позначення	Назва кроку	Значення атрибутів системи
E0	Підготовка базового образу ОС для заливання на накопичувач.	A0 = true; A1 = false; A2 = false; A3 = false; A4 = false; A5 = false; A6 = false; A7 = false; A8 = false.
E1	Під’єднання накопичувача до ARM SBC.	A0 = true; A1 = true; A2 = false; A3 = false; A4 = false;

Продовження таблиці 4.1

		A5 = false; A6 = false; A7 = false; A8 = false.
E2	Встановлення додаткового ПЗ для функціонування системи.	A0 = true; A1 = true; A2 = true; A3 = false; A4 = false; A5 = false; A6 = false; A7 = false; A8 = false.
E3	Запуск додаткового ПЗ на клієнтському ТЗ.	A0 = true; A1 = true; A2 = true; A3 = true; A4 = false; A5 = false; A6 = false; A7 = false; A8 = false.
E4	Запуск ARM SBC.	A0 = true; A1 = true; A2 = true; A3 = true; A4 = true; A5 = true;

Продовження таблиці 2.1

		A6 = false; A7 = false; A8 = false.
E5	Перенести файли для ПЗ на ARM SBC (та клієнтське ТЗ).	A0 = true; A1 = true; A2 = true; A3 = true; A4 = true; A5 = true; A6 = true; A7 = false; A8 = false.
E6	Встановити залежності ПЗ для виклику віддалених процедур на ARM SBC.	A0 = true; A1 = true; A2 = true; A3 = true; A4 = true; A5 = true; A6 = true; A7 = true; A8 = false.
E7	Запустити ПЗ на ARM SBC.	A0 = true; A1 = true; A2 = true; A3 = true; A4 = true; A5 = true; A6 = true;

Кінець таблиці 4.1

		A7 = true; A8 = true

Після виведення усіх можливих станів системи з атрибутами системи можливо провести експериментальну перевірку з урахуванням атрибутів системи.

4.2 Експериментальна перевірка удосконаленого методу

Вхідні дані для експерименту можуть бути сгенеровані у вигляді тестових даних, заснованих на доступних опціях вибору, що представлені у розділі 4.1.

На мові Python:

choice(network_protocol_choices) – вибір з одного елементу з ітеруємої структури даних

Код для генерації наведено на рисунку 4.5:

```python
from random import choice

network_protocol_choices = [
    'Not set',
    'TCP + WiFi',
    'Bluetooth + RFCOMM',
    'MQTT',
    'AMQP',
]
paradigm_to_choose_choices = [
    'Not set',
    'P2P',
]
system_generation_level_choices = [
    'Full code generation',
    'Partial code generation',
]

test_data = []
for i in range(10):
    test_attribute = {
        'network_protocol': choice(network_protocol_choices),
        'paradigm_to_choose': choice(paradigm_to_choose_choices),
        'system_generation_level': choice(system_generation_level_choices),
    }
    print(test_attribute)
    test_data.append(i)
```

Рисунок 4.7 – Код для автоматичної генерації тестових даних

```
{'network_protocol': 'Not set', 'paradigm_to_choose': 'Not set', 'system_generation_level': 'Partial code generation'}
{'network_protocol': 'AMQP', 'paradigm_to_choose': 'Not set', 'system_generation_level': 'Full code generation'}
{'network_protocol': 'TCP + WiFi', 'paradigm_to_choose': 'P2P', 'system_generation_level': 'Full code generation'}
{'network_protocol': 'Bluetooth + RFCOMM', 'paradigm_to_choose': 'P2P', 'system_generation_level': 'Full code generation'}
{'network_protocol': 'TCP + WiFi', 'paradigm_to_choose': 'P2P', 'system_generation_level': 'Partial code generation'}
{'network_protocol': 'Bluetooth + RFCOMM', 'paradigm_to_choose': 'Not set', 'system_generation_level': 'Partial code generation'}
{'network_protocol': 'Not set', 'paradigm_to_choose': 'Not set', 'system_generation_level': 'Partial code generation'}
{'network_protocol': 'AMQP', 'paradigm_to_choose': 'Not set', 'system_generation_level': 'Partial code generation'}
{'network_protocol': 'Not set', 'paradigm_to_choose': 'Not set', 'system_generation_level': 'Full code generation'}
{'network_protocol': 'TCP + WiFi', 'paradigm_to_choose': 'Not set', 'system_generation_level': 'Partial code generation'}
```

Рисунок 4.8 – Множина тестових даних, сгенерованих утилітою

Експериментальна перевірка удосконаленого методу формування з налагодження ARM-мережних пристроїв з використанням темпоральних знань містить у собі такі етапи:

– уточнення знань щодо станів процесу налагодження ARM-мережних пристроїв;

– попередня обробка даних, представлених у вигляді множини атрибутів;

– формування множини темпоральних знань щодо процесу налагодження ARM-мережних пристроїв;

– генерація базового коду для налагодження ARM-мережних пристроїв.

