Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
Кафедра Комп'ютерних інтелектуальних технологій та систем

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Агентна модель IoT-системи

Виконав:
студент 2 курсу, групи КІТм-20-1
Педан М.С.
Спеціальність 123 Комп'ютерна інженерія
Тип програми освітньо-професійна
Освітня програма Комп'ютерні інтелектуальні технології
Керівник проф. Аксак Н.Г.

Допускається до захисту

Зав. кафедри КІТС _____ Руденко О.Г.
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Комп'ютерних інтелектуальних технологій і систем

Рівень вищої освіти другий (магістерський)

Спеціальність (напрям) 123 – Комп'ютерна інженерія
(код і назва)

Освітня програма Комп'ютерні інтелектуальні технології
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Педану Микиті Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Агентна модель IoT-системи

затверджена наказом по університету від "08" листопада 2021 р. № 1666Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10 грудня 2021 р.

3. Вхідні дані до роботи
Інтернет речей,
багатоагентна система
поведінка агента

4. Перелік питань, що потрібно опрацювати в роботі
1) проведення дослідження мультиагентних систем;
2) аналіз предмету дослідження;
3) обґрунтувати вибір агентної моделі для вирішення завдання;
4) описати архітектуру інтелектуальних агентів;
5) експериментальні дослідження;
6) дослідити сучасні міжнародні стандарти створення агентів і платформи МА;
7) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційні матеріали. Плакати - 11 арк. ф. А4

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Видача та узгодження теми проекту	08.11.2021	
2	Огляд стану проблеми та постановка задачі	09.11.2021 – 10.11.2021	
3	Аналіз літератури за напрямком магістерської	10.11.2021 – 14.11.2021	
	роботи		
4	Вибір методів рішення для реалізації та їх	16.11.2021 – 20.11.2021	
	обґрунтування		
5	Експериментальні дослідження	21.11.2021 – 28.11.2021	
6	Оформлення пояснювальної записки	29.11.2021 – 05.12.2021	
7	Підготовка графічного матеріалу	06.12.2021 – 10.12.2021	
8	Перевірка виконаного проекту курівником	10.12.2021	
9	Захист проекту	16.12.2021 – 17.12.2021	

Дата видачі завдання 08 листопада 2021 р.

Студент _____
(підпис)

Керівник роботи _____ Аксак Н.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 90 с., 13 рис., 1 дод., 50 джерел.

АГЕНТНА СИСТЕМА, IoT-РЕЧЕЙ, МОДЕЛЬ IoT-СИСТЕМИ, СЕРВІС-ОРІЄНТОВАНІ СИСТЕМИ, НАДАННЯ ПОСЛУГ

У магістерській науковій роботі вирішено актуальну проблему створення агентської моделі IoT-системи, яка дозволяє проектувати агентів, здатних існувати та узгоджено взаємодіяти в складних системах та знаходити місцерозташування для своєчасного надання послуг.

Об'єктом дослідження процеси побудови агентно-орієтованнних систем. Предметом дослідження: моделі та методи надання послуг в IoT-системах.

Метою кваліфікаційної роботи є розробка агентської моделі IoT-системи, яка динамічно знаходить місцерозташування при наданні спеціалізованх послуг.

Виконано системний аналіз існуючих технологій та систем. Імітаційне моделюванні показало доцільність використання запровпонованних підходів.

ABSTRACT

Master’s thesis: 90 pages, 13 figures, 1 appendices, 50 sources.

AGENT SYSTEM, IoT THINGS, IoT SYSTEM MODEL, SERVICE-ORIENTED SYSTEMS, SERVICE PROVISION

In the master's scientific work the actual problem of creating an agency model of IoT-system is solved, which allows to design agents capable of existing and coordinated interaction in complex systems and finding locations for timely provision of services.

The object of study are the processes of building agent-oriented systems. Subject of research: models and methods of providing services in IoT-systems.

The purpose of the qualification work is to develop an agency model of IoT-system, which dynamically finds a location in the provision of specialized services.

System analysis of existing technologies and systems is performed. Simulation simulations have shown the feasibility of using the proposed approaches.

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління

Кафедра Комп’ютерних інтелектуальних технологій та систем

АНОТАЦІЯ

КВАЛІФІКАЦІЙНОЇ РОБОТИ

рівень вищої освіти другий (магістерський)

Агентна модель IoT-системи

Виконав:
студент 2 курсу, групи КІТм-20-1
Педан М.С.
Спеціальність 123 Комп’ютерна інженерія
Тип програми освітньо-професійна
Освітня програма
Комп’ютерні інтелектуальні технології
Керівник проф. Аксак Н.Г.

2021 р.

Педан М.С. Агентна модель IoT-системи. – Магістерська кваліфікаційна робота.

У магістерській науковій роботі вирішено актуальну проблему створення агентської моделі IoT-системи, яка дозволяє проектувати агентів, здатних існувати та узгоджено взаємодіяти в складних системах та знаходити місцерозташування для своєчасного надання послуг.

Об'єктом дослідження процеси побудови агентно-орієтованнних систем. Предметом дослідження: моделі та методи надання послуг в IoT-системах.

Метою кваліфікаційної роботи є розробка агентської моделі IoT-системи, яка динамічно знаходить місцерозташування при наданні спеціалізованх послуг.

Для цього було проведено дослідження платформ IoT та багатоагентних архітектур. Виконано системний аналіз існуючих технологій та систем. Імітаційне6 моделюванні показало доцільність використання запровпонованних підходів.

Актуальність теми дослідження**.** Концепція Інтернету речей, активно розвивається в останні роки, призначена для вирішення великомасштабних завдань, які втягують в процес вирішення величезна кількість об'єктів фізичного, соціального та віртуального світу. Ці об'єкти можуть бути джерелами вельми різнорідної інформації, засобами її обробки і прийняття рішень, виконавчими органами, які впливають, в свою чергу, на об'єкти фізичного, соціального та віртуального світу. До таких об'єктів належать також хмарні ресурси і семантичні сервіси Web, доступ до яких можна отримати за допомогою комунікаційного середовища Інтернет. В даний час йде активний пошук концепцій і інтелектуальних IT, які в змозі добре справлятися із завданнями описаного класу, а вони мають велику соціальну і комерційну цінність.

В магістерській роботі досліджено одна з таких інформаційних технологій, яка була запропонована зовсім недавно. Вона пропонує використовувати парадигму, названу Інтернет агентів. В роботі подано коротку характеристику істоти цієї концепції, абстрактна модель задач, для концептуалізації та програмної реалізації яких найкращим чином підходить IT IoA. Ця постановка задачі проілюстрована конкретними прикладами додатків. В роботі показано, що

додаток IoT, формалізоване в моделі IoA, зводить задачу до дослідження великомасштабного об'єкта мережевої структури, поведінка якого формується на основі безлічі локальних взаємодій простих агентів, які формують самоорганизующееся Емерджентні поведінку об'єкта мережевої структури.

У першому розділі розглянуто актуальність та проблеми IoA для додатків IoT. Можна подумати, що MAC-технологія, що має вже більш ніж тридцятирічну історію досліджень і розробок в області методологій та інструментальних засобів, може бути швидко і просто адаптована під клас додатків IoT. По-перше, технологія MAC, по суті, до теперішнього часу не має добре розвинених і методологічно зручних засобів розробки додатків промислового рівня. По-друге, існуючі методології та інструментальні засоби MAC розглядають зазвичай інший клас задач, ніж той, який характерний для MAC концептуалізації додатків IoT. По-третє, IoA як особлива парадигма обчислень в системах IoT має право на існування тільки в спеціальному класі додатків.

В IoT-додатках розглядаються, по суті, не окремі агенти, а складний мережевий об'єкт, що складається з великої кількості взаємодіючих простих автономних агентів, заданих в комунікаційному середовищі Internet.I це дуже важлива особливість IoA як парадигми обчислень.

Для таких систем основним завданням проектування і розробки буде не методологія і технологія розробки агентів, а створення програмно-комунікаційної інфраструктури, яка і перетворює безліч простих автономних агентів в єдиний мережевий об'єкт. Її прототипом в традиційних MAC є звичайна централізована агентська платформа, наприклад, платформа JADE, яка добре освоєна програмістами MAC [15].

Раніше для IoA-реалізації додатка IoT того типу, який розглядається в даній роботі, було показано, що безліч взаємодіючих агентів IoA, поміщених в програмно-комунікаційне середовище, званим інфраструктурою, перетворюється в мережевий об'єкт, поведінка якого визначається безліччю локальних взаємодій агентів. Функції, що реалізуються інфраструктурою, називаються сервісами. Слід відзначити, що в даній роботі розглядається варіант розподіленої інфраструктури, що підтримує відкриту архітектуру мережевого

об'єкту, в якому склад вузлів, а також топологія їх зв'язків є динамічними властивостями.

Комунікаційний сервіс. Фізичною основою цього сервісу є мережа Інтернет. Комунікаційний сервіс повинен надавати канали зв'язку для доставки повідомлень з вузла мережевого об'єкту як автономним сутностям, встановленим в самому вузлі, так і іншим вузлам мережі, для яких поточний вузол є або адресатом повідомлення, або проміжним вузлом для пошуку потрібного адресата за допомогою прийнятого протоколу маршрутизації.

Сервіс адресації повідомлень. Сервіс адресації повідомлень – це механізм пошуку адресата повідомлення у відкритій мережі. З огляду на відкритість мережі ця адресація повинна виконуватися зі спеціального протоколу, який реалізує деякий механізм пошуку маршруту доставки повідомлення в мережі з динамічною архітектурою.

У другому розділі були досліджені приклади архітектур агентів. Останнім часом все більше уваги приділяють розподіленим інтелектуальним системам, які обмінюються знаннями одна з однією, використовують спільні знання, реалізують можливості повторного використання знань. Сучасні робототехнічні системи складаються з множини досить самостійних модулів. Такий модуль, який часто є автономною частиною системи, можна розглядати як агента, що працює в межах цієї системи і який обмінюється знаннями з іншими її модулями за допомогою повідомлень. Методи координації й співпраці повинні відповідати поняттям агентно-орієнтованої методології й здійснювати спільну діяльність агентів. Ці методи повинні також брати до уваги особливості системи, що розробляється, й середовища, до якого застосовують цю систему. За дотримання цих умов координація між агентами може значно збільшити ефективність колективної дії агентів.

Багатоагентний підхід до розробки та управління IoT-системами. У сучасному світі концепцію IoT неможливо уявити без використання багатоагентних технологій. Для кожного фізичного об’єкта програмний агент приводиться у відповідність з певним ступенем інтелектуалізму, представляючи його інтереси в мережі. Застосування розподілених обчислювальних систем дозволяє делегувати складні завдання програмним системам (агентам), що, у

свою чергу, дозволяє представляти та вирішувати проблеми, які важко формалізувати. Відповідно до концепції багатоагентних систем (MAS) та технологій, агент володіє лише частиною знань із загальної проблеми, в результаті чого він здатний вирішити лише частину загального завдання. Тому, щоб вирішити складну проблему, потрібно мати безліч агентів, які взаємодіють між собою, тобто багатоагентну систему. Завдання в таких системах розподіляються між агентами відповідно до їх навичок та можливостей. Будь-який агент – це відкрита система, яка має свою поведінку. Таким чином, вважається, що агент здатний сприймати інформацію з обмеженого середовища, обробляти її на основі власних ресурсів, взаємодіяти з іншими агентами і деякий час діяти в середовищі, переслідуючи власні цілі.

У третьому розділі пропонується підхід на основі розподілених Cloud-Fog-Dew обчислень з метою забезпечення ефективного доступу до веб-порталу, що надає повний набір послуг з доступу до сервісів найбільшому числу користувачів з використанням усіляких датчиків і мобільних обладнань.

До системи віддаленого надання послуг додано сервіс, який знаходиться на Fog-сервері, який знаходить місцерозташування, що дозволяє прискорити надання послуг у разі виникнення надзвичайного стану на прикладі надання медичної допомоги. Розроблено метод знаходження місце розташування загальних пристроїв для Інтернет речей. Для цього використовується агент, який містить три основні компоненти: компонент аналізу контексту, що дозволяє агенту бути контекстуально обізнаним про поточне розташування об’єкта, статус мобільності, тип підключення до Інтернету та запитувача серед інших обчислюваних параметрів. Другим компонентом є менеджер конфіденційності, який зберігає визначені користувачами параметри конфіденційності. Третій компонент генерує локації.

Також надана організаційна модель для аналізу мультиагентних організацій зі здатністю адаптуватися до непередбачених обставин, підтримуючи надійність системи. Особливою частиною організаційної моделі є призначення ролей, тому що вони можуть вважатися двигунами організації.

У четвертому розділі проведено імітаційні експерименти**.** Через обмеження на час прийняття рішень в розподілених системи потребується

інфраструктура розподілу робіт учасників моніторингу для оптимізації використання скоординованих дій завдяки керуванню розподілених груп учасників (у часі й просторі). У користувачів є веб-камера й спеціалізовані прилади, що зчитують найбільш важливі показники здоров'я, що з'єднуються з мобільним обладнанням, яке доставляє отримані дані за допомогою Інтернету на основний сервер. Треба знайти це місце розташування. Для цього тимчасова формалізація була виражена як динамічна характеристика в TTL.

Для надійного функціонування IoT- системи організація має прикладати певні зусилля для кожного аспекту. Для постійного контролю всі аспекти були підтримані належним чином. Також були розглянуті найпоширеніші підходи до координації групової поведінки агентів.

Імітаційне моделювання підтвердило доцільність використання запропонованої моделі.

Ключові слова: стратегія поведінки агента, багатоагентна модель, інтернет речей, мультиагентна система, знаходження місцерозташування.

Список опублікованих робіт за темою магістерської роботи:

Педан М.С. Огляд методів штучного інтелекту для агентно-орієнтованого моделювання // XXV Міжнародний молодіжний форум «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ», 21-23 квітня 2021 р. м. Харків

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

BDI (Belief – Desire – Intention) – архітектура інтелектуального агента

БАС − багатоагентна система

БЗ – база знань

ГА – генетичний алгоритм

МАС – мультиагентна система

ШІ − штучний інтелект

ВСТУП

Інтернет речей (IoT) є нова IT-парадигма, яка має на меті об'єднати в рамках єдиної системи об'єкти і дані з просторів віртуальних, соціальних і фізичних сутностей, умовно званих «речами» [1]. Наприклад, вона може об'єднувати в єдину систему програмні засоби та розподілені дані, представлені моделлю хмарних обчислень, інформаційні та програмні компоненти соціальних мереж і спільноти людей, які в них сформувалися, а також сенсорні мережі, які формують безліч джерел інформації про стан об'єктів фізичного світу. Нова властивість моделі IoT полягає в тому, що вона розглядає безліч різнорідних сутностей, що володіють вбудованими обчислювальними і комунікаційними можливостями і мають доступ до різних джерел інформації, в єдиній програмній архітектурі і в єдиному семантичному інформаційному просторі, яке доповнюється багатим набором хмарних веб сервісів семантичного Web, встановленого поверх комунікаційної компоненти Інтернету. І саме ця властивість відкриває для систем IoT вже в даний час величезні перспективи.

Вона, по суті, надає людській спільноті і індустрії нову семантично багату і могутню інформаційну інтернет-технологію більш високого рівня інтелектуальності в порівнянні з окремими її компонентами, які розроблені, наприклад, стосовно концепції хмарних і грід-обчислень, соціальних мереж і в концепції інтелектуальних сенсорних мереж. Можливість формувати програми, що працюють одночасно з об'єктами соціального, віртуального і фізичного світів, є велика перевага IoT, істотно підвищує масштаб додатків, з якими ця технологія в змозі працювати, і що розширює безліч потенційних класів додатків.

Напевно, головна особливість систем IoT полягає в тому, що об'єкти віртуального, соціального і фізичного світів – це розподілені автономні сутності, а їх взаємодія в концепції IoT є основним механізмом, що керує поведінкою кожного об'єкта і формує поведінку системи в цілому. В окремому додатку індустріального рівня загальна кількість таких об'єктів може бути величезною, наприклад, обчислюватися десятками і сотнями тисяч, і навіть багато більше.

Далі, особливість систем типу IoT складається і в тому, що будь-яка її програма має мережеву структуру, централізоване управління якою неможливе. Дійсно, додаток IoT на мета-рівні є мережевий об'єкт з величезною кількістю вузлів, що функціонує на основі локальних взаємодій, для якого важко уявити навіть саму можливість централізованого управління.

Ще однією важливою особливістю систем IoT типу є те, що в мережі IoT може виконуватися паралельно багато додатків (вирішуватися багато завдань), які вимагають для свого виконання одні і ті ж ресурси і сервіси. Дуже важливо підкреслити, що в концепції IoT всі доступні ресурси є ресурсами загального користування. Іншими словами, в системі IoT жоден ресурс не має власника, і це відразу гостро ставить ключове питання про те, яким чином в системі буде вирішуватися проблема розподілу ресурсів і вирішення конфліктів.

Очевидно, що зазначені та інші особливості додатків IoT пред'являють специфічні вимоги до властивостей методів, архітектур і технологій програмування, які в змозі підтримати розробку подібних систем. Звісно ж, що з сучасних IT тільки технологія розподілених автономних програмних агентів, здатних моделювати взаємодіючі об'єкти віртуального, соціального і фізичного світів в єдиному інформаційному просторі, є практично єдиним претендентом на роль методології і технології концептуалізації і розробки додатків IoT.

Дійсно, технологія агентів визначається як технологія обчислень на основі взаємодій, і ця риса технології агентів є ключовою для IoT. У агентської моделі виконуються окремими агентами автономно (ці обчислення «невидимі» для інших модулів системи), а їх координація виконується за рахунок їх інформаційних взаємодій. Очевидно, що ці особливості технології автономних програмних агентів добре відповідають вимогам з боку завдань IoT.

Інтеграція моделі IoT з моделлю агентів і багатоагентних систем (MAC) на концептуальному рівні, на рівні архітектури та програмної реалізації програм IoT розглядається в даний час в науковому співтоваристві як новий довгостроковий тренд, а відповідний напрям досліджень і розробок вже отримало назву Інтернет агентів (англ. Internet of Agents, IoA) [1]. З цією концепцією пов'язують великі надії, перш за все в частині нових обчислювально ефективних розподілених

алгоритмічних засобів вирішення завдань з області IoT і технології їх програмної реалізації.

У даній роботі розглядається постановка завдання для IoA-додатків, даються приклади таких додатків, описуються основні особливості моделі таких IoA-додатків, пропонується архітектура основної компоненти моделі IoA, якою є розподілена програмно комунікаційна інфраструктура, що підтримує взаємодію агентів і перетворює безліч окремих агентів в мережевий об'єкт, що функціонує як єдине ціле.

1 АКТУАЛЬНІСТЬ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

Агент − це метафора, яка спочатку виникла в "Штучному інтелекті" (ШІ) [17].Його характеристики змінювалися, коли акцент робився на мультиагентних системах, а не на окремих агентах. Ключовою подією в історії мультиагентних систем стало визнання того, що агентів можна використовувати для моделювання та впровадження розподілених систем. Створення Foundation for Physical IntelligentAgents (FIPA) походить від визнання того, що мультиагентні системи є новим та перспективним підходом для впровадження та розгортання розподілених систем. Важливо визнати, що часто використовують агентів поза межами ШІ, тому що інструменти та методи, які мають призначення для одного, не обов'язково є корисними для іншого. Зокрема, в даний час мультиагентні системи мають два основних додатків за межами своєї традиційної ролі в ШІ:

1. Проектування розподілених систем з впровадженням агентно-орієнтованої програмної інженерії (AOSE) (наприклад, агенти-орієнтовані методології,агенти-орієнтовані мови програмування).

2. Моделювання та симуляція на основі агента (ABMS), про який йдеться в основному в цій роботі.

Агенти − це суб'єкти, які спостерігають за своїм оточенням та діють на нього, щоб досягти своїх цілей [1, 17]. Дві фундаментальні характеристики агентів − це автономія та ситуативність. Автономія означає, що агенти мають розумний цикл,який дає їм контроль над їх внутрішнім станом і поведінкою. Враховуючи те, що агенти розташовані, тому що вони можуть відчувати, сприймати та управляти середовищем, в якому вони працюють. Навколишнє середовище може бути фізичним або віртуальним, і це розуміється агентами з точки зору (релевантних) даних. Автономія передбачає активність, тобто здатність агента вживати заходів для досягнення своїх цілей, не вимагаючи цього.

З точки зору програмування, агент-орієнтоване програмування було введено Шохамом як "спеціалізоване об'єктно-орієнтоване програмування" [23]. Різниця між агентами та статичними об'єктами зрозуміла. Посилаючись на Wooldridge [1]: об'єкти не контролюють власну поведінку (це підсумовується добре відомим девізом "Об'єкти роблять це безкоштовно; агенти роблять це тому, що вони хочуть"), об'єкти не демонструють гнучкості у своїй поведінці, і у стандартних моделях об'єктів існує один потік керування, а агенти за своєю суттю мають багато потокові властивості.

Агентно-орієнтована парадигма також відрізняється від моделі Актора (Actor Model) [19] (і від активних об'єктів (Active Objects), значною мірою натхненних останнім). Насправді, актори не мають ні цілей, ні мети, навіть якщо їхспецифікація включає в себе процес. Агенти замість цього експлуатують свійобґрунтований цикл (як контрольний потік), можливо, разом з ключовими абстракціями віри, бажання та наміру (як логіку), з тим щоб реалізувати алгоритми,наприклад, процеси дій у своєму оточенні для здійснення їх цілі. Іншими словами,об'єкти "роблять це" безкоштовно, оскільки вони є даними, агенти - це процеси та "роблять це", оскільки вони є функціональними для їх завдань.

Навколишнє середовище, в якому розташовані агенти, не виявляє такої автономії, яка є типовою агентам, хоча вона може розвиватися, також завдяки внутрішньому процесу. Однак його діяльність не спрямована на досягнення мети, іце робить середовища більш схожими на активні об'єкти.

Біноміальне агент-середовище формалізується за допомогою моделювання підходів, таких як [20], де середовище розглядається як забезпечення "навколишніх умов для існування агентів і що опосередковує як взаємодію між агентами, так і доступ до ресурсів", і зокрема за допомогою мета-моделі агент і артефакт [23].

Система, що складається з декількох взаємодіючих агентів, називається мультиагентною системою. На цьому рівні широко визнається, що подальші абстракції стають в нагоді, як організації та взаємодії, спрямованих на сприяння

змістовному та плідному координуванню автономних та неоднорідних агентів у системі. Таким чином, агенти розташовуються не тільки в фізичному середовищі, вони також знаходяться в соціальному середовищі, де вони вступають у взаємодіюз іншими агентами та підпадають під дію правил середовища, до якого вони належать. Нормативна мультиагентна система − це "мультиагентна система разом з нормативними системами, в яких агенти, з одного боку, можуть вирішити, чи слід дотримуватися явно представлених норм, а з іншого - нормативні системи визначають, як і в якій мірі агенти можуть змінювати норми" [22]. Вплив на обґрунтований цикл агента полягає в тому, що агенти можуть обґрунтувати соціальні наслідки своїх дій.

1.1 Огляд літератури

Немає єдиного визначення слова-агента, і не існує єдиного визначення для терміна мультиагентної системи (MAC). Примітно, що основні прийняті визначення визначають спільні риси, такі як взаємодія агентів у системі: через спільне середовище, через структуровані повідомлення (онтології, протоколи взаємодії). Дійсно, MAC можна визначити з точки зору взаємодіючих сутностей, зокрема агентів. Зв'язок може відрізнятися від простих форм до складних. Проста форма зв'язку полягає в тому, що обмежується простими сигналами з фіксованимиінтерпретаціями. Такий підхід був використаний Джордом у плануванні мультиагентів, щоб уникнути конфліктів, коли план був синтезований кількома агентами. Більш складна форма зв'язку - за допомогою структури робочої області. Робоча область - загальний ресурс, зазвичай розділений на декілька областей, залежно від різних типів знань або різних рівнів абстракції у вирішенні проблем, за допомогою яких агенти можуть читати або записувати відповідну інформацію для своїх дій. Іншою формою зв'язку є повідомлення, що проходить між агентами.

Автономія є ще однією основною характеристикою агентів при визначенні MAC, також називається "самоорганізовані системи", що дозволяє їм знаходити найкраще рішення своїх проблем "без втручання". Головною особливістю, яка

досягається при розробці мультиагентних систем, є гнучкість, адже мультиагентна система може бути додана, модифікована та реконструйована, без необхідності детального переписування програми. МАС також має тенденцію запобігати розповсюдженню розломів, самостійно відновлюватися та бути відмовостійкою, в основному через нерелевантність компонентів.

Дуже важливо розрізняти Автоматичні та Автономні системи. Автоматичні системи повністю попередньо запрограмовані і діють повторно і незалежно від зовнішнього впливу або управління. Вона може бути описана як самостійна або саморегульована, і вона здатна стежити за зовнішнім заданим шляхом, компенсуючи невеликі відхилення, викликані зовнішніми розривами. Однак вона не може визначити шлях відповідно до певної мети або вибрати мету, що диктує його шлях. Враховуючи те, що автономні системи, як МАС, самостійно спрямовуються на мету, оскільки вони не вимагають зовнішнього контролю, а скоріше, вони керуються законами та стратегіями, які чітко відрізняють традиційні та мультиагентні системи. Якщо використовуються методи машинного навчання, автономні системи можуть розробляти гнучкі стратегії для себе, за допомогою яких вони вибирають свою поведінку.

Більш детально, норми є фундаментальним компонентом мультиагентних систем, які керують очікуваною поведінкою щодо конкретної ситуації. Задопомогою норм подано бажану поведінку для населення природної або штучної спільноти. Дійсно, вони, як правило, розуміються як правила, що вказують на очікувані дії, які мають бути обов'язковими, заборонними або допустимими, виходячи з певного набору фактів. За даними Холландра та Ву [20], норми були використані для позначення обмежень на поведінку [25], щоб створити рішення для проблеми макрорівні [16] і служити обов'язковим [21], регулюючі або контрольні пристрої для децентралізованих систем [28]. Найбільш поширені норми:

1. Правила, які є природними нормами, що виникають без будь-якого застосування [13]. Правила вирішують проблеми координації, коли немає

конфлікту між особистістю та колективними інтересами; наприклад, кожен відповідає бажаній поведінці. Основні норми використовуються для вирішення або полегшення проблем колективної дії, коли виникає конфлікт між особистістю та колективними інтересами [22, 25]. Наприклад, норма не забруднювати міські вулиці має важливе значення, оскільки вона вимагає від людей транспортувати їх сміття, а не розпоряджатися ним на місці, актом, який вигідний кожному.

2. Нормативні норми. Нормативні норми призначені для регулювання діяльності шляхом встановлення обов'язку чи заборони при виконанні дії.

3. Конституційні норми, які затверджуються для вироблення нових цілей норм або станів справ, наприклад, правила гри, як шахи.

4. Процедурні норми, які класифікуються як об'єктивні та суб'єктивні. Об'єктивні процесуальні норми являють собою правила, що виражають, як рішення дійсно виробляються в нормативній системі, тоді як суб'єктивні процесуальні норми являють собою інструмент для осіб, які працюють у системі, наприклад, процедури зворотного зв'язку.

Координація - ще один відмінний фактор МАС. По суті, агент існує і здійснює свою діяльність у середовищі, в який входять інші агенти. Тому координація дій між агентами має важливе значення для досягнення цілей та послідовної роботи. Координація передбачає розглянути дії інших агентів у системі при плануванні та виконанні дій одного агента. Координація дозволяє агентам досягти узгодженої поведінки всієї системи. Координація може означати співпрацю, і в цьому випадку середовище агентів працює над досягненням спільних цілей, але може також означати конкуренцію, агентам, що мають розбіжності чи навіть антагоністичні цілі. У цьому останньому випадку координація важлива, оскільки агент повинен враховувати дії інших, наприклад, конкуруючи за певний ресурс або пропонуючи той же сервіс.

Ще однією характеристикою МАС є його виняткова поведінка. Поведінка, щорозвивається в агентах, зазвичай визначається як поведінка, яка не приписується жодному окремому агенту, але є глобальним результатом координації агентів [29].Це визначення підкреслює, що поведінка, що виникає, є колективною поведінкою.Є також інші визначення. Поведінка, що виникає, − це те, що неможливо прогнозувати за допомогою аналізу на будь-якому рівні простіше, ніж системою в цілому. Вихідна поведінка, за визначенням, залишається після того, як все інше було прояснено [31]. Це визначення висвітлює складність у прогнозуванні та поясненні поведінки, що виникає. Якщо поведінка є передбачуваною та зрозумілою, то вона не буде розглядатися як непередбачувана поведінка, і підходиможуть бути розроблені для обробки поведінки. Виникнення також визначається якдію простих правил, які об'єднують для отримання складних результатів [9]. Це визначення говорить, що правила, що застосовуються до осіб, можуть бути досить простими, але колективне поведінка групи може виявитися досить складною і непередбачуваною. Дослідники розробили експерименти, щоб продемонструвати таку ситуацію. Хоча це правда, що вся поведінка походить від окремих людей, взаємодії - це те, що ускладнює розуміння речей. Похідне поведінки − це, по суті, будь-яка поведінка системи, яка не є властивістю будь-якого компонента цієї системи, і виникає через взаємодії між компонентами системи. Запозичення з біологічних моделей, таких як колонія мурашок, евакуаційна поведінка також може розглядатися як виробництво високого рівня або складної поведінки через взаємодію декількох простих правил. Деякі приклади взаємодій, що виникають: поведінка колонії бджіл, де колективний збір нектару оптимізується за допомогою танцювального руху окремих робочих бджіл; Стадність птахів не можна описати поведінкою окремих птахів; збої ринку не можна пояснити "підсумовуванням" поведінки окремих інвесторів.

1.2 Постановка задачі

Метою даної роботи є розробка агентської моделі IoT-системи, яка динамічно знаходить місцерозташування при наданні спеціалізованх послуг.

Для досягнення поставленої мети в роботі необхідно вирішити наступні завдання:

− провести дослідження мультиагентних систем;

− обґрунтувати вибір агентної моделі для вирішення завдання;

− розробити метод знаходження місце розташування загальних пристроїв для Інтернет речей;

− провести імітаційне моделювання.

2 РОЗПОДІЛЕНА ПЛАТФОРМА IOA

Раніше для IoA-реалізації додатка IoT того типу, який розглядається в даній роботі, було показано, що безліч взаємодіючих агентів IoA, поміщених в програмно-комунікаційне середовище, званим інфраструктурою, перетворюється в мережевий об'єкт, поведінка якого визначається безліччю локальних взаємодій агентів. Розглянемо ті завдання, вирішення яких повинна підтримувати інфраструктура.

Функції, що реалізуються інфраструктурою, називаються далі сервісами, які вона надає окремим агентам, а також мережевому об'єкту в цілому. Слід відзначити, що в даній роботі розглядається варіант розподіленої інфраструктури, що підтримує відкриту архітектуру мережевого об'єкту, в якому склад вузлів, а також топологія їх зв'язків є динамічними властивостями. Використання цього припущення є істотним, тому що в розглянутому класі додатків IoT фізичні об'єкти можуть бути мобільними і використовують бездротові комунікації, які мають обмежену дальність. Крім того, як можна бачити з наведених вище прикладів конкретних класів додатків IoT, нові вузли можуть приєднуватися до мережевого об'єкту динамічно, а також залишати його в будь-який момент часу. Ці властивості мережевого об'єкту якраз і характеризують його як відкриту систему.

Розглянемо базові сервіси, які повинна забезпечувати ця інфраструктура в загальному випадку системи, що реалізує парадигму IoA.

Комунікаційний сервіс. Фізичною основою цього сервісу є мережа Інтернет. Комунікаційний сервіс повинен надавати канали зв'язку для доставки повідомлень з вузла мережевого об'єкту як автономним сутностям, встановленим в самому вузлі, так і іншим вузлам мережі, для яких поточний вузол є або адресатом повідомлення, або проміжним вузлом для пошуку потрібного адресата за допомогою прийнятого протоколу маршрутизації. У разі відкритої P2P-мережі комунікаційну компоненту вузла, що забезпечує сервіс адресації повідомлень, прийнято називати «піром» (від англ. Peer –

спеціальний робочий, рівноправний). Його основне завдання − управління списком своїх контактів. Суть цього завдання полягає у веденні актуального списку доступних каналів зв'язку і сусідів по мережі, досяжних за допомогою цих каналів з урахуванням динаміки складу сусідів як за рахунок зміни складу вузлів мережі, так і за рахунок зміни її топології.

Сервіс адресації повідомлень. Сервіс адресації повідомлень − це механізм пошуку адресата повідомлення у відкритій мережі. З огляду на відкритість мережі ця адресація повинна виконуватися зі спеціального протоколу, який реалізує деякий механізм пошуку маршруту доставки повідомлення в мережі з динамічною архітектурою. Серед варіантів реалізації цього механізму в відкритих мережах найбільш поширеним способом пошуку адресата є використання сервісів білих і жовтих сторінок. Для випадку мережі з динамікою топології і складу вузлів є робочий варіант стандарту FIPA [16] для розподіленого сервісу жовтих і білих сторінок.

Він пропонує використовувати розподілену P2P-агентську платформу, що забезпечує типовий механізм P2P-взаємодії в мережі, який реалізує «повне роз'єднання зв'язку агента з «піром» від прикладного рівня, дозволяючи їх незалежну модифікацію» [16, 17]. При цьому розподілена агентська платформа реалізується як оверлейна (віртуальна) мережа, встановлена поверх комунікаційного рівня, яку представляє мережа «пірів». Аналогічно прикладний рівень, утворений вузлами мережі агентів, реалізується в цій концепції як оверлейна мережа, встановлена в свою чергу поверх оверлейної мережі екземплярів розподіленої агентської платформи. Слід зауважити, що в даний час існують і інші можливості підтримки адресації повідомлень в P2P-мережі агентів, що базуються на обміні повідомленнями, наприклад, акторні системи, різні реалізації стандарту JMS (від англ. Java Messaging System) та інші.

Підтримка відкритості мережі. Відкритість мережі забезпечується, з одного боку, великою кількістю стандартних протоколів, за допомогою яких будь-яке підприємство може увійти в мережу, оголосити про свої сервіси,

отримати інформацію про сусідів і їх сервіси, і стати таким способом рівноправним вузлом мережі. З іншого боку, відкритість мережі забезпечується стандартними протоколами, яким слідують вузли мережі, які мають намір покинути мережу. При цьому інфраструктура повинна забезпечити маршрутизацію повідомлень при модифікації топології мережі і зафіксувати зміну складу сервісів, доступних в ній. Така постановка задачі розглядається в літературі по відкритих мережах. Крім класичних варіантів реалізації відкритості мережі типу технології JXTA [18], яка є Java-реалізацією стандарту, розробленого UPnP- форумом [19], останнім часом з'явилися і інші можливості по реалізації P2P-концепції програмно-комунікаційної інфраструктури для реалізації відкритих ad-hoc мереж, але їх переваги та недоліки ще належить оцінити.

Забезпечення інформаційної сумісності вузлів. Це завдання є однією з найбільш важливих для підтримки можливості кооперації вузлів мережі при розподіленому виконанні ними загальних бізнес-процесів [20]. Прикладом такої кооперації є завдання крос-докінгу, згадане в попередньому розділі при описі двох додатків IoT. Відповідно до сучасної концепції основним методом забезпечення інформаційної сумісності вузлів є використання загальної онтології. Цей аспект агентських мереж активно досліджується в науковому співтоваристві. Він досліджується практично в кожному проекті програми Європейської комісії, зокрема, програми Horizon 2020 року, присвяченій розподіленим моделям бізнесу, колективного поводження роботів та ін. [21].