Програмна реалізація.

Удосконалений метод формування з налагодження ARM-мережних пристроїв з використанням темпоральних знань реалізовано мовою Python в середовищі PyCharm з використанням стандартних бібліотек для роботи з даними (Pandas, Scipy, Numpy, Random, Implicit, Sklearn,), а також за допомогою бібліотеки для побудови графічного інтерфейсу Anvil. Програмно реалізовані кроки уточнення знань щодо станів процесу налагодження ARM-мережних пристроїв, попередня обробка даних, представлених у вигляді множини атрибутів, формування множини темпоральних знань щодо процесу налагодження ARM-мережних пристроїв, генерація базового коду для налагодження ARM-мережних пристроїв.

Результати експерименту.

Завдяки утиліті для генерації тестових даних були створені вибірки з множинами атрибутів для подальшого налаштування ARM-мережних пристроїв.

Оцінку помилок при конфігуруванні наведено у таблиці 1. Дані були взяти завдяки опитуванню користувачів.

Таблиця 4.1 – Порівняння відсотку помилок до/після вдосконалення методу

	Кількість ARM-пристроїв для налагодження				
	1	2	3	4	5
	Відсоток помилок користувачів				
До вдосконалення методу	70	65	60	55	50

Кінець таблиці 4.1

Після вдосконалення методу	10	10	10	10	10

Також для кожної тестової вибірки були зроблені заміри часу виконання налаштування ARM-мережевого пристрою користувачами.

Таблиця 4.2 – Порівняння виконання налаштування ARM-мережевого пристрою користувачами до/після вдосконалення методу

	Кількість ARM-пристроїв для налагодження				
	1	2	3	4	5
	Час виконання налаштування ARM-мережевого пристрою користувачами (хвилини)				
До вдосконалення методу	10	20	30	40	50
Після вдосконалення методу	5	10	15	20	25

Потрібно також розглянути на індивідуальному прикладі вихідні дані для користувача. Для прикладу можливо взяти одну з множин, що були сгенеровані у тестових вибірках. Для прикладу можна взяти вибірку даних з такими параметрами:

– 'network_protocol': 'TCP + WiFi';

– 'paradigm_to_choose': 'Not set';

– 'system_generation_level': 'Partial code generation'.

Тобто ми очікуємо, що система має сгенерувати кодову базу для ‘TCP + WiFi’ підходу.

Результат гененерації системою кодовою бази наведено на рисунках 4.9, 4.10, 4.11, 4.12, 4.13:

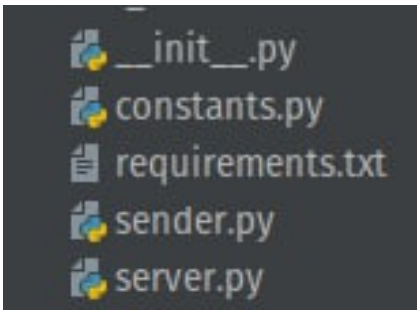

Рисунок 4.9 – Структура директорії з сгенерованим кодом

```
HOST = "192.168.31.97"  # Standard loopback interface address (localhost
LOCAL_HOST = "192.168.31.130"
PORT = 65432  # Port to listen on (non-privileged ports are > 1023)
DEFAULT_ENCODING = "utf-8"
DEFAULT_BUFFER_SIZE = 1024
```

Рисунок 4.10 – Код у файлі “constants.py”

```
import socket
import time
from json import dumps, loads

from constants import HOST, PORT, DEFAULT_ENCODING, DEFAULT_BUFFER_SIZE

start_time_of_execution = time.time()
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client:
    client.connect((HOST, PORT))
    for i in range(100):
        for num_to_calculate in range(100):
            data_to_send = {"method_path": "math.factorial", "value": num_to_calculate}
            client.sendall(dumps(data_to_send).encode(DEFAULT_ENCODING))
            data = client.recvmsg(DEFAULT_BUFFER_SIZE)
            loads(data[0])
    client.close()
    print(f"Time for execution: {time.time() - start_time_of_execution}")
```