Підтримка стандартних протоколів взаємодії вузлів мережі в різних випадках використання (use cases) і конкретних сценаріях їх реалізації. Це завдання має на меті забезпечення однаковості протоколів взаємодії вузлів мережі при вирішенні змістовних задач. Прикладом цих завдань є, наприклад, завдання планування розподіленого виконання складного замовлення, що надійшло в деякий вузол мережі. Добре зарекомендували себе алгоритми, що використовують механізми самоорганізації на основі моделі аукціону, наприклад, на основі його найпростішого варіанту – протоколу контрактних

мереж [22]. Іншим прикладом протоколу взаємодії вузлів мережі може бути протокол розподіленої координації локальних розкладів роботи безлічі вузлів мережі при спільному виконанні ними будь-якої кількості бізнес-процесів. Приклад такого протоколу є в [23, 24].

2.1 Приклади архітектур агентів

Останнім часом все більше уваги приділяють розподіленим інтелектуальним системам, які обмінюються знаннями одна з однією, використовують спільні знання, реалізують можливості повторного використання знань. Сучасні робототехнічні системи складаються з множини досить самостійних модулів. Такий модуль, який часто є автономною частиною системи, можна розглядати як агента, що працює в межах цієї системи і який обмінюється знаннями з іншими її модулями за допомогою повідомлень [11].

Істотний ріст складності комп'ютерних систем висуває вимоги до вищої надійності, гнучкості, маштабованості, адаптивності й здатності системи до інтеграції. Агенти й багатоагентні системи породжують новий підхід до розвитку складних систем програмного забезпечення загалом й апаратних систем зокрема. Через такі принципи, як децентралізація, автономія, орієнтація на мету, реактивність і проактивність індивідуальних агентів та взаємодія між агентами, агентноорієнтований підхід спроможний виконати вищезгадані вимоги.

Методи координації й співпраці повинні відповідати поняттям агентно-орієнтованої методології й здійснювати спільну діяльність агентів. Ці методи повинні також брати до уваги особливості системи, що розробляється, й середовища, до якого застосовують цю систему. За дотримання цих умов координація між агентами може значно збільшити ефективність колективної дії агентів

2.2 Композиційна архітектура багатоагентній системи

Ця архітектура описана в роботі [12] і має ім'я DESIRE − «framework for DEsign and Specification of Interacting REasoning components». Вона базується на понятті композиційної архітектури, яка дозволяє «описувати складного агента в прозорій манері, а також інтегрувати міркування і дії в єдиному (декларативному) логічному середовищі». Автори припускають, що агент в процесі роботи виконує дії наступного типу:

– активно сприймає і фільтрує інформацію із зовнішнього світу;

– робить висновки з цієї інформації;

– ініціалізує і виконує комунікації з іншими агентами в інтересах кооперації;

– генерує і оновлює свої переконання, роблячи і відхиляючи додаткові припущення;

– змінює зовнішнє середовище, впливаючи на нього.

Основу компетенції агента становлять знання, які в цій архітектурі класифікуються наступним чином:

– знання про матеріальний світ;

– знання про ментальний світ самого агента;

– знання про ментальні світи інших агентів;

– знання про взаємодію з матеріальним світом;

– знання про комунікації з іншими агентами (які комунікації можливі і корисні для отримання додаткової інформації).

Також необхідно брати до уваги динаміку знань і її неповноту. Розрізняють частину структури знань, що залежить від часу (динамічний стан інформації, або базу фактів) та її інваріантну частину, яка не змінюється у всіх станах.

Головна ідея композиційної архітектури полягає в тому, щоб можна було будь-якого складного агента створити як композицію компонент-примітивів, кожна з яких описує одну з підзадач, яка повинна ними

виконуватися. Компоненти повинні з'єднуватися один з одним відповідно до визначеної семантикою зв'язку. Кожна з компонент має мати простий локальний опис і використовуэ свій набір знань. Складну поведінку, яка охоплює і міркування, і дії, може забезпечуватися (динамічної) компонентою взаємодії агентів. Аналогічним чином система в цілому може композуватись з окремих агентів. Компоненти описуються в термінах багатосортності логіки предикатів.

Приклад агента з використанням композиційної архітектури:

– його власне ментальний стан, який включає в себе переконання агента, знання агента про себе (що він знає і чого він не знає), знання про стратегії управління і т.д., компоненту, яка генерує припущення, що дозволяє заповнювати неповноту знань, і керуюча частина;

– компонента комунікації, яка пов'язує агента із зовнішнім матеріальним світом і іншими агентами. Ця компонента забезпечує зв'язок із зовнішнім світом шляхом генерації спостережень і генерації дій, і те ж саме по відношенню до інших агентів (ставить питання і отримує відповіді);

– компонента аналізу стану світу, яка містить предметні знання про матеріальний світ.

Можна бачити, що ця архітектура не структурована за рівнями і компоненти відповідають функціональним. Автори в якості переваг цієї архітектури висувають такі:

– інтеграція різних типів міркувань і дій в єдиних декларативних рамках;

– використання знань про стратегії для явного управління міркуваннями;

– гнучкість в побудові агентів різних типів;

– явні і керовані акти спостереження;

– явні і керовані акти комунікації.

Однак ця архітектура поки не реалізована в рамках будь-якої програми, автори тільки мають намір використовувати її для діагностики електричних

мереж. Взагалі кажучи, ця архітектура не здається перспективною вже хоча б з огляду на її однорівневу структуризацію. Формалізація завдання має дуже обмеженими можливостями, тому що в рамках чисто предикатної логіки невимовно більшість властивостей агента.

2.3 Багаторівнева архітектура для автономного агента

Ця архітектура розроблена для спеціального додатку автономного агента − рухливого робота [16]. На відміну від більшості інших розробок, вона розрахована на реальний додаток, а не на демонстраційний варіант. У реальному додатку агент має справу з непередбаченими подіями зовнішнього світу як в просторі, так і в часі і в присутності інших агентів.

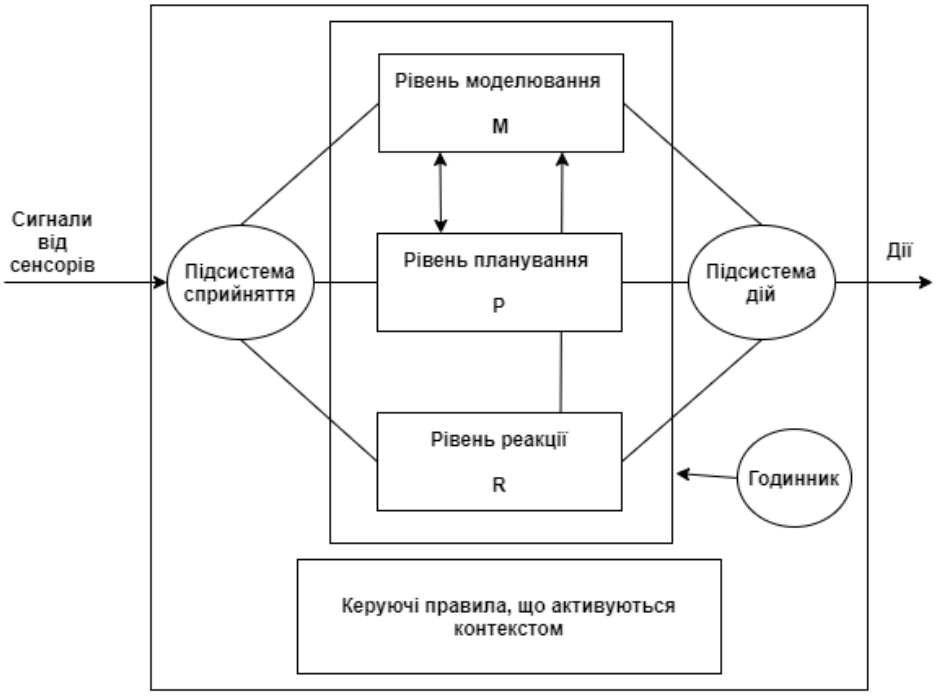

Рисунок 2.1 – Багаторівнева архітектура

При цьому він повинен бути спроможними адекватно реагувати на них і приймати рішення. Але зовнішній світ неможливо моделювати в деталях. З цієї причини архітектура агента і є, як правило, гібридною. Агент повинен мати архітектуру, яка дозволить йому справлятися з невизначеністю і

неповнотою інформації, реагувати на непередбачувані події, користуючись відносно простими правилами. Це – вихідна позиція авторів даної архітектури.

Рівень планування Р генерує, виконує і динамічно реконструює часткові плани, наприклад, для вибору маршруту рухомого робота.

Рівень передбачення, або моделювання М моделює поведінку сутностей зовнішнього середовища і самого агента, що може використовуватися для пояснення спостережуваної поведінки і передбачення можливої їх поведінки в майбутньому.

Кожен з цих рівнів має модель світу агента на відповідному рівні абстракції і містить можливості, що відповідають рівню. Кожен з рівнів безпосередньо пов'язаний з компонентою сприйняття і дії, і в стані незалежно від інших рівнів вирішувати, реагувати або не реагувати в поточному стані світу. В архітектуру включена підсистема управління на основі правил, що активується контекстом з завданням забезпечити відповідну поведінку агента в разі конфлікту варіантів поведінки, що ініціюється різними рівнями. Система реалізована як комбінація технології обміну повідомленнями та контекстної активації керуючих правил (відповідно до специфіки предметної області), яка виступає в ролі посередника, який досліджує дані різних рівнів (сприймаються входи і виходи різних рівнів), вводить на різні рівні нові дані і видаляє деякі дані.

Синхронізація входів і виходів рівнів також забезпечується цією підсистемою. Фактично правила підсистеми виступають в ролі фільтра між сенсорами агента і внутрішніми рівнями агента ("supressors") і між рівнями і їх виконавчими елементами ("censors"). Посередництво це залишається активним весь час роботи агента, однак воно "прозоре" для рівнів, кожен з яких продовжує діяти так, як якщо б він був єдиним при управлінні агентом, не піклуючись про можливий конфлікт.

Дана архітектура має реалізацію і на думку авторів цілком працездатна. Вона інтегрує в собі ряд традиційних механізмів міркувань на основі знань і механізмів чисто поведінкового, "реактивного" характеру. Вона є досить

характерним представником горизонтально організованої багаторівневої архітектури.

2.4 Багаторівнева архітектура для розподілених додатків

Ця архітектура [23] була розроблена спеціально для системи охорони здоров'я. Вона включає в себе багаторівневу структуру знань, робочу пам'ять, менеджера комунікацій і людино-машинний інтерфейс (рисунок 2.2)

Оскільки дана архітектура повинна бути релевантною медичним програмам, агент повинен володіти обома типами поведінки – як поведінкою на основі знань (наприклад, для вибору планів, декомпозиції задач, розміщення завдань), так і поведінкою на основі швидкої реакції на події (наприклад, для формування відповідей в реальному часі, на надходження нових даних, зміна наявних даних, на зміну поточних угод з іншими агентами). Таким чином, ця архітектура, як і всі раніше розглянуті, є гібридною.

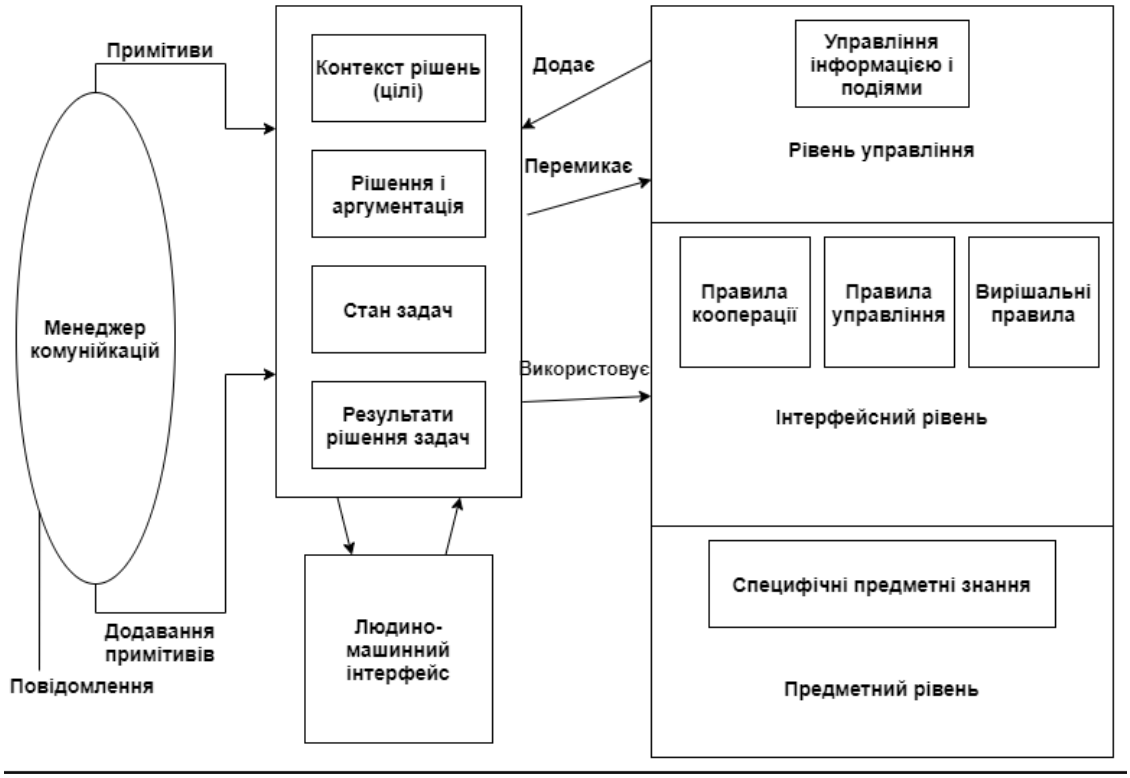

Рисунок 2.2 – Гібридна архітектура

У цій архітектурі інтелектуальна поведінка підтримується спільною роботою таких компонент, як блок вирішальних правил для обчислення плану, блок правил для управління завданнями, їх декомпозицією і розміщенням, а також блок правил для підтримки угод з іншими агентами при кооперативному вирішенні завдань. Реактивна поведінка реалізується за допомогою керуючого рівня, який реагує на зміну стану робочої пам'яті (наприклад, під час надходження нових результатів рішення задачі, цілей або повідомлень, а також при зміні наявних даних, цілей, міжагентських угод або станів завдань). Ключовим моментом даної архітектури є трирівнева організація знань, при цьому виділяються такі рівні:

Рівень специфічних предметних знань, в якому містяться медичні знання про хвороби, знання про плани управління лікуванням хвороб ("протоколи"), база даних про пацієнтів (історії хвороб) і база даних про доступні ресурси. Однак предметні знання не містять будь-якої інформації про те, як їх слід використовувати, тут представлені тільки властивості предметної області.

Рівень знань про процедури виведення; він містить декларативні правила виведення, які повинні застосовуватися до предметних знань про конкретного пацієнта, щоб вивести нові дані. Цей рівень – основний в архітектурі. У свою чергу він поділяється на компоненти прийняття рішень в умовах невизначеності, управління завданнями і управління кооперацією агентів. Наприклад, модуль управління завданнями містить декларативну схему виведення для управління переходами станів завдання. Особливості системи виведення рішень полягають у тому, що вона не використовує поняття ментального стану агента (переконання, бажання, наміри) і не використовує будь-яку логічну мову для виведення, для цього вона використовує стратегії аргументації в умовах невизначеності. Це означає, що ця архітектура не є BDI-архітектурою.

Менеджер завдань відповідальний за декомпозицію задач на

підзадачі і їх розподіл за відповідними агентам, а також за управління переходами станів завдань. Управління кооперацією агентів використовує механізм, заснований на взаємному зобов'язанні агентів ("будь-який агент згоден робити схему дій, яка має на меті виконати завдання за відповідний час"), і угоди про те, за яких умов агент має право відмовитися від своїх зобов'язань і як він повинен себе вести по відношенню до інших агентів, коли такі обставини виникнуть.

Рівень керуючих знань, який застосовує знання про процес виведення до предметних знань, щоб генерувати схему виведення, якщо в робочу пам'ять додаються нові знання.

Автори переконані, що такий функціональний розподіл знань на предметні знання, знання про процедури виведення і керуючі знання істотно спрощує їх подання, повторне використання та експлуатацію, оскільки ці компоненти можуть створюватися і підтримуватися незалежно. Крім того, ця архітектура дозволяє просто вбудовувати програми вилучення знань, кожна з компонент яких може виходити і модифікуватися незалежно один від одного.

Інші три компоненти розглянутої архітектури − це робоча пам'ять, менеджер комунікацій і людино-машинний інтерфейс.

Робоча пам'ять служить для запам'ятовування поточних даних, що генеруються рівнем управління, користувача і менеджера комунікацій. Типи інформації, яка зберігається в робочій пам'яті, такі: цілі, які повинні бути досягнуті; стан завдань, які знаходяться в поточному стані процесу виконання угод з іншими агентами. Фактично, в звичній нам термінології, робоча пам'ять є ні що інше, як дошка оголошень.

Менеджер комунікацій містить в собі повідомлення, які повинні бути надіслані іншим агентам, представлені на мові комунікацій з примітивами, схожу на примітиви мови KQML: звернутися з проханням, вжити, відкинути, змінити, запропонувати, проінформувати, зробити запит, відмовитися і підтвердити.

Людино-машинний інтерфейс визначає схему взаємодії між системою і користувачем, оскільки дана багатоагентна система не є автономною, що пов'язано з особистою відповідальністю користувача за здоров'я пацієнта.

Цю архітектура заснована на знаннях, має горизонтальну схему взаємодії рівнів. Головна її особливість в тому, що вона досить сильно орієнтована на додаток.

2.5 IDS-архітектура

Ця архітектура виникла в результаті комбінування двох напрямків досліджень. Перший з них – це логіка міркувань про дії і зміни з вихідним поняттям «населеної (живими істотами) динамічної системи» («Inhabited Dynamic System» – IDS). Другий напрямок – це побудова ефективної реалізації інтелектуальної системи.

Архітектура має трирівневу структуру і є гібридною. Вважається, що IDS-система розміщується в деякому світі (середовищі) і складається з двох базових частин – «мислячої частини» і «машини» («рухома частина об'єкта»). Автор інтерпретує поняття «мисляча частина» як інтелектуальну, засновану на знаннях частину автономного агента, його «мозок», в той час як «машина» – це тіло агента, тобто його несвідома частина, яка в порядку реакції на сприйняття і накази на виконання щось робить.

IDS сприймає зовнішнє середовище. Використовуючи процес сприйняття, система редукує і істотно узагальнює сприйняту інформацію, і посилає вихід в «мислячу частина». У свою чергу, ця частина посилає команди на свою рухому частину, яка їх відпрацьовує без будь-якого додаткового управління або зміни, викликаючи відповідні зміни в зовнішньому світі.

Ця ідея реалізується у вигляді трирівневої архітектури, представленої на рисунку 2.3. Поділ за рівнями проводиться відповідно до характеру тих обчислень, які на них виконуються. Перший рівень – це рівень процесів, на якому періодично виконуються із заданою частотою деякі обчислення, а також

здійснюється управління процесами сприйняття і виконання. Другий рівень, званий рівнем відповідної реакції, обчислює відповідну реакцію на асинхронні події, які або сприймаються рівнем процесів, або їм генеруються.

Рівень аналізу виконує символічні міркування, такі, як прогноз, планування та перепланування, а також є тим місцем, де розташовується компонента навчання агента. Дана архітектура є типовим представником багаторівневої архітектури, яка дещо близька до архітектури «Touring Machine» і відрізняється від неї варіантом розподілу завдань за рівнями.

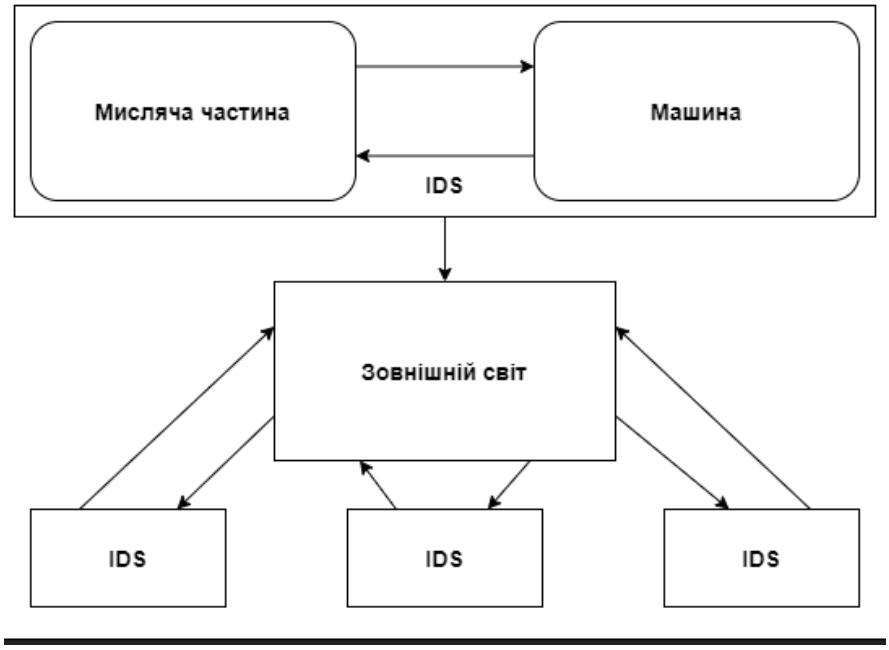

Рисунок 2.3 – Архітектура IDS

Переваги архітектури, на думку автора, слід розглядати в трьох аспектах: в ній має місце явне розділення завдань, які вимагають різних концептуальних і обчислювальних рамок; вона дозволяє при проектуванні використовувати різні інструментальні засоби (мови, алгоритми) для спрощення розробки; вона дозволяє підтримувати процес проектування простими програмними інструментальними засобами, забезпечуючи простоту процесу прототипування.

2.6 WILL-архітектура

Ця архітектура інтенсивно використовує метафори і поняття, що традиційно застосовуються до опису людської інтелектуальної діяльності, що робить її привабливою і зрозумілою, але від цього вона не стає в чомусь принципово новою по відношенню до інших архітектур, а, як видається, тільки віддаляє можливість її практичної реалізації. Однак автори стверджують, що це найбільш проста архітектура автономного агента. Слід, однак, мати на увазі, що це архітектура розрахована на одного агента, який має одну мету і його функціонування спрямовується його власними мотивами, які називаються інтересами («concerns»). Питання про методи кооперації і комунікації агентів такої архітектури залишається без уваги.

Для того, щоб агент функціонував в світі раціонально, йому необхідні різні функції, включаючи сприйняття. Припускається, що агент має для кожної з цих функцій окремий модуль. Зокрема, агент має сенсорний блок, планувальник і виконавчий пристрій в якості базових модулів, які якимось чином повинні бути інтегровані.

Головною проблемою при цьому є питання про те, як організувати спільну узгоджену роботу цих модулів, зокрема, узгодити взаємодію потоків інформації і потоків управління. Щоб вирішити проблему узгодженої взаємодії потоків інформації, вони пропонують застосувати щось на зразок схеми «бродкастінг», коли з'єднані всі входи і всі виходи модулів між собою, так що будь-яке повідомлення, що генерується тим чи іншим блоком стає доступним будь-якому іншому блоку. Всі ці повідомлення збираються в глобальному буфері, який називається пам'яттю. Всі блоки можуть читати інформацію з пам'яті, крім сенсорів, і всі вони можуть писати інформацію в пам'ять, крім виконавчого пристрою. Кожен модуль може просто брати інформацію з пам'яті, коли йому це потрібно.

Автори цієї архітектури вважають, що цілі системи можуть змінюватися і генеруватися «зсередини» агента, будучи зумовленими якимись фундаментальними цілями агента.

Вони визначаються як якісь переваги агента перебувати в якихось станах і якихось станів уникати. Коли агент отримує інформацію, яка відповідно до його інтересів відповідає кращому стану (скажімо, температура середовища дорівнює 20 градусів), то генерується внутрішній сигнал про те, що бажано, щоб в цьому стан середовища залишалася і в майбутньому. Для кожного стану зовнішнього середовища агент повинен вміти оцінювати міру її релевантності своїм інтересам. Це означає, що коли якийсь модуль звертається до пам'яті, він «бачить» той її фрагмент, який має «найбільший збіг» і обробляє цей фрагмент.

Автори стверджують, що головне нововведення цієї архітектури в наявності блоку пам'яті і використанні поняття інтересів, однак модуль пам'ять по суті близький до того, що ми звикли називати дошкою оголошень, а поняття інтереси за змістом близько до відомого в теорії агентів поняттю бажання агента. З іншого боку, автори не аналізують складність проблеми організації узгодженої роботи різних модулів агента в цій архітектурі, яка по суті може бути реалізована тільки при високому рівні самоорганізації системи, алгоритми якої можуть виявитися найтоншим місцем при спробі реалізації.

2.7 InteRRaP-архітектура

Основна ідея цієї архітектури [14] в тому, щоб представити агента як безліч рівнів, які пов'язані через керуючу структуру і використовують загальну базу знань. Ця архітектура представлена на рис.14. Вона складається з п'яти основних частин:

- інтерфейс з зовнішнім світом;
- компонента, заснована на поведінці;

– плануюча компонента;
– компонента, відповідальна за кооперацію з іншими агентами;
– база знань агента.

Інтерфейс із зовнішнім світом містить можливості агента по сприйняттю подій зовнішнього світу, впливу на нього і засоби комунікації.

Компонента, відповідальна за реактивну поведінку, використовує базові можливості агента по реактивній поведінці, а також частково використовує знання агента процедурного характеру. Вона базується на понятті «фрагмента поведінки» як деякої заготовки реакції агента на деякі стандартні ситуації. Це дозволяє агенту в стандартних ситуаціях не звертатися до планування на основі знань і реалізовувати значну частину своєї поведінки рутинним чином з гарною ефективністю. З бази знань їй доступні тільки знання нижнього рівня абстракції, де міститься інформація про фрагменти поведінки.

Компонента, відповідальна за планування, містить механізм планування, що дозволяє будувати локальні плани агента, тобто плани, які пов'язані з кооперативним поведінкою. План представляється у вигляді графа, вузлами якого можуть бути або конкретні набори дій аж до елементарних кроків поведінки, або нові субплани, які підлягають подальшій конкретизації. Таким чином, плануюча компонента активує поведінку (через нижчележачу компоненту), що керується цілями. Вона ж бере участь і в плануванні, пов'язаному з кооперативною поведінкою агентів. Ця компонента може використовувати знання двох нижніх рівнів абстракції.

Компонента, відповідальна за кооперацію агентів, бере участь в конструюванні планів спільної поведінки агентів для досягнення деяких спільних цілей або виконання своїх зобов'язань перед іншими агентами, а також виконання угод. Цій компоненті доступні знання всіх трьох рівнів абстракції.

База знань агента має трирівневу структуру і побудована за принципом дошки оголошень. Рівні бази знань фактично відповідають рівням абстракції знань відповідно до структури керуючих компонент. Модель світу агента

містить переконання агента відповідно до рівня, орієнтованими на поведінку. Другий рівень відповідає моделі ментальних знань агента і знань про поточний ментальний стан агента (наміри, цілі, плани). Нарешті, третій рівень містить знання і переконання агента про інших агентів, інформацію про спільні плани, цілі і наміри. Всередині бази знань, як уже зазначалося, можливий доступ з верхніх рівнів до нижніх. Наприклад, компонента, відповідальна за поведінку, не має доступу до знань про ментальну модель і до знань про кооперативному поведінку.

Загальне управління поведінкою здійснюється шляхом комунікацій між рівнями. При деякій вхідній події агент намагається розпізнати ситуацію в зовнішньому світі і управління поступово зсувається від низу до верху до тих пір, поки не досягне рівня, здатного впоратися з виниклою ситуацією.

Очевидно, існує три варіанти реакції агента на зовнішні події:

– реакція з використанням тільки поведінкового рівня, коли цей рівень знаходить фрагмент поведінки, адекватний ситуації, без явного залучення локального планування;

– реакція з використанням локального планування, коли завдання переміщається з нижнього рівня на рівень локального планування, де і конструюється план;

– реакція з використанням рівня кооперативного планування, коли пошук плану з рівня локального планування переміщається далі на рівень планування кооперативної поведінки.

Звичайно, існують і більш складні варіанти побудови плану, коли, наприклад, протокол взаємодії між рівнем локального планування і планування кооперативної поведінки передбачає складну схему обміну інформацією, наприклад, для побудови оцінок можливості вирішення деяких завдань багатоагентної системи за заданий час.

3 МЕТОДИ РОЗРОБКИ ТА УПРАВЛІННЯ IOT-СИСТЕМАМИ

У сучасному світі концепцію IoT неможливо уявити без використання багатоагентних технологій [17]. Для кожного фізичного об’єкта програмний агент приводиться у відповідність з певним ступенем інтелектуалізму, представляючи його інтереси в мережі.

Застосування розподілених обчислювальних систем дозволяє делегувати складні завдання програмним системам (агентам), що, у свою чергу, дозволяє представляти та вирішувати проблеми, які важко формалізувати. Коли розроблені системи розподіленого доступу, багатоагентна технологія дозволяє поєднувати як протокол, так і будь-яке прикладне програмне середовище в єдиній системі для обробки та взаємодії з різними типами даних [18]. Така система має гнучкість, масштабованість та ефективність розподілу навантаження між серверами.

Відповідно до концепції багатоагентних систем (MAS) та технологій, агент володіє лише частиною знань із загальної проблеми, в результаті чого він здатний вирішити лише частину загального завдання. Тому, щоб вирішити складну проблему, потрібно мати безліч агентів, які взаємодіють між собою, тобто багатоагентну систему. Завдання в таких системах розподіляються між агентами відповідно до їх навичок та можливостей. Будь-який агент – це відкрита система, яка має свою поведінку. Таким чином, вважається, що агент здатний сприймати інформацію з обмеженого середовища, обробляти її на основі власних ресурсів, взаємодіяти з іншими агентами і деякий час діяти в середовищі, переслідуючи власні цілі [17].

3.1 Типи та характеристики агентів

Основи багатоагентних систем сформувалися в результаті вивчення розподілених комп'ютерних систем, паралельних обчислень та мережевих

технологій. Автономність окремих системних модулів є основою багатоагентства, і такі модулі називаються агентами. Кожен агент працює в розподіленій системі, де відбувається кілька процесів, які, можливо, були взаємопов'язані одночасно. Автономний об'єкт або програма, здатна до активної мотивованої поведінки та взаємодії з іншими об'єктами в динамічному середовищі, називається агентом. Агенти мають можливість отримувати повідомлення шляхом інтерпретації їх змісту та генерування нових повідомлень, які можна надсилати іншим агентам або до ядра системи мультиагентної дошки повідомлень, яка буде доступна для всіх компонентів системи [17].

Багатоагентний підхід використовується в різних сферах, серед них є розподілені рішення складних завдань, реінжиніринг на підприємстві, взаємодія роботизованих систем IoT[18].

Існує два класи завдань, які вирішуються за допомогою багатоагентного підходу. Перший клас включає завдання розподіленого контролю та планування. Водночас зусилля різних агентів спрямовані на вирішення загальної проблеми, у таких завданнях необхідно забезпечити ефективну взаємодію агентів. Другий клас включає завдання, де кожен агент самостійно вирішує свою проблему, використовуючи спільні ресурси [19-21].

Функціонування MAS базується на принципі розподілу обов’язків між окремими підсистемами, тобто в загальному середовищі існують автономні агенти, робота яких спрямована на задоволення інтересів різних користувачів. У цьому випадку агенти взаємодіють між собою під час вирішення своїх завдань. Ці завдання включають управління інформаційними потоками, адміністрування мережі та пошук інформації в інтернеті, управління трафіком, колективне вирішення багатокритеріальних завдань та багато інших [20].

Поява агентів співпраці в розподілених системах призвела до формування сучасного представлення агента. Протягом тривалого часу багатоагентна парадигма накопичувала значну теоретичну базу та досвід [21]. Також це дослідження призвело до появи різних модельних агентів, їх типів та

характеристик, а також інструментів та засобів, необхідних для їх розробки. Були сформовані різні принципи взаємодії агентів.

Зростання складності завдань для систем IoT та розвиток розподілених обчислень збільшило інтерес до використання програмних агентів. Програмний агент – це автономний процес, який може реагувати на середовище та спричиняти зміни разом із користувачами чи іншими агентами [21]. Слід зазначити, що середовище також впливає на агент [1, 17]. Програмні агенти класифікуються за такими основними ознаками.

На основі мобільності [20]:

– стаціонарні агенти;

– мобільні агенти.

Основна відмінність між мобільними та стаціонарними агентами в цій класифікації полягає в тому, що мобільні агенти здатні переміщатися між вузлами обчислювального середовища.

За типом взаємодії [1]:

– кооперативні агенти;

– агенти -конкуренти.

Кооперативний агент має здатність інтегруватися з іншими агентами в середовищі для вирішення спільного завдання. У свою чергу, конкуренти по своїй суті конкурують за свої інтереси.

Існує також багато інших особливостей, які можна використовувати для класифікації агентів [1]. По -перше, слід зазначити, що агенти можуть виступати як живі істоти. Ознаки, які ми розглянемо далі, стосуються класифікацій штучних агентів (роботів, автоматів або комп’ютерних програм).

За функціональними цілями агентів можна розділити на дві великі групи [17]:

– функціональні агенти – це ті, які існують і працюють у реальному світі та можуть бути наділені датчиками для отримання інформації з навколишнього середовища. Прикладом таких агентів можуть бути роботи;

– інформаційні агенти існують лише у програмному середовищі, вони переважно виконують завдання, пов’язані з комп’ютерними розрахунками.

Розрізняють такі типи агентів за здатністю до навчання [18]:

– агенти, здатні навчатись, і поведінка таких агентів базується на досвіді, отриманому раніше;

– не здатні до навчання агенти, вони діють за заздалегідь записаними правилами, які реагують на зміни навколишнього середовища.

За здатністю взаємодії [17]:

– автономні агенти;

– ноу-хау взаємодії з іншими агентами.

З іншого боку, агентів можна класифікувати відповідно до їх здатності міркувати або «думати». Це найефективніший підхід до проектування інтелектуальних систем IoT. Відповідно до цієї класифікації існує два типи агентів: інтелектуальні (когнітивні) та реактивні.

Інтелектуальні агенти мають добре розвинену математичну модель зовнішнього світу, яка постійно поповнюється. Це досягається завдяки наявності агента бази знань та механізмів аналізу дій. Цей тип агентів здатний проводити аналіз на основі моделі навколишнього середовища з використанням методу вибіркового картографування і на основі цих даних приймати рішення або виконувати певну роботу. Якщо агент має певні ресурси, його база знань міститиме інформацію про структуру та стан ресурсів, що матиме значний вплив на подальшу поведінку. Інтелектуальний агент обов'язково поєднує п'ять основних функцій: когнітивну; регулятивну, здатність міркувати; комунікативну; винахідливу [17-20].

У свою чергу, реактивні агенти не мають даних про навколишнє середовище, механізм аналізу даних та ресурси. Тому ці агенти не мають механізму прогнозування змін навколишнього середовища та їхніх дій [17].

Крім того, інтелектуальний агент характеризується більшою автономією, ніж реакційний агент, маючи свої цілі, для задоволення яких вони

можуть використовувати ресурси інших агентів. У свою чергу, реакційноздатні речовини сильно залежать від зовнішнього середовища і здатні лише до відповідних реакцій.

Типові завдання, які ставляться перед агентами, включають [21]:

– тимчасові розрахунки. Робота агентів здійснюється не тільки між стаціонарними підмережами мережі, але і між мобільними платформами, які підключені до мережі. Наприклад, це може бути так: мобільний пристрій під'єднаний до мережі і додає агента, який має виконати певну роботу, а потім від'єднує його. Після виконання завдання агента пристрій знову підключається до іншого вузла мережі та завантажує результати його роботи;

– пошук, обробка та аналіз інформації. Людині важко працювати з великими обсягами даних, тому використання агентів для пошуку та обробки інформації є досить ефективним;

– моніторинг даних. Агент в режимі реального часу відстежує джерело даних і повідомляє про будь -які зміни.