Рисунок 4.11 – Код у файлі “sender.py”

```python
import socket
from json import loads, dumps

from constants import PORT, DEFAULT_BUFFER_SIZE, DEFAULT_ENCODING, LOCAL_HOST


def execute_method_with_data(data: bytes):
    decoded_dict = loads(data.decode(DEFAULT_ENCODING))
    method_path, value = decoded_dict["method_path"], decoded_dict["value"]
    module_name, method_name = method_path.rsplit('.', 1)
    module_to_import = __import__(module_name)
    method_to_execute = getattr(module_to_import, method_name)
    return_value = method_to_execute(value)
    return dumps({'return_value': return_value}).encode(DEFAULT_ENCODING)


with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    print(f"Starting server at port: {PORT} ...")
    s.bind((LOCAL_HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print(f"Connected by {addr}")
        while True:
            data = conn.recvmsg(DEFAULT_BUFFER_SIZE)[0]
            if not data:
                conn.close()
                break
            value_to_send = execute_method_with_data(data)
            conn.send(value_to_send)
```

Рисунок 4.12 – код у файлі “sever.py”

```
paho-mqtt==1.6.1
PyBluez==0.23
pika==1.3.1
```

Рисунок 4.13 – залежності у файлі “requirements.txt”

Система сгенерувала користувачу код, який базується на TCP сокеті та використанні бездротової мережної технології WiFi, а також згенеровано додаткові залежності для забезпечення роботи системи на ARM-мережевому пристрої та клієнтському ТЗ. Таким чином, реалізований удосконаленого

методу формування з налагодження ARM-мережних пристроїв з використанням темпоральних знань генерує релевантний код з точки зору користувача.

Результати експериментальної перевірки.

Спочатку необхідно побудувати графік залежності між кількістю ARM-мережних пристроїв для налаштування та відсотком помилок користувачів, які представлені на рисунку 4.14:

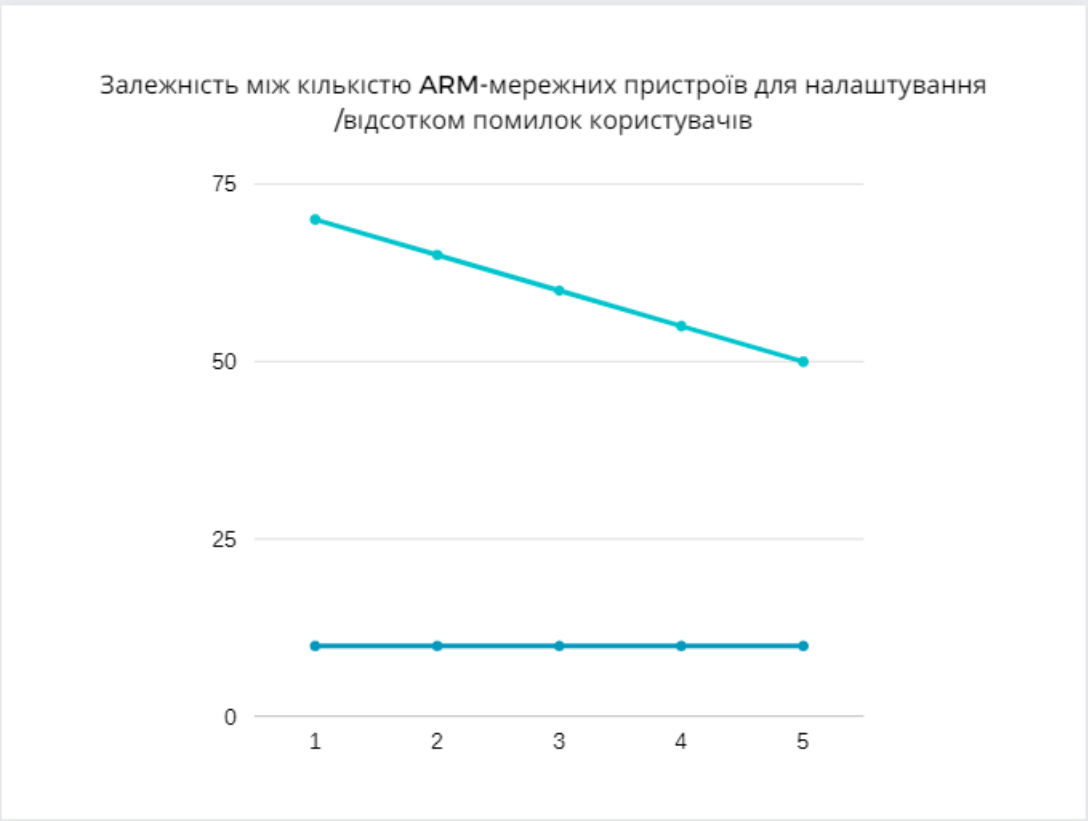

Рисунок 4.14 – графік залежності між кількістю ARM-мережних пристроїв для налаштування та відсотком помилок користувачів