3.2 Зв'язок агентів із зовнішнім середовищем

До основних типів взаємодії агентів між собою та з навколишнім середовищем належать [17]:

– співробітництво (це основна форма взаємодії між агентами та середовищем, що характеризується уніфікацією їх дій, ресурсів та засобів для досягнення спільної мети, з розподілом функцій між ними);

– конкуренція (протистояння, конфлікт);

– компроміс (важливо відповідати як власним вимогам, так і вимогам опонента);

– конформізм (відмова від їх претензій на користь опонента);

– відмова від взаємодії.

Реактивні агенти взаємодіють з іншими агентами, щоб вижити, їх спілкування не можна назвати навмисним, воно базується насамперед на

природних принципах. На відміну від розумних агентів, які співпрацюють "свідомо" для задоволення потреб. Зрештою, оскільки агент чи система здатна перебувати під впливом середовища, це відображає її ефективність. Співпраця між агентами та оточенням може виникнути як на принципах співробітництва чи вимушеному, так і на основі ситуаційного співробітництва або добровільно. Потрібні угоди та співпраця між агентами. Можна виділити такі основні причини співпраці агентів [19].

Спільна мета. Як правило, якщо агенти пов'язані цією причиною, вони будуть взаємодіяти з типом співпраці. Однак необхідно перевірити, чи така співпраця не призводить до знищення агента або його життєздатності. Можлива ще одна ситуація, коли агенти не збігаються. Тоді виникають конфлікти між об’єктами MAS. Але в цій ситуації конфлікти також можуть мати позитивний вплив на систему. Вони сприяють розвитку та стимулюють агентів. Існують системи з одночасною взаємодією, типи співпраці та протистояння. Прикладом може служити модель хижаків -жертв [20].

Для досягнення своєї мети агенту потрібні певні ресурси, тобто ресурси. Якщо у агентів немає спільних ресурсів - виникають конфлікти. Для вирішення цієї проблеми сказано, що правило "виграє сильніше". Тобто сильніший агент забиратиме ресурси у слабшого. Це можна назвати найефективнішим і найпростішим способом вирішення таких конфліктів. Але в деяких ситуаціях доцільно вести переговори [21]. У цьому випадку агенти йдуть на компроміс, враховуючи інтереси кожного.

Кожен агент використовує обмежений набір знань, які йому або їй можуть знадобитися для досягнення місцевих та глобальних проблем. Тому йому доводиться шукати взаємодії з іншими агентами [17].

Таким чином, виділяють наступні обставини:

– агент здатний досягти мети без допомоги інших, тобто самостійно;

– агент здатний досягти мети самостійно, але шляхом взаємодії (проблему можна вирішити ефективніше або швидше);

– агент може досягти мети лише за допомогою сторонньої допомоги.

Агенти можуть самостійно вибирати тип взаємодії з кожним агентом або середовищем, залежно від актуальності зв'язку.

Для того, щоб встановити порядок між агентами в процесі взаємодії, існують зобов'язання. За допомогою зобов’язань можна передбачати дії інших агентів та планувати їх власні. Нижче наведені такі види зобов’язань [1]:

– агент зобов'язаний перед іншими агентами;

– агент зобов'язаний групі;

– група зобов'язана агенту;

– агент зобов'язаний сам собі.

Офіційне представлення цілей, зобов'язань, бажань і намірів, а також усіх інших даних, лежить в основі ментальної моделі інтелектуального агента, що забезпечує його мотивовану поведінку в автономному режимі.

Існують різні форми агентської співпраці [17]:

– звичайне співробітництво, яке досягається шляхом обміну досвідом кожного агента (обмін завданнями, обмін знаннями тощо) без спеціальних заходів щодо координації їх дій;

– скоординована співпраця, якщо агентам доводиться координувати свої кроки (іноді використовуючи так званий агент-координатор) для ефективного використання ресурсів та свого досвіду;

– невиробнича співпраця, якщо агенти разом використовують ресурси або вирішують спільні проблеми, не обмінюючись досвідом та не заважаючи один одному.

3.3 Методи передачі даних між агентами в IoT-системах

Для того, щоб агенти передавали інформацію один одному в розподілених системах, вони використовують взаємодію агента. Для цього MAS використовує [17]:

– універсальні мови програмування, такі як Java;

– мови, орієнтовані на знання, тобто мови представлення знань (формат обміну знаннями (KIF)); мова взаємодії агента (фонд інтелектуальних фізичних агентів (FIPA), мова запитів та маніпуляцій знаннями (KQML), AgentSpeak, April);

– мова специфікацій агента;

– спеціалізовані мови програмування для агентів (TeleScript);

– мови опису сценарію (Tc/Tk);

– мови логічного програмування (Oz);

– мови, подібні до Lisp.

Для розробки мови обміну даними між агентами можна використовувати два різних підходи. Перший підхід є процедурним, що означає, що спілкування ґрунтується на реалізації інструкцій. Така мова може бути розроблена та запрограмована на Java або на такому інструменті розробки, як Tcl. Другий підхід - декларативний, тобто спілкування базується на описах. Цей підхід став більш широко використовуватися для створення мов спільного використання між агентами [1]. Найпопулярнішими стандартами, що визначають мову спілкування агентів, є FIPA [11] та KQML [19].

Стандарт FIPA. Розроблено Комітетом FIPA [17]. Він включає мову FIPA ACL (мова спілкування агента) [7], за допомогою якої агенти можуть передавати повідомлення певного формату за допомогою різних служб передачі даних, та мову, подібну до LISP, що описує зміст повідомлення FIPA SL (семантична мова). Внутрішня архітектура стандарту FIPA складається з таких служб, які інтегровані в загальний реєстр:

– служба повідомлень;

– послуга реєстрації агентів у середовищі (тобто в MAS);

– службовий опис мови агентів зв'язку;

– реєстр усіх послуг.

Ця архітектура може взаємодіяти із зовнішніми системами для управління та реалізації поточних завдань агента.

Стандарт KQML. Розроблено Комітетом ARPA (Агентство перспективних дослідницьких проектів) [17]. Він включає в себе мову KQML, яка визначає набір перформативних дій, та мову, подібну до LISP, для опису змісту повідомлення, KIF. Стандарт складається з трьох рівнів: комунікативного рівня (описує такі параметри, як відправник, одержувач та різні ідентифікатори повідомлень), рівня повідомлення (описує запити, дії керування та протокол інтерпретації повідомлення) та рівня вмісту (містить інформацію, що супроводжує запити рівня повідомлень).

Стандарт характеризується такими основними особливостями:

– агенти з'єднані односторонніми каналами зв'язку, по яких передаються фіксовані комунікації;

– канали зв'язку можуть мати ненульову затримку в передачі повідомлення;

– при отриманні повідомлення агент визначає, на кого і на яке вхідне повідомлення надійшло це повідомлення;

– агент може надіслати повідомлення лише через певний канал;

– повідомлення для конкретного адресата надходять у порядку відправлення;

– доставка повідомлень абсолютно надійна.

Стандарт підтримує як синхронну, так і асинхронну передачу повідомлень. Агенти можуть спілкуватися безпосередньо з іншими агентами (із символічним іменем), надсилати трансляційні повідомлення або "запитувати" інших агентів-учасників розмови. Нижче наведені приклади систем, реалізованих за допомогою KQML [1, 17]:

– програмне забезпечення Next-Link [1], розроблене в Стенфордському університеті, спрямоване на вивчення принципів координації, що дозволяє штатним агентам здійснювати розподілене проектування та проектування систем;

– логічно-центрований дизайн [18], розроблений у «Центрі штучного інтелекту» Lockheed AI Center, який позиціонується як інтелектуальна інформаційна система для проектування систем;

– Concur [19] презентації веб-серверів, розроблені в Стенфордському університеті Грегорі Р. Олсеном, який використовує агентський підхід до обчислень у розподілених системах проектування;

– програмний комплекс для розподіленого збору даних[20].

3.4 Вимоги до мов програмування агентів

Перш ніж почати порівняння і аналіз агентських мов, розглянемо основні вимоги, що пред'являються до них. Найбільш важливими представляються нижченаведені.

1) Забезпечення переносимості коду на різні платформи. Ця вимога виникає завжди, коли необхідно забезпечити агента властивістю мобільності. Щоб забезпечити мобільність агента, мова повинна підтримувати механізм посилки, передачі, отримання та виконання кодів, що містять агентів. Є два різних підходи, які вирішують проблему мобільності. Перший – це передача агента в текстовій формі, як спеціального сценарію (script) з подальшою інтерпретацією цього сценарію на приймаючій машині. Другий – передача агента в формі машинно-незалежного байт-коду. Цей байт-код генерується транслятором на етапі створення агентської системи, надсилається через мережу і виконується інтерпретатором байт-кодів на приймаючому комп'ютері. Обидва з цих методів мають свої переваги і недоліки.

2) Доступність на багатьох платформах. Ця вимога безпосередньо випливає з попереднього. Інтелектуальні агенти повинні працювати в гетерогенної комп'ютерної середовищі. Будь-який комп'ютер, який отримує агента, повинен бути здатний прийняти і виконати його.

3) Підтримка мережевої взаємодії. Властивість агентів брати участь в переговорах і багато інших особливостей агентів потребують доступу до

віддалених ресурсів. Підтримка мережевих послуг може включати сімейства відповідних програмних інтерфейсів (APIs) таких як: sockets, інтерфейси до баз даних, інтерфейси взаємодії об'єктів (CORBA, OLE, ActiveX і т.д.), спеціальні механізми, вбудовані в мову (типу Remote Method Invocation в мові Java), спеціальні примітиви мови для здійснення переговорів агента і т.д.

4) Багатопотокова обробка («multithreading»). Агент може виконувати деякі дії одночасно. Тим самим, мова програмування агентів повинна включати підтримку паралельного виконання різних функцій агента («threads») і різних примітивів синхронізації (семафори, монітори, критичні секції, і т. д.). Крім того, процес який виконують всі агенти (і може, фактично, розглядатися як мета-агент) повинен підтримувати паралельне виконання агентів. Останнього можна досягти за допомогою окремої віртуальної машини з реалізованим режимом витіснення багатозадачності і власною стратегією поділу часу.

5) Підтримка символьних обчислень. Так як в рамках сучасних поглядів агент повинен активно використовувати досягнення і методи штучного інтелекту, було б корисно мати підтримку символьних обчислень і, можливо, логічного програмування, вбудовану в мову (подібно PROLOG і LISP), а також мати вбудований механізм висновків, що включає різні стратегії пошуку рішення. Автоматичне управління пам'яттю і видалення сміття – стандартні засоби для таких мов.

6) Безпека, зокрема, наявність системи захисту від несанкціонованого доступу і «поганих кодів». Це важливо з багатьох причин. Мобільні агенти, які приходять з мережі, можуть приховувати в собі безліч небезпек для приймаючої машини, так як вони виконуються в її адресному просторі. Для забезпечення безпеки перед передачею управління кожному агенту необхідно виконати процес авторизації агента, тобто перевірити, чи зареєстрований він і чи має відповідні повноваження (привілеї), щоб виконати ту чи іншу дію або звернутися до деяких ресурсів. Система безпеки повинна запобігати будь-які несанкціоновані дії агента.

7) Справжня об'єктна орієнтованість. Мова повинна мати механізми наслідування, розглядати виклики процедур як повідомлення, що передаються від одних об'єктів іншими, включати можливості синхронної і асинхронної взаємодії об'єктів, а також допускати паралельність всередині об'єкта.

8) Мовна підтримка властивостей агента. Було б, ймовірно, зручно мати підтримку специфічних для агента властивостей, вбудовану в мову на рівні синтаксичних конструкцій, так, щоб, наприклад, властивості типу «beliefs-desires-intentions» (BDI), інструкції для переговорів і забезпечення мобільності, місця зустрічі, і т.д. могли б бути виражені за допомогою відповідних примітивів мови.

9) Ефективність, достатня для потреб прикладних програм.

4 АГЕНТНА МОДЕЛЬ IOT-СИСТЕМИ

Хмарні обчислення – це ефективне рішення для розподіленої обробки великих даних. Сучасні системи моніторингу вимагають використання хмарних обчислень. Зазвичай в системах моніторингу не використовуються методи та моделі ШІ. В хмарному середовищі система моніторингу має збирати всю інформацію на всіх етапи обробки завдань [49].

Об'єднання можливостей віддаленого підключення всіляких пристроїв надає концепція Інтернету речей.

4.1 Cloud-Fog-Dew архітектура для IoT-системи

Проблемно-орієнтований підхід найчастіше використовується для планування хмарних обчислень. При цьому завдання мають бути незалежними один від одного, також треба враховувати обсяг оброблюваних даних, та послідовність виконання всіх завдань [49].

Отримати послуги на вимогу через Інтернет можна завдяки парадигмі хмарних обчислень. Однак, існує проблема оптимізації завдання шляхом скорочення часу виконання клієнтських завдань при мінімізації пов'язаних з цим витрат [50]. Сервіс-орієнтовані портали мають можливість багато разів надавати сервіси, які не знають застосунки, що їх викликають, та навпаки, користувачі не знають, яким чином буде здійснена викликана послуга.

Технологія Інтернет речей (IoT) поєднує як фізичні об’єкти, так і комп’ютерні пристрої, що надає віддалений доступ до сервісів. Інтернет речами можуть бути різні об’єкти (роботи, різноманітні датчики, смартфони, тощо), які можна ідентифікувати в комунікаційні мережі.

Архітектура Cloud-Fog-Dew дозволяє забезпечити ефективний доступ до сервіс-орієнтованого порталу користувачам з використанням різноманітних датчиків і мобільних пристроїв [51]. В роботі пропонується

модифікувати систему, що запропонована в [50] шляхом подавання на Fog-сервері сервісу визначення місця розташування (рис.4.1).

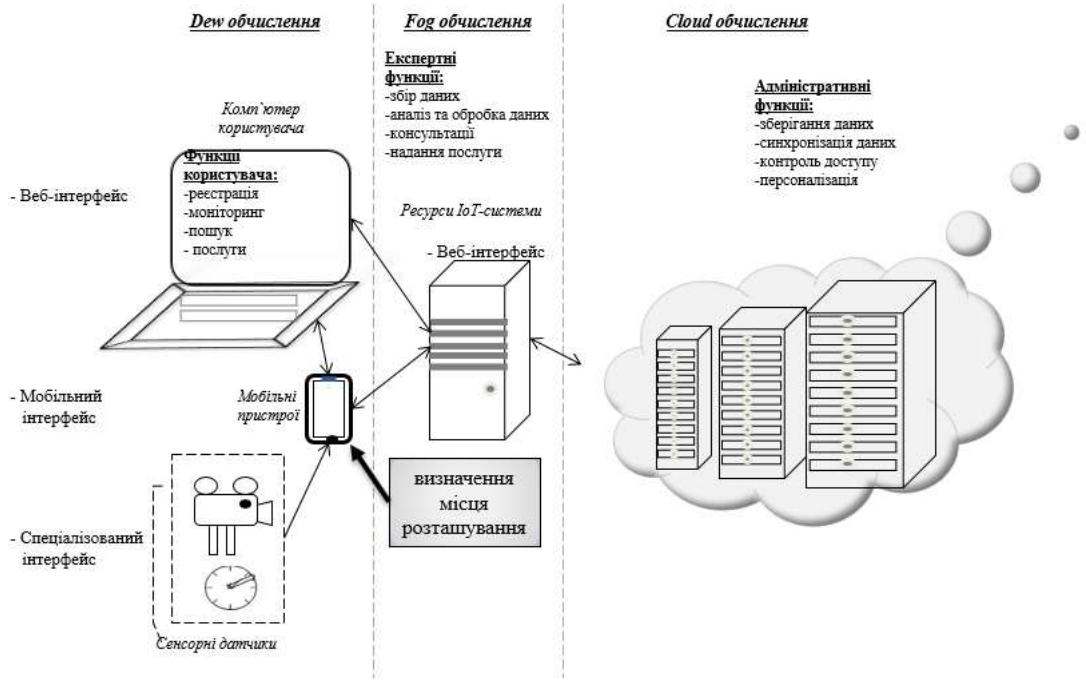

Рисунок 4.1 – Cloud-Fog-Dew архітектура для IoT-системи

Така система призначена для віддаленого моніторингу стану здоров’я користувача, та в разі необхідності – надання медичної допомоги. У разі виникнення критичного стану пацієнта, пропонована система визначає його місце розташування. Для визначення стану здоров’я пацієнта система використовує механізм «градації» вимірюваних значень за рівнем ваги на: критичний, високий, середній, низький. Прикладами були вимірювання кардіограми, кров’яного тиску, рівня цукру, температури, тощо. Користувачами були пацієнти з хронічними захворюваннями на різних етапах, та медичний персонал (лікарі, доглядальниці, спостерігачі) [53]. Користувач (пацієнт) реєструється спеціалізованому веб-порталі та на його мобільний пристрій встановлюється спеціальний застосунок. До системи можуть бути підключено всіляке медичне обладнання, мобільні пристрої, камери відеоспостереження, які поєднані між собою спеціальним інтерфейсом. Пропонована послуга визначає місце розташування мобільного пристрою, з

якого дані пересилаються Fog-серверу для подальшої обробки.

До Dew обчислень відноситься реєстрація персональної та діагностичної інформації, обробка зображень користувача у рухомому стані та в стані спокою, а також постановка попереднього діагнозу.

Використання такої архітектури дозволяє працювати в автономному режимі. А в разі підключення до Інтернет стан Dew сервера синхронізується із хмарою. Обсяг такого серверу значно менший ніж Cloud, тому що зберігає тільки власні дані користувача.

Перевагою Fog-сервера є мінімальна затримка в обробці даних для надання швидкої медичної допомоги завдяки визначенню місця розташування та близькому розміщенню до кінцевого користувача.

4.1.1 Метод знаходження місце розташування загальних пристроїв для Інтернет речей

Вбудовані об’єкти в загальнодоступних зонах можуть створювати слабкі зв’язки, які можуть використовувати зловмисні об’єкти, і можуть здійснювати незаконне спостереження, відстеження, відстеження та профілювання переміщень та діяльності користувачів. Також існує ряд автоматизованих векторів атак на конфіденційність в Інтернеті речей, наявні та успадковані проблеми конфіденційності, наприклад інциденти, які призводять до загроз конфіденційності місцезнаходження користувачів. Можна збирати особисту інформацію користувачів з об’єктів IoT без відома або згоди особи. Ці проблеми конфіденційності, з якими стикається Інтернет речей, мають новий характер у порівнянні з тими, які виникають у сучасному онлайн-комунікації, яка здебільшого стосується безпосередньо користувача. Як правило, у традиційному онлайн-середовищі користувач контролює свій вибір щодо конфіденційності, оскільки він чи вона безпосередньо бере участь у доступі до Інтернету. Це більше не може бути у випадку з автоматизованими об’єктами Інтернету речей, де засоби контролю доступу та політики конфіденційності до

інформації мають бути заздалегідь визначені та виконані без участі користувача в реальному часі. Принцип, відомий як повідомлення та вибір, кидається під сумнів в Інтернеті речей. Також виникає питання про те, чи мають користувачі контроль над розкриттям свого місцезнаходження в Інтернеті речей і принципи, які мають регулювати розгортання технологій IoT із підтримкою місцезнаходження.

З появою повсюдних обчислень і повсюдних обчислень, а також поширенням портативних і бездротових пристроїв, термін усвідомлення контексту став більш помітним. Спочатку контекстна обізнаність описує пристрої, які можуть відчувати своє фізичне середовище та відповідно змінювати свою поведінку. З іншого боку, поінформованість про місцезнаходження також з'явилася як зростаюча тенденція в апаратних і програмних додатках і стала більш поширеною на портативних пристроях. Останні розробки в технологіях визначення місця розташування дозволили портативним пристроям дізнаватися про своє місцезнаходження. Це призводить до впровадження послуг на основі місцезнаходження та додатків на основі розташування.

Інновації в бездротових і комунікаційних мережах, які живлять портативні пристрої з визначенням місцезнаходження, зробили захист конфіденційності громіздким завданням, зокрема конфіденційності місцезнаходження. Користувачі швидко втрачають контроль над рішеннями про розголошення їх особистої інформації. Наприклад, додатки, що базуються на місцезнаходженнях, здебільшого вважаються вразливими до великомасштабної втрати конфіденційності. Ці програми по суті залежать від розташування користувачів, у якому користувачі довіряють програми, що працюють на ненадійних сторонніх серверах. Контекстно обізнані системи, такі як мобільні додатки, активно збирають особисту інформацію користувача, зокрема інформацію про місцезнаходження, без згоди або відома користувача. Хоча в звичайній ситуації користувач може охоче розкрити інформацію про своє місцезнаходження, він або вона може не захотіти розкривати своє

місцезнаходження всім у будь-який час. Насправді, інформація про місцезнаходження стає дуже чутливою, коли вона поєднується з іншими контекстними даними, такими як особистість користувача. Знання особистості, місцезнаходження та інших контекстних даних, таких як торгові звички користувачів, покращило послуги, які надаються користувачам, підвищивши якість послуг, що надаються користувачам. З іншого боку, коли ця інформація потрапляє в чужі руки,

IoT розширює взаємодію між людьми та додатками на новий вимір спілкування через об'єкти. Замість того, щоб завжди взаємодіяти з людьми-користувачами, об'єкти будуть взаємодіяти один з одним автономно, виконуючи дії від імені користувачів та оновлюючи їхній щоденний розклад. Тому контекстна обізнаність розглядається як технологія, яка сприяє розвитку Інтернету речей. Контекстно обізнані об'єкти в IoT займаються отриманням контексту, наприклад, за допомогою сенсорів, щоб відчувати навколишнє середовище і, отже, сприймати ситуацію на основі цього; або виконання механічного руху з використанням приводів. Об'єкт також займається аналізом контексту, наприклад, встановленням відповідності служб до контексту, а також розпізнаванням контексту, наприклад, виконання деяких дій або ініціювання події на основі розпізнаного контексту. таким чином,

В основному, контекстні дані в Інтернеті речей використовуються для надання індивідуальних послуг, підвищення якості/точності інформації, виявлення сусідніх служб і здійснення неявних взаємодій користувачів. Однак це має свою ціну. Автоматизована атака вторгнення – це поступовий процес атак висновку, в якому зловмисник поступово збирає більше знань про життя або діяльність користувача за допомогою комбінації та зв'язування інформації, зібраної з різних джерел об'єктів,

якими володіє, керує або контактує з ними. користувач. Контекстна інформація може стосуватися безпосередньо користувача, або вона може бути пов'язана із завданнями чи діяльністю користувачів та їх соціальними взаємодіями. Місцезнаходження, дата, час, особистість, знання торгових

звичок, тип спілкування, виконання завдання та фізичні параметри (шум, світло).

В Інтернеті речей користувач більше не є неявним джерелом інформації; і тому вибір конфіденційності не може прямо покладатися на рішення користувачів - навантаження, якого ми хочемо уникнути, якщо це взагалі можливо, оскільки комунікація в Інтернеті речей у великій частині є автономною між об'єктами, що не вимагає участі людини-користувача. безпосередньо. Тому залишаються проблеми із забезпеченням конфіденційних рішень, які автономно адаптуються до змін у контексті.

Термін обфускація визначає практику навмисного погіршення якості інформації про місцезнаходження, щоб захистити конфіденційність особи, до якої ця інформація відноситься. Обфускація місцезнаходження – це техніка, яка використовується для захисту місцезнаходження користувача шляхом узагальнення інформації про місцезнаходження або використання заміни. Однак приховування інформації про місцезнаходження неефективне, якщо власники інформації про місцезнаходження можуть не захотіти приховувати інформацію про своє місцезнаходження в будь-який час або в будь-яких ситуаціях. Проблема полягає в тому, щоб надати рішення, яке б варіювало ступінь конфіденційності розташування, використовуючи різні рівні обфускації. Визначення рівня обфускації здійснюється на основі контексту спілкування та політики конфіденційності, визначеної користувачем. Для досягнення цього пропонується метод, який доповнює та використовує існуючі методи захисту конфіденційності. В основу метода покладена агентська технологія для динамічного розкриття розташування (ДРР). Об'єкт 1 запитує розташування об'єкта 2. Об’єкт 2 пересилає запит до агента ДРР, приєднаного до нього, шляхом введення поточного розташування. Агент ДРР визначає вихідне місце розташування за допомогою аналізу контексту та компонентів формування розташування. Вихід місцезнаходження пересилається назад до об’єкта 2. Об'єкт 2 надсилає об'єкту 1 вихідне місцезнаходження як його поточне розташування (рис. 4.2).

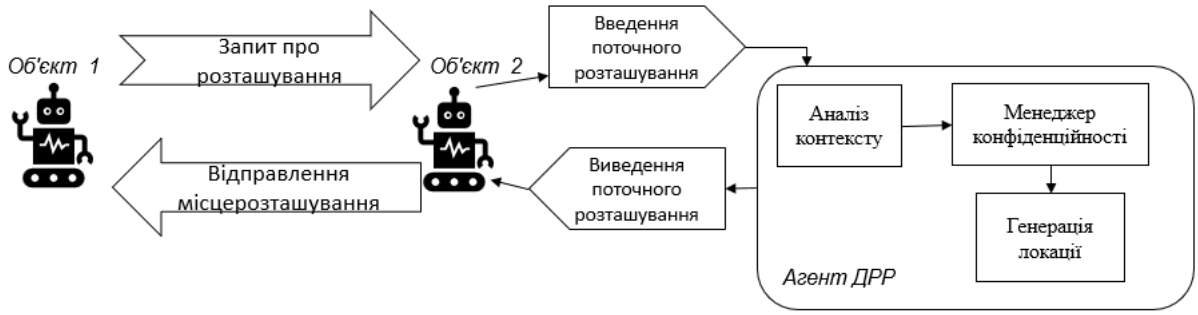

Рисунок 4.2 - Агент ДРР

Передбачається, що об’єкти мають вбудовані можливості обробки та зв’язку. Об’єкт запитує інформацію про місцезнаходження іншого об’єкта через бездротові чи мобільні мережі. Агент бере поточне розташування другого об'єкта як вхідні дані та виводить заплутане розташування, яке змінюється за ступенем чи точністю залежно від контексту зв'язку та на основі значень контекстних параметрів. Наприклад, у певному місці для двох різних запитувачів інформації про місцезнаходження агент може надати різну інформацію про місцезнаходження, кожен відповідно до своїх параметрів, які ґрунтуються на контексті. Агент ДРР містить три основні компоненти: компонент аналізу контексту, який дозволяє агенту бути контекстуально обізнаним про поточне розташування об’єкта, статус мобільності, тип підключення до Інтернету та запитувача серед інших обчислюваних параметрів. Другим компонентом є менеджер конфіденційності, який зберігає визначені користувачами параметри конфіденційності. Потім агент визначає, чи можна розкрити місцезнаходження запитувачу і який рівень обфускації буде використаний. Це робиться за допомогою компонента генерування розташування, який застосовує деякі просторові обмеження до кожного виходу розташування. Просторові обмеження можуть бути у формі обмежень, заснованих на часі, даті та часі та даті закінчення терміну дії. Зокрема, компоненти агента ДРР взаємодіють наступним чином: об'єкт 1 запитує місцезнаходження об'єкта 2. Це може бути прямий запит як частина запиту

зв'язку, або об'єкт 2 може запитувати деяку інформацію від об'єкта 1, який, у свою чергу, запитує об'єкт 2 про його розташування. Об’єкт 2 посилається на агента ДРР і розміщує запит на розкриття місцезнаходження, вказуючи його поточне справжнє місцезнаходження. Агент запитує від об'єкта 2 іншу інформацію, необхідну для компонента аналізу контексту. Ця інформація описує поточний контекст об’єкта 2, наприклад, його статус мобільності та поточні налаштування мережі. Аналогічно, агент також запитує деяку інформацію про об'єкт 1, яка може бути відома об'єкту 2. Це може бути у формі будь-якої ідентифікаційної інформації для об'єкта 1 та його поточних мережевих налаштувань. далі, агент запитує дозвіл у менеджера конфіденційності, щоб отримати будь-які визначені політики конфіденційності користувачів. Потім об’єкт 2 використовує вихідні дані розташування для зв’язку з об’єктом 1. Процес повторюється, якщо інший об’єкт, наприклад об’єкт 3, запитав місцезнаходження об'єкта 2, навіть якщо об'єкт 2 все ще знаходиться в тому самому поточному місці. Рівень обфускації обчислюється та визначається на основі аналізу поточного контексту за допомогою компонента аналізу контексту. Контекст аналізується за допомогою контекстних параметрів чотирьох шарів: рівня мережі, розташування, періоду виконання та запитувача, як показано на рисунку 4.3.

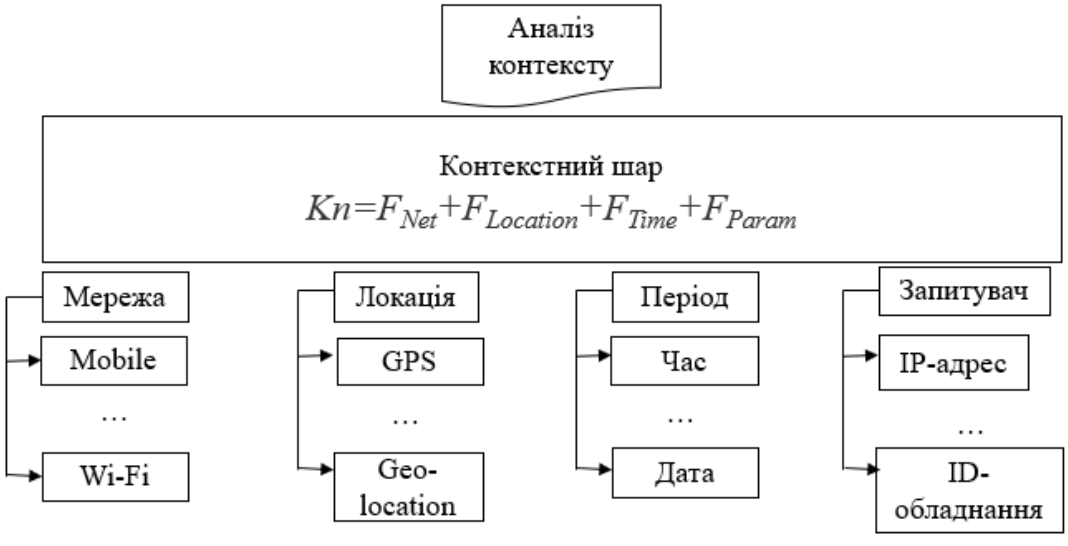

Рисунок 4.3 – Компонент аналізу контексту

Таким чином, для даного сценарію контекст K_n, визначається за допомогою твердження:

$$Kn=F_{Net}+F_{Location}+F_{Time}+F_{Param} \quad (4.1)$$

де F_{Net} представляє контекстні параметри, пов’язані з налаштуваннями мережі, наприклад, мобільна мережа або Wi-Fi; $F_{Location}$ представляє поточне розташування об'єкта; F_{Time} включає час і дату взаємодії; F_{Param} представляє контекстні параметри, які ідентифікують об'єкт від іншого, наприклад ідентифікатор об'єкта або IP-адреса. Положення будь-якого місця на Землі в двовимірному масштабі можна визначити за допомогою спряженої сітки, де перетинаються широта і довгота. Визначення точних координат широти та довготи місця доступне за допомогою багатьох технологій, таких як супутниковий приймач глобального позиціонування, який може зв’язуватися з супутниками над Землею для тріангуляції до певного положення. Отже, розташування об’єкта в географічному просторі можна представити у вигляді точки на карті і позначити L, де L (це кортеж з 2 величин (широта, довгота)) – член множини LS, такий що L∈SL, де SL — це набір локацій. Для кожного елемента L∈ SL, визначимо базову точку SL (X_{si},Y_{sj}), щоб представляти кожну L∈SL. Тоді множина SL є підмножиною іншої множини Ω (SL). У свою чергу нехай Ω (LS) be є підмножиною головного набору Ω (Ω(LS)). Кожен з цих трьох наборів має базову точку у відповідній підмножині. Таким чином, вибравши набір, можна використовувати інше розташування базової точки, і, отже, різні рівні обфускації забезпечуються за допомогою різних базових точок. На рисунку 4.4 зображено цю логіку.

Політику конфіденційності користувач може визначити в компоненті менеджера конфіденційності. Наприклад, політику можна визначити, щоб не розкривати точне місцезнаходження об’єкта на певний час чи дату. Інша політика може додавати обмеження за часом і датою до виводу розташування для конкретних запитувачів у певних мережах.

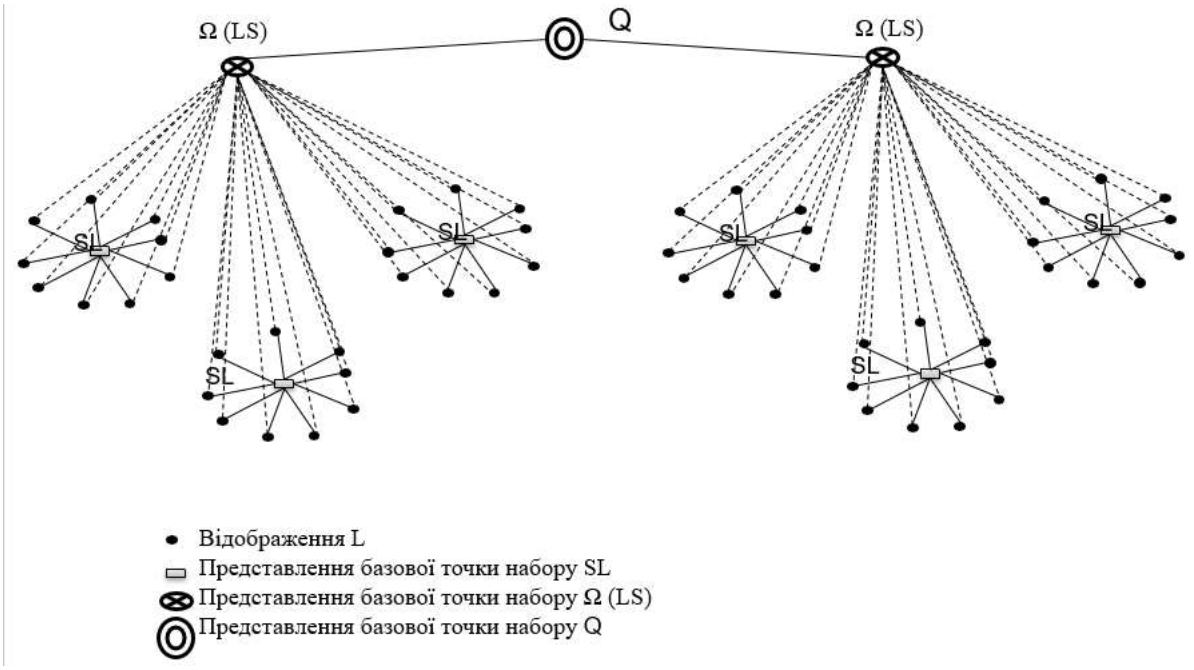

Рисунок 4. 4 – Декартове представлення трьох множин

Користувач може також визначити, який рівень обфускації використовуватиметься в певних контекстах. Наприклад, можна визначити політику конфіденційності, яка стверджує: з понеділка по п’ятницю з 12:00 до 13:00 не розголошувати точне місцезнаходження; натомість розкривати моє місцезнаходження, наприклад, в інші години. Профіль конфіденційності за замовчуванням також можна визначити в компоненті диспетчера конфіденційності. Цей профіль за замовчуванням застосовуватиметься за відсутності будь-якої визначеної політики конфіденційності. Наприклад, політика конфіденційності розташування за замовчуванням може визначати, щоб не розголошувати точну інформацію про місцезнаходження об’єктів невідомим об’єктам у певний час доби та в певних мережах.

Далі, обмеження часу існування (TTL) встановлює дату закінчення терміну дії та час для виводу місцезнаходження, щоб запобігти безперервному відстеженню розташування. Використання TTL гарантує, що розташування об’єкта не відстежується постійно. Таким чином, агент ДРР дозволяє користувачам або операторам об’єкта визначати точність і ступінь розкриття інформації про їх місцезнаходження. Таким чином, пропонується контекстно-

адаптивне рішення, яке різниться за ступенем деталізації та обмеженості.