Також необхідно побудувати графік залежності між кількістю ARM-мережних пристроїв для налаштування та часом виконання налаштування ARM-мережевого пристрою користувачами.

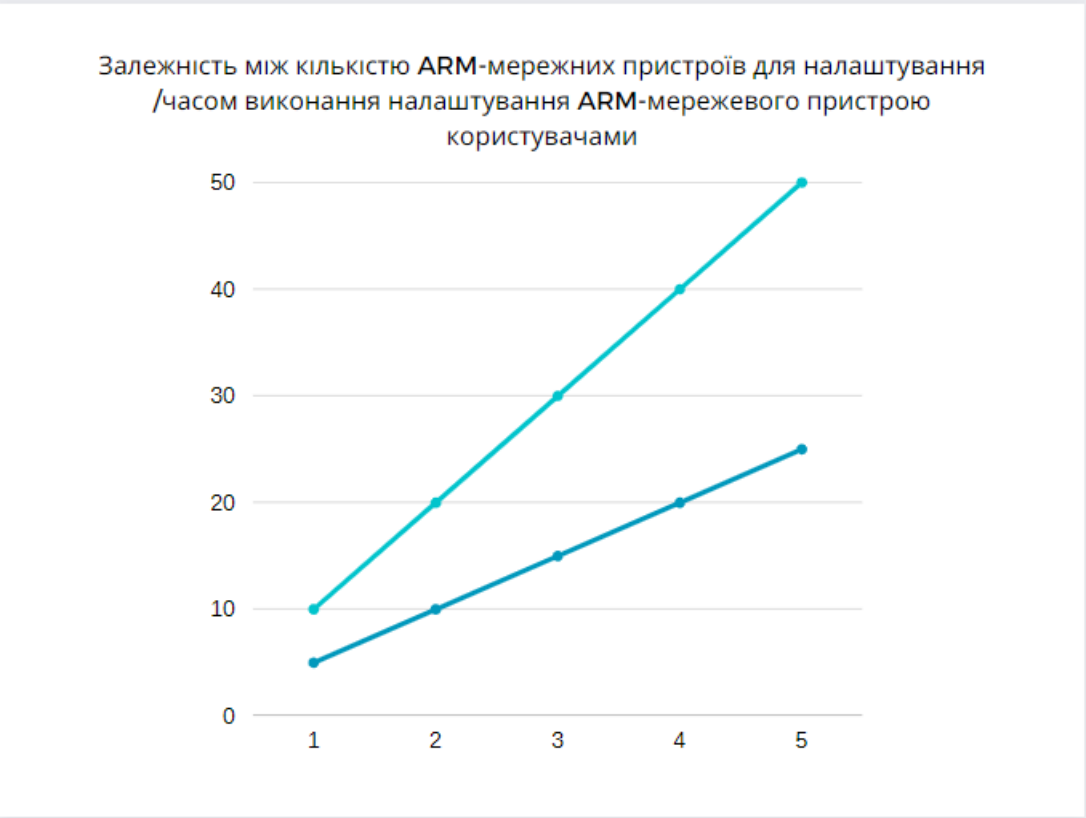

Рисунок 4.15 – графік залежності між кількістю ARM-мережних пристроїв для налаштування та часом виконання налаштування ARM-мережевого пристрою користувачами

Як можна побачити усі графікі залежностей близькі до лінійних графіків.

У середньому відсоток помилок користувачів після вдосконалення скоротилась до 10 відсотків, тож метод зменшує кількість помилок кориистувачів.

У середньому час виконання налаштування зменшився на 50 відсотків, тож удосконалений метод дозволяє пришвидшувати процесс налагодження ARM-мережевих пристроїв, у порівнянні з неудосконаленим методом.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було проведення дослідження особливостей задачі налагодження мережних ARM пристроїв, були розглянуті можливі системи підтримки прийняття рішень та їх класифікація. У роботі проведене дослідження темпоральних методів підтримки прийняття рішень, сформовано методи підтримки прийняття рішень з використанням темпоральних знань, удосконалено методи формування з налагодження ARM-мережних пристроїв з використанням темпоральних знань, розробка програмного модуля (засобів) формування рішення щодо налагодження ARM-мережних пристрої з використанням темпоральних знань, експериментальна перевірка удосконаленого методу.