Розроблено сценарій для моделювання взаємодії описаної між об’єктами 1 та 2: комп’ютерна програма, що представляє об’єкт 1, запитує місцезнаходження мобільного пристрою, який представляє об’єкт 2. Взаємодія відбувається наступним чином: Використовуючи інтерфейс користувача комп’ютерної програми (об’єкт 1), через Інтернет розміщується запит на місцезнаходження об'єкта 2 (мобільний пристрій). Мобільний пристрій отримує цей запит і посилається на приєднаного до нього агента, одержуючи його поточне місцезнаходження. Мобільний пристрій також надає агенту контекстну інформацію, необхідну для аналізу контексту. Це час, дата та поточна назва мережі, наприклад WiFi home, а також будь-яка відома інформація про запитувача, наприклад його IP-адреса. Крім того, агент має панель керування, яка дозволяє користувачеві заздалегідь визначити політику конфіденційності. Ці політики конфіденційності порівнюються з поточною отриманою контекстною інформацією. Це призводить до вибору відповідного рівня обфускації. Агент генерує координати розташування базових точок для цього вибраного рівня обфускації, додає до нього TTL та будь-які інші обмеження, визначені в диспетчері конфіденційності, і надсилає цей результат на мобільний пристрій. Мобільний пристрій відповідає комп’ютерній програмі з виведенням місцезнаходження як поточного розташування. Агент дозволяє визначати наведені нижче політики конфіденційності:

Мережні обмеження: це обмеження політики для певної мережі, наприклад мобільної мережі.

Часові обмеження: тут також можна застосувати обмеження за часом, наприклад, з 9:00 до 17:00.

Обмеження дати: тут також можна застосувати обмеження щодо днів тижня. Наприклад, з понеділка по п'ятницю.

Обмеження розташування: це політика, яка встановлює певний рівень обфускації.

Налаштування за замовчуванням. Дозволяють застосовувати певний

набір конфігурацій (обмежень) як профіль за замовчуванням. Цей профіль за замовчуванням використовується за відсутності спеціальних політик, які керують конкретними об’єктами.

Мобільний пристрій запитує поточне положення (довгота та широта) через GPS. Якщо GPS недоступний, він запитує його через мережу Wi-Fi. Якщо Wi-Fi також недоступний, він витягує координати розташування з мережі мобільного телефону. Потім пристрій передає свою позицію агенту. Агент встановлює поточне справжнє розташування як LO і створює фіктивне розташування, яке буде використовуватися для рівня 5 (L5). Потім агент переходить до пошуку координат трьох базових точок, що відповідають трьом рівням (L1, L2 і L3). Це базові точки множин.

Процес виглядає наступним чином:

Агент надсилає поточні координати місцезнаходження в Google Geocoder API у форматі (X, Y), де X та Y – цілі числа, що представляють широту та довготу. Слід зазначити, що Google Geocoder API використовується лише як сервіс для перетворення координат у можливу адресу. Google Geocoder API не повідомляє про справжнє місцезнаходження об’єкта.

Google API перетворює ці координати в читабельні адреси у форматі (назва вулиці, передмістя, штат і країна).

Агент визначає передмістя як перший набір L5, стан як другий набір ψ і країна як третій набір ξ. Потім він зв’язується з Google API, щоб отримати координати кожної базової точки, знайденої в кожному наборі, за допомогою зворотного пошуку. Таким чином, агент визначає 4 координати, які можуть представляти поточне місцезнаходження, кожна в іншому наборі (враховуючи L0, це справжнє точне розташування).

Фальшиве розташування L5 генерується шляхом рандомізації значень широти та довготи. На рисунку 4.5 показано, як множини отримуються від заданої адреси.

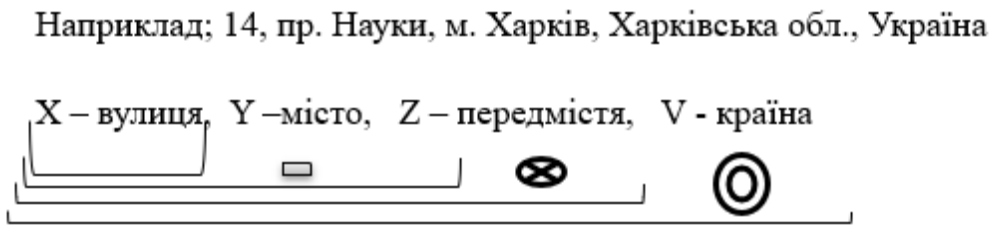

Рисунок 4.5 – Перетворення базових точок у реальні адреси

Після визначення базових точок у кожному наборі агент приступає до аналізу контексту та політики конфіденційності, щоб визначити, яка базова точка буде використовуватися як вихідне місце розташування. За відсутності будь-якої відповідної політики конфіденційності агент використовує профіль налаштувань за замовчуванням.

План тестування включав двадцять тестових випадків, які перевіряють генерацію кожного з рівнів обфускації. Мобільний пристрій автоматично збирав дані про місцезнаходження в Харкові та прилеглих передмістях. Вхідні дані про місцезнаходження також збиралися за допомогою різних мереж (Wi-Fi і мобільна мережа); і з використанням GPS і без нього.

Порівнявши поточне розташування (вхідне місцезнаходження) з очікуваним розташуванням (вихідним місцезнаходженням) та отриманим розташуванням (це розташування, отримане комп’ютерною програмою, що представляє об’єкт 1), можна перевірити, чи були результати місцезнаходження успішно згенеровані агентом для заданого контексту. Для кожного з двадцяти тестових випадків, виконаних у цьому тестуванні, агент створив вихідні дані про розташування з нульовими помилками.

На рисунку 4.6 показаний вихід (відповідь), отриманий об'єктом 2 (комп'ютером).

Хоча агент DLDA генерував вихідні дані про розташування без помилок, йому не вдалося приховати розташування в одному конкретному випадку.

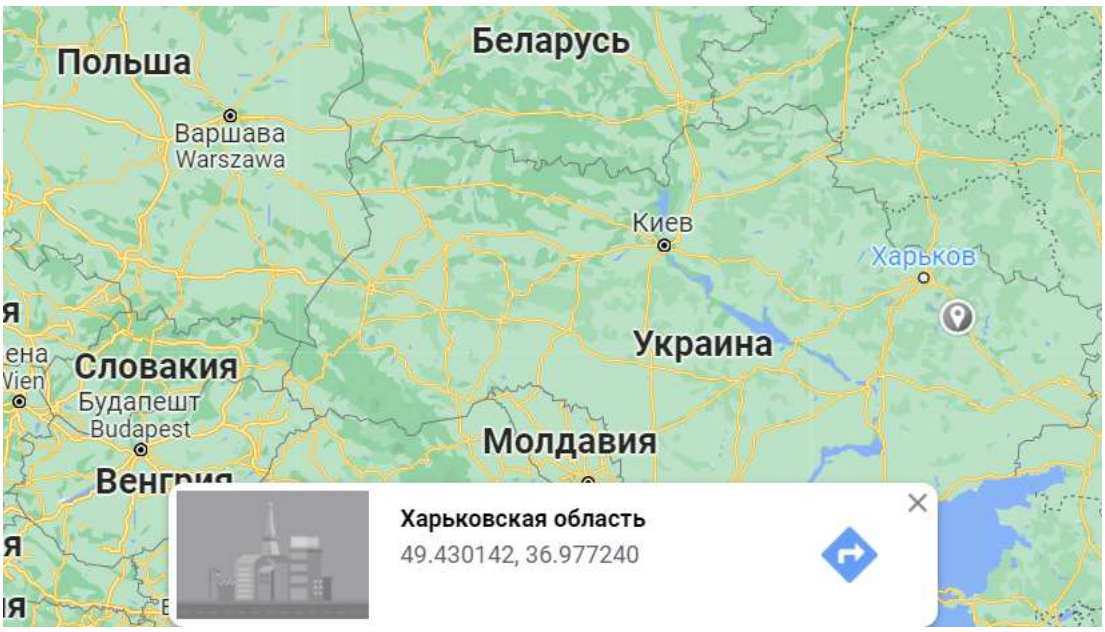

Рисунок 4.6 – Відповідь сервера

Якщо поточне справжнє місцезнаходження об’єкта 2 розташоване дуже близько до координат базової точки рівня 1 (передмістя), і якщо агент налаштований на розкриття розташування передмістя (обфускація L1), то зазначається, що розташування виводить L0 і L1 географічно близькі один до одного, якщо не однакові.

4.2 Загальна структура IoT системи оперативного реагування

В критичних областях особливе значення має надійність функціонування сервіс-орієнтованої системи. Тому важливим є надання можливостей сервіс-орієнтованій системі, які дозволяють адаптуватися у різних ситуаціях.

Для цього можна використовувати концепцію агента на організаційному рівні. Агентам можуть бути призначені різні ролі, дозволяючи використовувати такий підхід багато раз. Будемо використовувати організаційну модель у вигляді «агент/група/роль» (АГР) для аналізу агентно-орієнтованих систем на рівні організацій зі здатністю адаптуватися до непередбачених обставин, підтримуючи надійність системи. Де агент виконує ролі всередині групи та

може спілкуватися один з кожним; групи – набори ролей; роль – функція, яку виконує агент. Декілька агентів можуть виконувати однакові ролі, при цьому один агент може грати не одну роль. Зв'язок між ролями визначається їх нею взаємодією. Набор ролей створює групову структуру з взаємозв'язками між ними.

Поведінкові характеристики організації описуються тимчасовими зв'язками станів ролей. Динамічні характеристики надаються за допомогою мови темпорального трасування (Temporal Trace Language, TTL) з такими поняттями як стан, момент часу та трасування. Характеристики ролі описують поведінку окремої ролі, групові характеристики показують поведінку групи в цілому.

Структура IoT системи оперативного реагування в екстрених ситуаціях з використанням організаційної концепції агент-група-роль наведена на рисунку 4.7. Призначенням такої системи є виявлення та усунення критичних станів досліджуваного об'єкта з визначенням його місцярозташування.

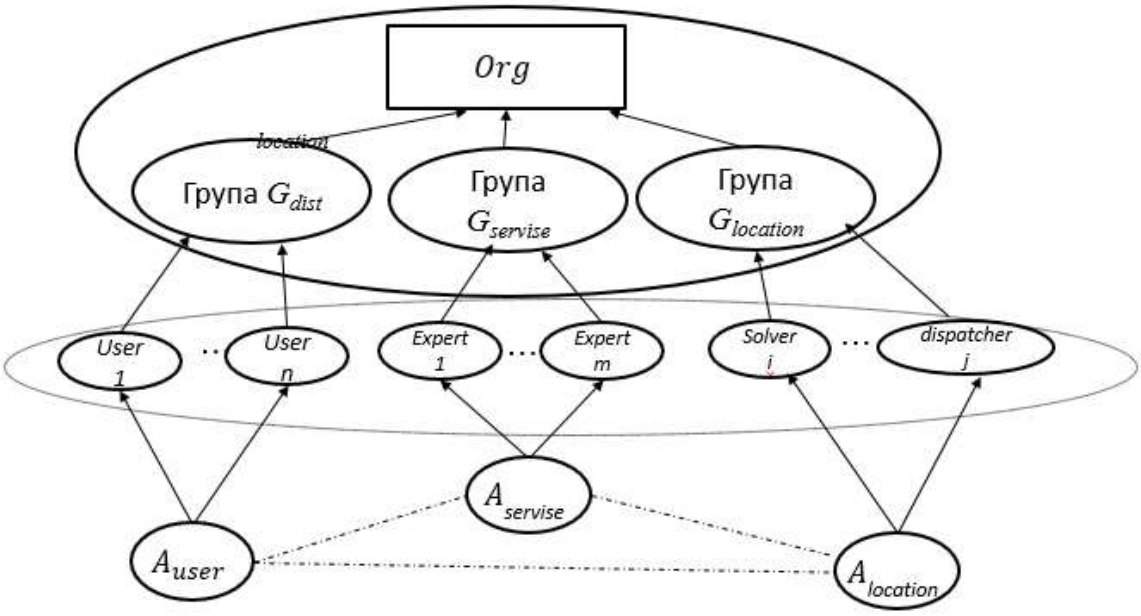

Рисунок 4.7 – Загальна структура IoT системи оперативного реагування

Організаційна модель подана групами $Org = G_{dist}, G_{servise}, G_{location}$.

Маленькі овали позначають ролі R; великі овали – групи Org (G_{dist} –

структурна група користувачів; $G_{servise}$ – структурна група надання сервісу; $G_{location}$ – структурна група, яка визначає місцезнаходження користувача;) суцільні лінії позначають зв'язки між ролями, пунктирні лінії – міжгрупові зв'язки; Агенти групи Org контролюють рольові, групові та організаційні характеристики.

Мультиагентна система подається трійкою

$$MAS = \{A, E, Org\}, \tag{4.2}$$

де $A = \{A_{user}, A_{service}, A_{\text{ДРР}}\}$ – множина агентів, що функціонують в агентному середовищі, яке являє собою програмну платформу; E – множина станів зовнішнього середовища; Org – зв'язки між агентами й оточенням, які надані організаційної моделлю.

Для динаміки MAS використовується така умова:
Визначається місцезнаходження кожного користувача, після деякої тимчасової затримки його запит має обслуговуватися належним чином:

Якщо в момент часу t1 *агент* A_{user} *видає* ⟶ *запит на отримання послуги агентові* A_{expert}

Тоді в момент часу t2 після t1 агент A_{expert} *посилає* ⟶ *пропозицію агентові* $A_{ДРР}$ *визначити місцезнаходження користувача*

у момент часу t3 *після t2 агент* $A_{ДРР}$ *видає* ⟶ *агентові* A_{expert} *місце локації*

у момент часу t4 після t3 агент A_{expert} *посилає* ⟶ *пропозицію агентові* A_{user} *підключити вимірювальні прилади та камеру відеоспостереження*

у момент часу t5 після t4 агент A_{user} *посилає* ⟶ *агентові* A_{expert} *на обробку інформації від підключених пристроїв користувача*

у момент часу t6 після t5 агент A_{expert} *посилає* ⟶ *агентові* A_{user} *повідомлення про використання сервісу*

4.3 Метод кооперації агентів IoT-системи

Організація взаємодії між агентами так.

В мультиагентній системі (4.2) кожний агент описується таким набором

$$A_i = \{A_i^a, E_i^e, Feaa, A_i^s, Fase, Fasaa\},$$

де A_i^a – множина дій агентів:

i=$_{user}$: «Агент користувача» A_{user} збирає інформацію про користувача веб-ресурсу та повертає результат;

i=$_{expert}$: «Агент експерта» A_{expert} звертається за наданням послуг;

i=$_{ДРР}$: «Агент знаходження місця розташування;

E_i^e – множина станів зовнішнього середовища;

$Feaa: A_i^a \times E_i^e \to 2^{A_i^a}$ – функції поведінки зовнішнього середовища;

A_i^s – множина внутрішніх станів агентів (передає, отримує або очікує повідомлення);

$Fase: A_i^s \times E_i^e \to A_i^s$ – функції відновлення стану агента;

$Fasaa: A_i^s \to E_i^e$ – функції прийняття рішень для здійснення дій агентом за поточним внутрішнім станом.

Планувати дії агент може в такий спосіб

$$AP = \{A_i^a, P_i^{ap}, I_i^{ap}, \rho_{ap}, \iota_{ap,0}\}, \quad (4.3)$$

де P_i^{ap} – множина сприйняття агентом станів зовнішнього середовища; I_i^{ap} – підмножина внутрішніх станів агента, що є частиною множини внутрішніх станів

$$I = I_i^{ap} \times I'; \quad (4.4)$$

$\rho_{ap} \subseteq P_i^{ap} \times I_i^{ap} \times I_i^{ap} \times A_i^a$; – відношення переходів, що визначає за сприйняттям p_i^{ap} поточне сприйняття зовнішнього середовища та поточний внутрішній стан плану $\iota_i^{ap\prime}$; $\iota_{ap,0} \in I_i^{ap}$ – початковий стан агента.

Результатом планування є множина ланцюжків переходів агентів з початкового стану в кінцевий, що реалізує поставлену мету.

4.4 Імітаційне моделювання

Для того, щоб показати як мультиагентна система функціонує з використанням організаційної моделі, виконано моделювання, засноване на спостереженнях за пацієнтом за показниками датчиків. Адаптаційні процеси моделювалися як характеристики нижчого рівня (наприклад, рольові характеристики), які були переведені в здійсненну підмножину TTL, називається leadsto [55]. Спрощений формат leadsto дозволяє моделювати прямі тимчасові залежності між двома характеристиками стану в такий спосіб. Нехай α і β –характеристики стану ‘кон’юнкції літералів’ (де літерал – це атом або узгодження атома), і e, f, g, h – ненегативні дійсні числа. У мові *leads to* вираз $\alpha \rightarrow\rightarrow \varepsilon_{, f, g, h} \beta$ означає:

«якщо характеристика стану α знаходиться на певному інтервалі часу із тривалістю g, тоді після деякої затримки (між e і) характеристика стану β перебуватиме на певному інтервалі часу довжиною h»

Реалізована приватна постановка завдання – нагляд за рівнем артеріального тиску пацієнта та міри для збереження здоров’я;

Для моделювання та реалізації використана мова LEADSTO, Leads To Editor, а також TTL Editor для перевірки правильності моделювання та побудови правильних слідів.

У завданні, що пов’язано з артеріальним тиском, змінні пов’язані саме з тиском: high_stress_lvl, normal_stress_lvl, low_stress_lvl. Також, змінні для позначення того, що саме потрібно робити з показниками тиску: home_care, call_amb, do_not_call_amb. Задано час для кінцевого моделювання та додано інтервали для кожної зі змінних артеріального тиску (наприклад, 10 одиниць), а також початок сліду та кінець (рис. 4.8). Кінець сліду означає початок наслідку для даної змінної. Змінна high_stress_lvl сигналізує, що потрібно викликати швидку та надавати негайну допомогу пацієнту пацієнту за визначеною адресою.