Результати даної роботи можуть бути використані практично при вирішенні задач, що потребують систематизації ланцюгів дій до єдиної системи прийняття рішень, а також будуть застосовуватися у системах, що побудовані на основі ARM пристроїв, звичайних x86 пристроїв та мікроконтролерів, які мають необхідне ТЗ для роботи з розглянутих протоколів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки до передатестаційної практики для студентів усіх форм навчання спеціальності 122 – Комп'ютерні науки, освітньо-професійної програми "Інформаційні управляючі системи та технології" / Упоряд.: Чалий С.Ф., Євланов М. В., Чала О. В. - Харків: ХНУРЕ, 2021

2. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлювання. – Чинний від 22.06.2015. – Київ: ДП «УкрНДНЦ», 2016. – 31 с.

3. ДСТУ 8302:2015. Інформація та документація. Бібліографічні посилання. Загальні положення та правила складаня. – Чинний від 04.03.2016. – Київ: ДП «УкрНДНЦ», 2016. – 20 с.

4. Turban, E. Decision support and expert systems: management support systems. - Englewood Cliffs, N.J.: Prentice Hall, 1995.

5. Power D. J. A Brief History of Decision Support Systems [Electron resource]/ Power D.J. – DSSResources.COM, World Wide Web, version 2.8, May 31, 2003. – Mode of access: http://dssresources.com/history/dsshistory.html

6. Нейман Дж. Теория игр и экономическое поведение Пер. с англ. / Нейман Дж., Моргенштерн О. – Москва : Наука, 1970. – 707 с.

7. Темпоральна логіка [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BC%D0%BF%D0%BE%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D0%BB%D0%BE%D0%B3%D1%96%D0%BA%D0%B0.

8. Orange Pi Zero 2 [Електронний ресурс] – Режим доступу до ресурсу: http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/service-and-support/Orange-Pi-Zero-2.html.

9. Чала О. В. ПОБУДОВА ТЕМПОРАЛЬНИХ ПРАВИЛ ДЛЯ ПРЕДСТАВЛЕННЯ ЗНАНЬ В ІНФОРМАЦІЙНО-УПРАВЛЯЮЧИХ СИСТЕМАХ / Оксана Вікторовна Чала. – 2018. – №2. – С. 6.

10. Vom Brocke, J. (2015), Handbook *on Business Process Management 1. Introduction, Methods, and Information Systems*, Springer-Verlag Berlin Heidelberg, 709 p., available at: http://dx.doi.org/10.1007/978-3-642-45100-3.

11. Kalynychenko, O., Chalyi, S., Bodyanskiy, Y., Golian, V. and Golian, N. (2013, September), "Implementation of search mechanism for implicit dependences in process mining", *2013 IEEE 7th International Conference on Intelligent Data Acqui-sition and Advanced Computing Systems (IDAACS)*. Institute of Electrical and Electronics Engineers (IEEE), available at: https://doi.org/10.1109/idaacs.2013.6662.

12. Richardson, M. and Domingos, P. (2006), "Markov logic networks", *Machine learning*, 62(1-2), pp. 107-136, available at: http://dx.doi.org/10.1007/s10994- 006-8633-8.

13. Gogate, V. and Domingos, P. (2010), "Formula-Based Probabilistic Inference", *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, Catalina Island, CA – July 08 - 11, 2010.

14. Chala O.V. (2018), "Method of hierarchical deduction in the knowledge base of the information-control system in the para-digm "Enterprise 2.0"", *Control, navigation and communication systems*, 2018, No. 4 (50), pp. 86-90.

15. Kalenkova A.A., van der Aalst W.M.P., Lomazova I.A. and Rubin V.A. (2017), "Process Mining Using BPMN: Relating Event Logs and Process Models*"*, *Software and Systems Modeling*, 16 (4), pp. 1019-1048, available at: http://dx.doi.org/10.1007/s10270-015-0502-0.

16. Müller, D., Reichert, M. and Herbst, J. (2007), "Data-Driven

Modeling and Coordination of Large Process Structures", *On the Move to Meaningful Internet Systems 2007*: CoopIS, DOA, ODBASE, GADA, and IS, available at: http://dx.doi.org/10.1007/978-3-540-76848-7_10.