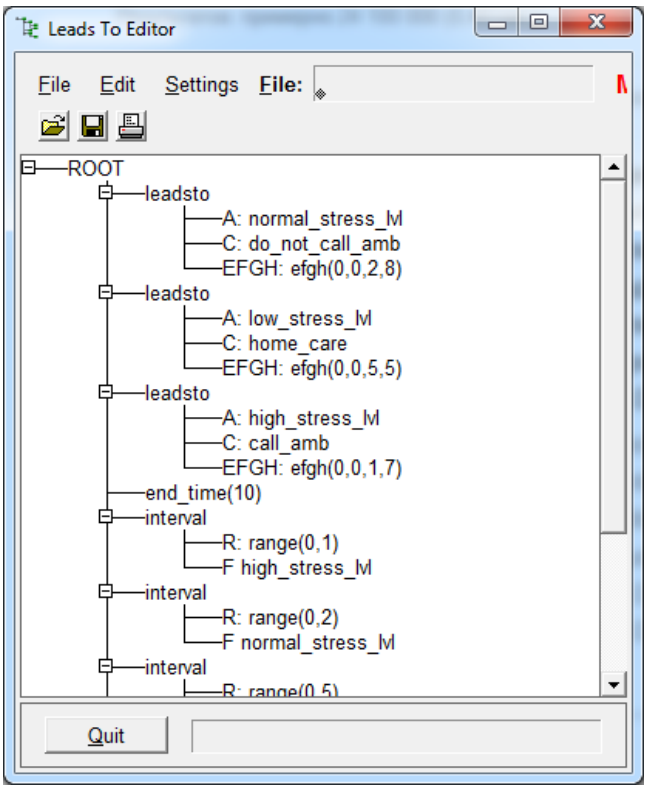

Рисунок 4.8 – Часові інтервали для усіх Leadsto-умов

Усіма часовими відмітками пояснено, що якщо вираз носитиме правдивий характер з тривалістю *g*, то весь вираз матиме значення тривалістю *h* із затримкою 0.0. Для перевірки коректної роботи специфікації запускається LEADSTO Simulation Tool і, якщо під час введення специфікації не виникло ніяких помилок, то побачимо такий результат (рис.4.2, 4.3). Розроблена експериментальна система добре узгоджується із запропонованою Cloud-Fog-Dew Architecture для сервіс-орієнтованих систем [56].

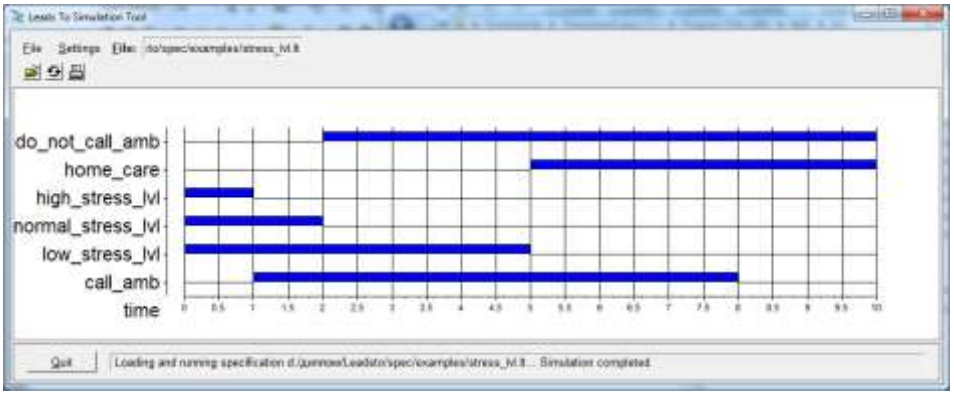

Рисунок 4.9– Успішно виконана специфікація (1)

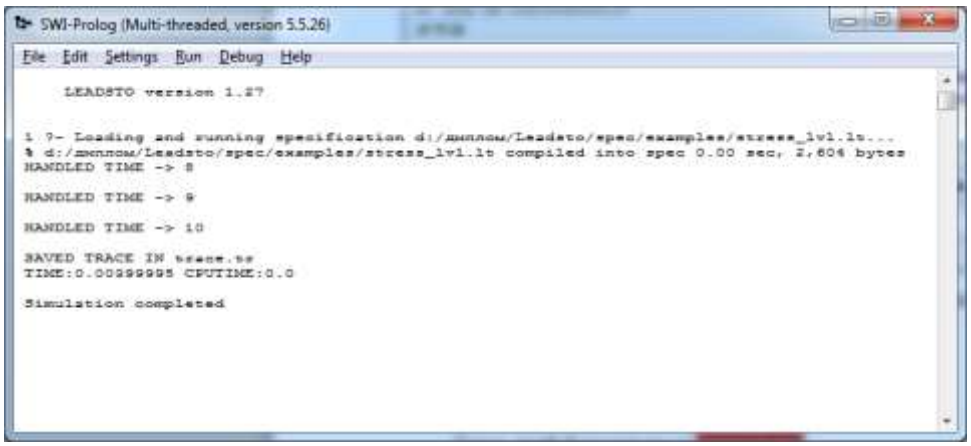

Рисунок 4.10 – Успішно виконана специфікація (2)

Ця система включає чотири типи програмного забезпечення.

1. Персональне програмне забезпечення встановлюється на особистому цифровому пристрої пацієнта, та взаємодіє з Hadware (датчики, вимірювальні пристрої тощо), які збирають інформацію про пацієнта та оцифровану інформацію відправляє Master серверу.

2. Програмне забезпечення обслуговуючого персоналу з’єднується з Master сервером, і отримує дані відповідно до вимог користувача. Це програмне забезпечення дозволяє взаємодіяти з даними про стан пацієнта не залежно від місця розташування спостерігача.

3. Серверне програмне забезпечення встановлено на Master сервері на платформі .NET Framework із застосуванням об’єктно-орієнтованої мови програмування C #. Воно обробляє і зберігає дані про пацієнтів. Для зберігання інформації про користувачів обрана база даних MS SQL Server. Підключення до бази даних здійснюється за допомогою ORM (Object-Relational Mapping).

4. Резервне копіювання даних і їх відтворення це механізм, що забезпечує безпеку даних при збої в роботі серверів.

Веб-інтерфейс системи реалізований за допомогою мови гіпертекстової розмітки HTML, каскадних таблиць стилів CSS і скриптової мови JavaScript. Мова JavaScript дозволила отримати максимально повну інформацію про користувача, яка застосовувалася для автоматичної локалізації сторінки, а також розміри дисплея, що дозволило автоматично перебудовувати ресурс

залежно від них.

Реалізована окрема постановка задачі первинної діагностики гіпертонії на підставі показань цифрового вимірювача артеріального тиску. У процесі моніторингу пацієнт віддалено відповідає на запитання, наведені в анкеті. На підставі відповідей формується вектор опитування, який разом з результатом обробки тонометра надходить на вхід нейронної мережі для постановки первинного діагнозу. Залежно від отриманих результатів мобільний агент A_{doctor} посилає рекомендації про лікування, дає розпорядження обслуговуючому персоналу або здійснюється дзвінок в швидку допомогу.

Критерієм ефективності розробки є задоволення вимог

$$k: \forall(D_i \in D_H)[(\tau^a < \tau$$

$$max^{p^{max}} \& (\tau^r < \tau^{max^{min_D}},$$

де D_i – підзадача спільної справи надання сервісу D_H; τ^a – час адаптації локального сайту, τ^p – час обробки медико-діагностичних показників, τ^r – час реакції системи; ϖ – ступінь релевантності відображеної інформації.

Багато людей не приділяють належної уваги своєму здоров'ю, і часто незаслужено відтягують відвідування лікаря, що може призвести до тяжких наслідків. Запропонована система допоможе пацієнтам, які перебувають на обліку в медичному закладі, своєчасно отримати медичну допомогу за рахунок віддаленого автоматичного контролю стану здоров'я. Тестування показало, що розроблена система виявлення критичного стану здоров'я дозволяє підвищити точність діагностики на 8.7%.

ВИСНОВКИ

Концепція Інтернету речей, активно розвивається в останні роки, призначена для вирішення великомасштабних завдань, які втягують в процес вирішення величезна кількість об'єктів фізичного, соціального та віртуального світу. Ці об'єкти можуть бути джерелами вельми різнорідної інформації, засобами її обробки і прийняття рішень, виконавчими органами, які впливають, в свою чергу, на об'єкти фізичного, соціального та віртуального світу. До таких об'єктів належать також хмарні ресурси і семантичні сервіси Web, доступ до яких можна отримати за допомогою комунікаційного середовища Інтернет. В даний час йде активний пошук концепцій і інтелектуальних IT, які в змозі добре справлятися із завданнями описаного класу, а вони мають велику соціальну і комерційну цінність.

Отже, в ході виконання магістерської роботи було:

– досліджено методи та моделі побудови агентно-орієнтованих систем;

– описано рішення для великомасштабної розподіленої обробки даних у вигляді хмарних обчислень. Показано ефективне використання такого середовища, управління високою складністю і забезпечення відповідного рівня якості обслуговування для сучасних систем моніторингу;

– розроблено метод знаходження місце розташування загальних пристроїв для Інтернет речей.

Імітаційне моделювання підтвердило доцільність використання запропонованої моделі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. H. Yu, Z. Shen, C. Leung. From Internet of Things to Internet of Agents. 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, IEEE Computer Society, 2013 P. 1054-1057. DOI 10.1109/GreenCom-iThings-CPSCom.2013.179.
2. Luck M., et al. Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing). AgentLink. 2005. –URL: http://www.agentlink.org/roadmap/
3. Городецкий В.И., Карсаев О.В. Самоорганизация группового поведения кластера малых спутников распределенной системы наблюдения // Известия Южного федерального университета. Технические науки. 2017. № 3
4. JCtrans.Net. – URL : http://www.jctrans.net.
5. RosettaNet. –URL : www.rosettanet.org.
6. База данных проектов программ Европейской комиссии FP-5, FP-6 и FP-7 и Horizon- 2020 и поисковая система. – URL : http://cordis.europa.eu/search/advanced_en?projects
7. Rzhevsky G., Skobelev P. Managing complexity. WIT Press, London-Boston. 2014.156 p.
8. Mueller J., Fisher K. Application Impact of Multiagent Systems and Technologies: A Survey // In Agent-Oriented Software Engineering book series. Springer. 2013. P. 1-26.
9. De Loach S. A Moving multi-agent systems from research to practice // Int. J. Agent- Oriented Software Engineering. Vol. 3. No. 4. 2009. P. 378-382.
10. Leitao P., Vrba P. Recent Developments and Future Trends of Industrial Agents // Proceedings of HoloMAS-2011, Holonic and Multi-Agent Systems for Manufacturing, Volume 6867 of the series Lecture Notes in Computer Science. Springer Verlag. 2011-2012. P. 15-28.

11. JADE. – URL : http://sharon.cselt.it/projects/jade/

12. Nomadic Agent Working Group, – URL : http://www.fipa.org/subgroups/P2PNA-WG-docs/P2PNA-Spec-Draft0.12.doc.

13. SMITH G. Contract Net Protocol: High-level Communication and Control in a Distributed Problem Solver // IEEE Transactions on Computers, C-29 (12). 1980. P. 1104-1113.

14. Bukhvalov O., Gorodetsky V., Karsaev O, Koudryavtsev G., Samoylov V. Privacy- Preserved Distributed Coordination of Production Scheduling in B2B Networks: A Multi-agent Approach. //Proc. 7th IFAC Conference on Manufacturing Modeling, Management, and Control, 2013, Материалы конференции, Volume 7. Part 1. P. 2122-2127.

15. Allen, J. F., Kautz, H. A., Pelavin, R. N., & Tenenberg, J. D. (1991). Reasoning About Plans. Morgan Kaufmann Publishers, Inc., San Mateo, California.

16. Messina F. et al. A multi-agent protocol for service level agreement negotiation in cloud federations //International Journal of Grid and Utility Computing. 2016. Т. 7, №. 2. P. 101–112.

17. Messina F. et al. A multi-agent protocol for service level agreement negotiation in cloud federations //International Journal of Grid and Utility Computing. 2016. Т. 7, №. 2. P. 101–112.

18. Valogianni K. A multiagent approach to variable-rate electric vehicle charging coordination / K. Valogianni, W. Ketter, J. Collins // Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. – International Foundation for Autonomous Agents and Multiagent Systems, 2015. P. 1131-1139.

19. Wang G., Yang J., Xu J. Granular computing: from granularity optimization to multi-granularity joint problem solving //Granular Computing. 2017. Т. 2, №. 3. P. 105–120.

20. Wu B. F. Adaptive Feature Mapping for Customizing Deep Learning Based Facial Expression Recognition Model / B. F. Wu, C. H. Lin // IEEE Access. 2018. Т. 6. P. 12451-12461.

21. Wooldridge, M., Ciancarini, P., 2001. Agent-oriented software engineering: The state of the art. Agent-Oriented Software Engineering. Springer Verlag. p. 1–28.

22. Wooldridge, M.J., 2009. Introduction to multiagent systems, second ed. Wiley.

23. Севідов П.М. Прийняття колективних рішень в мультиагентних системах / П.М. Севідов. // Міжнародний науковий журнал "Інтернаука". – 2018. – №7. – С. 81–101.

24. ИТ инфраструктура и телекоммуникации – URL : http://www.anylogic.ru/consulting/information-and- telecommunication-networks.

25. Системи управління – URL : http://goo.gl/VBxMa.

26. Subversion – URL : http://uk.wikipedia.org/wiki/Subversion.

27. Управління проектами – URL : http://uk.wikipedia.org/wiki/Управління_проектами

28. Діаграма Ганта – URL : http://uk.wikipedia.org/wiki/Діаграма_Ганта

29. Smart Python Agent Development Environment – URL : https://github.com/javipalanca/spade.

30. S. Railsback and V. Grimm. /Agent-based and Individual-based Modeling: A Practical Introduction. // Princeton University Press, 2011.

31. Odd Myklebust, Enterprise Modelling supported by Manufacturing Systems Theory: Doctoral dissertation, Norwegian University of Science and TechnologyDepartment of Production- and Quality Engineering.

32. Baldoni, M., Baroglio, C., Mascardi, V., Omicini, A., Torroni, P., 2010. Agents, multi-agent systems and declarative programming: What, when, where, why, who,how?

33. Dovier, A., Pontelli, E. (Eds.), A 25-Year Perspective on Logic Programming: Achievements of the Italian Association for Logic Programming, GULP. Springer,pp. 204–230.

34. Bellifemine, F., Caire, G., Greenwood, D., 2007. Developing multi-agent systems with JADE. Wiley Series in Agent Technology. John Wiley & Sons.

35. Bergenti, F., Caire, G., Gotta, D., 2015. Large-scale network and service management with WANTS. Industrial Agents: Emerging Applications of SoftwareAgents in Industry. Elsevier. pp. 231–246.

36. Bergenti, F., Gleizes, M.P., Zambonelli, F. (Eds.), 2004. Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook.Boella, G., van der Torre, L.W.N., Verhagen, H., 2007. Introduction to normative multiagent systems. In: Boella, G., van der Torre, L.W.N., Verhagen, H. (Eds.), Normative Multi-agent Systems, 18–23, March 2007, Internationales Begegnung- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

37. Bordini, R.H., Braubach, L., Dastani, M., et al., 2006. A survey of programming languages and platforms for multi-agent systems. Informatica.

38. Borah S. ANN based human facial expression recognition in color images / S. Borah, S. Konwar // High Performance Computing and Applications (ICHPCA), 2014 International Conference on. – IEEE, 2014. – P. 1-6.

39. Blanchet G. Digital signal and image processing using MATLAB / G. Blanchet, M. Charbit. – Wiley, ISTE, 2006. – T. 4. – 764p.

40. Bychkov I. V. et al. Service-oriented multiagent control of distributed computations //Automation and Remote Control. 2015. – T. 76.,№. 11. P. 2000–2010.

41. Elst L. Van. Agent-Mediated Knowledge Management / Van Elst L., V. Dignum, A. Abecker // Proceedings of the Agent-Mediated Knowledge Management. – Standford, 2003. P. 1-30.

42. Grzonka D. et al. Using a multi-agent system and artificial intelligence for monitoring and improving the cloud performance and security //Future Generation Computer Systems. – 2018. – T. 86. – P. 1106–1117.

43. Аксак Н.Г. Мультиагентная система нейромережевої диагностики и удаленного мониторинга пациента / Н.Г. Аксак // Інформаційні технології: проблеми та перспективи: монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г., 2017. – С. 325–340.

44. Аксак Н.Г. Разработка системы персонализации специализированного веб-портала / Н.Г. Аксак // Радіоелектроніка, інформатика, управління. – 2018. – № 1. – С. 91–99. – DOI 10.15588/1607-3274-2018-1-11

45. Аксак Н.Г. Организационный вид медицинской диагностической мультиагентной системы / Н.Г. Аксак // Системы и средства искусственного интеллекта. – 2014. – №1. – С. 8–10.

46. Axak N. Development of multi-agent system of neural network diagnostics and remote monitoring of patient / N. Axak // Eastern-European Journal of Enterprise Technologies. – 2016. – № 4/9 (82) – P. 4–11.

47. Axak N. Cloud-fog-dew Architecture for Personalized Service-oriented Systems / N. Axak, D. Rosinskiy, O. Barkovska, I. Novoseltsev // The 9th IEEE International Conference on Dependable Systems, Services and Technologies, DESSERT'2018, Kyiv, Ukraine – 2018. – P. 80–84.

48. Аксак Н. Г. Архитектура сервис-ориентированной системы распределенной обработки больших данных / Н. Г. Аксак, Д. Н. Росинский // Information-Management Systems and Technologies: VII Intern. Scientific Conf., 2018: materials of the conf. – Odessa, Ukraine, 2018. – С. 195-196.

49. Аксак Н. Г. Взаимодействие агентов в системе удаленного контроля за пациентом / Н. Г. Аксак // Обчислювальний інтелект (результати, проблеми, перспективи): III Міжнар. наук.-практ. конф., 2015 р.: матер. конф. – Черкаси, 2015. – С. 166-167.

50. Аксак Н. Г. Дистанционный мониторинг пациента на основе концепции агент-группа-роль / Н. Г. Аксак, Н. М. Кораблев // Системні дослідження та інформаційні технології. 2018. №. 3. С. 7-18.