17. Polyvyanyy A., Smirnov S. and Weske M. (2015), "Business process model abstraction", *Handbook on Business Process Management*, 1, pp. 147-165, available at: http://dx.doi.org/10.1007/978-3-642-45100-3.

18. Levikin V.M. and Chala O.V. (2018), "Development of the representation of causal relationships for the knowledge base of the process management system", *Bulletin of the National Technical University "Kharkiv Polytechnic Institute". Series: Sys-tem Analysis, Management and Information Technology*, No. 21 (1297), pp. 48-53.

19. Levykin V. and Chala O. (2008), "Method of determining weights of temporal rules in Markov logic network for building knowledge base in information control system", EUREKA: Physics and Engineering, 5, pp. 29-35.

20. x86 [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/X86.

21. Wi-Fi [Електронний ресурс] – Режим доступу до ресурсу: https://www.britannica.com/technology/Wi-Fi.

22. Internet of Things with Python [Електронний ресурс] – Режим доступу до ресурсу: https://svitla.com/blog/internet-of-things-with-python.

23. Why Python is so popular? [Електронний ресурс] – Режим доступу до ресурсу: https://www.pulumi.com/why-is-python-so-popular/.

24. Wi-Fi [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Wi-Fi.

25. MQTT [Електронний ресурс] – Режим доступу до ресурсу: https://mqtt.org/.

26. Bluetooth [Електронний ресурс] – Режим доступу до ресурсу:

https://en.wikipedia.org/wiki/Bluetooth.

27. What is AMQP? [Електронний ресурс] – Режим доступу до ресурсу: https://www.wallarm.com/what/what-is-amqp

28. RPC [Електронний ресурс] – Режим доступу до ресурсу: https://www.rabbitmq.com/tutorials/tutorial-six-python.html.

29. paho-mqtt [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/paho-mqtt/.

30. PyBluez [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/pybluez/pybluez.

31. MQTT setup [Електронний ресурс] – Режим доступу до ресурсу: http://www.steves-internet-guide.com/.

32. Самойчук К. О. МЕТОДИ ТЕОРЕТИЧНИХ І ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ [Електронний ресурс] / К. О. Самойчук, В. О. Верхоланцева – Режим доступу до ресурсу: http://elib.tsatu.edu.ua/dep/mtf/ophv_12/index.html.

33. Ситник В. Ф. Системи підтримки прийняття рішень. – Київ: КНЕУ, 2004. – 614 с.

34. Левыкин В. М., Чалая О.В. Модель многоуровневого представления темпоральных знаний в задачах интеллектуального анализа процессов. *Вісник Академії митної служби України. Технічні науки.* 2015. №1 (51). С.5-12.

35. Левыкин В. М., Чалая О.В. Модель многоуровневого представления темпоральных знаний в задачах интеллектуального анализа процессов. *Вісник Академії митної служби України. Технічні науки.* 2015. №1 (51). С.5-12.

36. Чалий С.Ф., Лещинський В.О., Лещинська І.О. Доповнення вхідних даних рекомендаційної системи в ситуації циклічного холодного

старту з використанням темпоральних обмежень типу «NEXT». Системи управління, навігації та зв'язку, 2019. Вип. 4(56). С. 105-109.

37. Чалий С.Ф., Лещинський В.О., Лещинська І.О. Декларативно-темпоральний підхід до побудови пояснень в інтелектуальних інформаційних системах. Вісник Національного технічного університету "ХПІ". Сер.: Системний аналіз, управління та інформаційні технології . Харків : НТУ "ХПІ", 2020. № 2 (4). С. 51-56.

38. Чалий С.Ф., Лещинський В.О., Лещинська І.О. Інформаційна технологія побудови пояснень з урахуванням темпоральних змін у вимогах користувачів рекомендаційної системи. Системи управління, навігації та зв’язку. Збірник наукових праць, 2020, 3(61), 99-103. https://doi.org/https://doi.org/10.26906/SUNZ.2020.3.099.

39. Методичні вказівки щодо розробки та оформлення кваліфікаційної роботи (для студентів усіх форм навчання другого (магістерського) рівня програми"Інформаційні управляючі системи та технології) / Упоряд.:Петров К.Е., Левикін В.М., Чалий С.Ф., Євланов М.В., Саєнко В.І., Міхнов Д.К., Міхнова А.В., Чала О.В. - Харків: ХНУРЕ,2021.- 30с.